

# Introduction To Data Science

---

Building Recommender Systems



# Introduction

---

## Chapter 1



# Course Chapters

- **Introduction**
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Chapter Topics

## Introduction

- **About this course**
- About Cloudera
- Course logistics
- Introductions

# Course Objectives

---

**After successfully completing this course, you will be able to:**

- **Describe the role and responsibilities of a data scientist**
- **Explain several ways in which data scientists create value for their organizations, using several industries as examples**
- **Locate and acquire data from diverse sources**
- **Use transformation and normalization techniques**
- **Determine the most appropriate type of analysis to perform for a given problem**
- **Be able to implement an automated recommendation system**
- **Develop, evaluate, and refine scoring systems for recommenders**

## Course Objectives (cont'd)

---

- Understand important considerations for working at scale
- Identify meaningful, actionable, and business-oriented results from the analysis performed

# Chapter Topics

## Introduction

- About this course
- **About Cloudera**
- Course logistics
- Introductions

## About Cloudera

---

- **Cloudera is “The commercial Hadoop company”**
- **Founded by leading experts on Hadoop from Facebook, Google, Oracle and Yahoo**
- **Provides consulting and training services for Hadoop users**
- **Staff includes committers to virtually all Hadoop projects**

# Cloudera Software

---

- **CDH**

- A set of easy-to-install packages built from the Apache Hadoop core repository, integrated with several additional open source Hadoop ecosystem projects
- Includes a stable version of Hadoop, plus critical bug fixes and solid new features from the development version
- 100% open source

- **Cloudera Manager, Free Edition**

- The easiest way to deploy a Hadoop cluster
- Automates installation of Hadoop software
- Installation, monitoring and configuration is performed from a central machine
- Manages up to 50 nodes
- Completely free

# Cloudera Enterprise

---

- **Cloudera Enterprise**

- Complete package of software and support
- Built on top of CDH
- Includes full version of Cloudera Manager
  - Install, manage, and maintain a cluster of any size
  - LDAP integration
  - Includes powerful cluster monitoring and auditing tools
    - Resource consumption tracking
    - Proactive health checks
    - Alerting
    - Configuration change audit trails
    - And more
- 24 x 7 support

# Cloudera Services

---

- **Provides consultancy services to many key users of Hadoop**
  - Including AOL Advertising, Samsung, Groupon, NAVTEQ, Trulia, Tynt, RapLeaf, Explorys Medical...
- **Solutions Architects are experts in Hadoop and related technologies**
  - Several are committers to the Apache Hadoop project
- **Provides training in key areas of Hadoop administration and development**
  - Courses include System Administrator training, Developer training, Hive and Pig training, HBase Training, Essentials for Managers
  - Custom course development available
  - Both public and on-site training available

# Chapter Topics

## Introduction

- About this course
- About Cloudera
- **Course logistics**
- Introductions

# Logistics

---

- Course start and end times
- Lunch
- Breaks
- Restrooms
- Wi-Fi access
- Virtual machines
-

# Chapter Topics

## Introduction

- About this course
- About Cloudera
- Course logistics
- **Introductions**

# Introductions

---

- **About your instructor**

- **About you**

- Where do you work and what do you do there?
- Do you have a scientific or mathematical background?
- What programming languages have you used?
- Are you experienced with Apache Hadoop or related tools?
- What do you expect to gain from this course?

# Data Science Overview

---

## Chapter 2



# Course Chapters

- Introduction
- **Data Science Overview**
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Data Science Overview

---

## In this chapter you will learn

- **What data science is**
- **Why data scientists are in demand**
- **What data products they help to create**
- **Which skills a successful data scientist must possess**

# Chapter Topics

## Data Science Overview

- **What is data science?**
- The growing need for data science
- The role of a data scientist
- Review questions
- Essential points
- Conclusion

# What is a Data Scientist?

---

- There's no single standard definition
- Let's look at how some in the industry describe the role...

## What is a Data Scientist? (cont'd)

---

“They are half hacker, half analyst,  
they use data to build products and find insights.”

Monica Rogati  
Senior Data Scientist  
LinkedIn

## What is a Data Scientist? (cont'd)

---

“A data scientist can find patterns in data that you haven’t sent them to find”

Tom Wheeler  
Senior Vice President  
ClickFox

## What is a Data Scientist? (cont'd)

---

“Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.”

Josh Wills  
Sr. Director of Data Science  
Cloudera

## Our Definition

---

**Data science is a multidisciplinary field that combines skills in software engineering and statistics with domain experience to support the end-to-end analysis of large and diverse data sets, ultimately uncovering value for an organization and then communicating it to stakeholders as actionable results.**

# Data Products

---

- One frequent goal of data science is to create *data products*
- Data products have three important characteristics
  - They are built from data
  - The very act of using them generates new data
  - This new data can be used to improve them

## Data Product Examples: Google

---

- **Google wasn't the first Web search engine when it launched**
  - Yahoo!, Lycos, AltaVista and Excite were market leaders
  - But search results were cluttered and often irrelevant
- **But Google had an exceptional data product**
  - The PageRank algorithm produced highly relevant results
  - PageRank allowed Google to quickly dominate the market
- **Google extended this success with additional data products**
  - AdSense (targeted ads for Web properties)
  - AdWords (targeted ads for search result pages)
  - Google Analytics (service for analyzing Web site traffic)

## Data Product Examples: LinkedIn

---

- **LinkedIn is a popular social network for business**
- **It was instrumental in the emergence of data science**
- **The company offers several data products**
  - LinkedIn Network Updates (e-mail news summary of your connections)
  - People You May Know (list of potential connections)
  - InMaps (visualization of your connections)

## Data Product Examples: Amazon

---

- **Amazon popularized the use of product recommendations**
  - Based on collaborative filtering, as we'll see later
- **They then added variations on this to offer even more insight**
  - What items do customers buy after viewing this item?
  - Customers who bought this item also bought...
- **Other data products from Amazon include**
  - Find visually similar items
  - Statistically Improbable Phrases
  - Product advertising API

## Data Product Examples: eBay

---

- **eBay offers data products to users, marketers, and developers**
  - Intelligent spelling correction for auction titles
  - Sorry you didn't win, here are auctions for similar items
  - eBay Market Data
  - Milo Open API (real-time local product availability)

# Chapter Topics

## Data Science Overview

- What is data science?
- **The growing need for data science**
- The role of a data scientist
- Review questions
- Essential points
- Conclusion

# Why is Data Science in Demand?

---

- **The term “data scientist” has only recently become popular**
- **Is this a new field or just newly important?**
  - Data science is the intersection of several fields
  - Each of those fields has existed for years
  - The *specific combination* of them has recently become important

# Why is Data Science in Demand? (cont'd)

---

- **Why is this combination important now?**
- **Data drives the demand for data scientists**
  - We're generating more data than ever
  - We're generating new data faster

# The Data Deluge

---

- **We're generating more data than ever**
  - Financial transactions
  - Sensor networks
  - Application logs
  - e-mail and text messages
  - Social media

# The Data Deluge (cont'd)

---

- **And we're generating new data faster than ever**
  - Automation
  - Ubiquitous Internet connectivity
  - User-generated content
- **For example, every day**
  - Twitter processes 340 million messages
  - Trend Micro processes six TB of data to identify new security threats
  - Facebook users generate 2.7 billion comments and “Likes”
  - The Large Hadron Collider produces more than 68 TB of data

# Cost of Large-Scale Storage

---

- Fortunately, the size and cost of storage has kept pace
  - Capacity has increased while price has decreased

Year	Capacity (GB)	Cost per GB (USD)
1992	0.08	\$3,827.20
1997	2.10	\$157.00
2002	80.00	\$3.74
2007	750.00	\$0.35
2012	3,000.00	\$0.05

- We can now afford to retain what we previously discarded
  - Although storage is important, only analysis will yield value
  - But what value can we produce with all of this data?

# Value of Large-Scale Analysis

---

- **This data has many valuable applications, including**
  - Product recommendations
  - Marketing analysis
  - Demand forecasting
  - Fraud detection
- **The more data we have, the more valuable our applications become**
- **Moving from data collection to data product requires many skills**

# Chapter Topics

## Data Science Overview

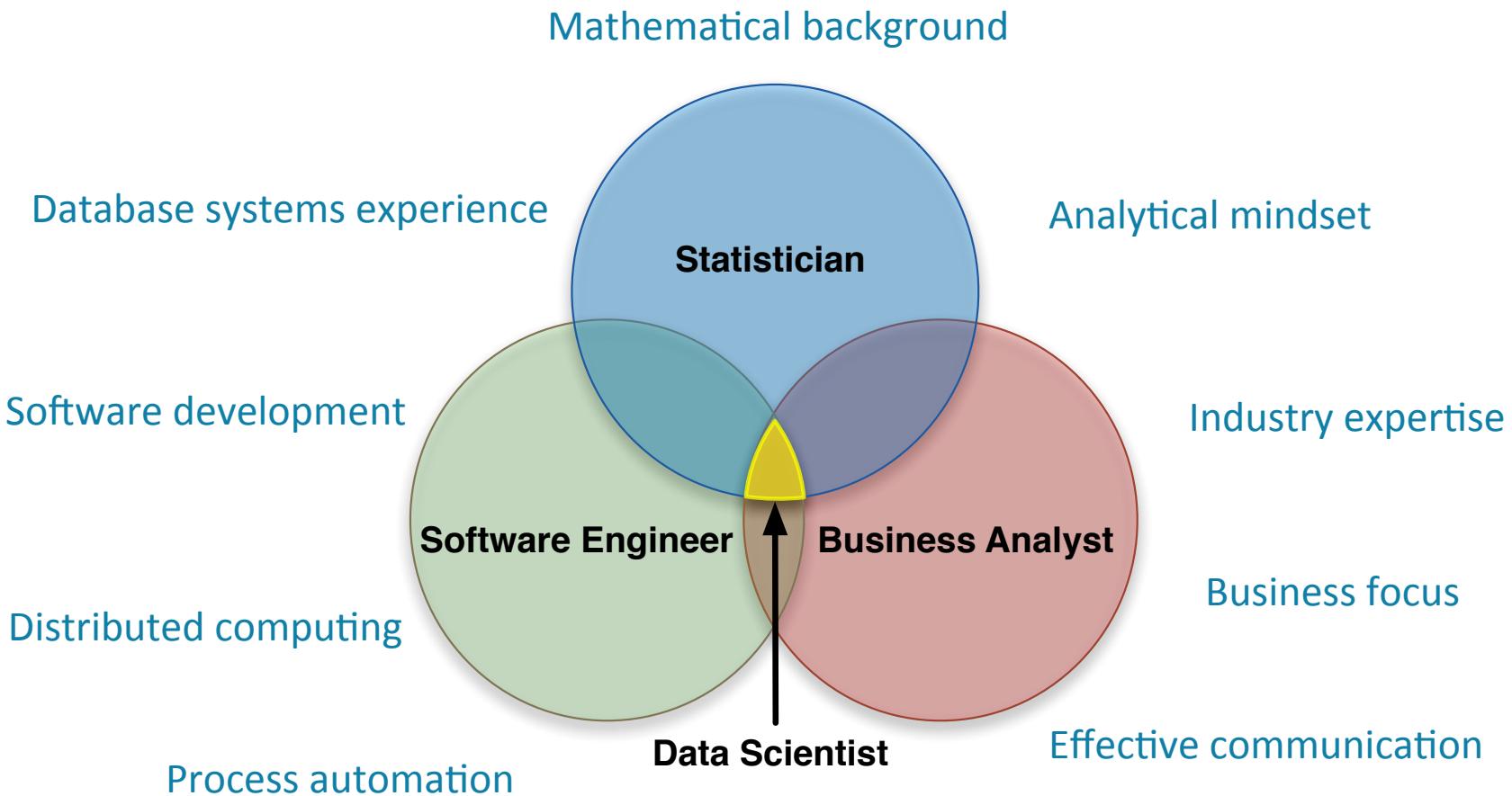
- What is data science?
- The growing need for data science
- **The role of a data scientist**
- Review questions
- Essential points
- Conclusion

# Data Science Skills

---

- **Data scientists have an unusual combination of skills**
- **An effective data scientist will possess**
  - An analytical mindset
  - A broad mathematical background
  - Database systems experience
  - Skill with software engineering, particularly at scale
  - Ability to communicate effectively with both business and technical audiences
  - Expertise in a particular industry
- **How do these map to longstanding roles in an organization?**

# Data Scientists Are Found at the Confluence



# How Does a Data Scientist Differ...

---

- **From a software engineer?**

- Data scientists don't develop general purpose software
  - They develop software to help them solve problems
  - Data scientists tend to focus more on scripting and automation

- **From a data analyst or statistician?**

- These roles usually rely on curated data and predefined tools
  - If a tool is lacking or data is incomplete, questions are left unanswered
  - Data scientists instead collect more data and write new tools
    - They also typically work with huge, disparate, and “dirty” data

- **From a business analyst?**

- Both have business focus, so they may ask similar questions
  - Data scientist has technical skills to find answers without help

# Data Scientists Within Organizations

---

- **Data Scientist is a relatively new term**
  - First used in a National Science Board publication in 2005
  - Popularized after teams were created at Facebook and LinkedIn
- **The name given to this field often varies by industry**
  - Bioinformatics
  - Computational social science
  - Research or statistical science
  - Predictive modeling
- **Analysts have traditionally been internal consultants**
  - Data scientists often serve a similar role
- **Some organizations use data scientists for product development**

# Chapter Topics

## Data Science Overview

- What is data science?
- The growing need for data science
- The role of a data scientist
- **Review questions**
- Essential points
- Conclusion

## Review Questions

---

- **How would you define data science?**
- **What other examples of data products have you seen?**
- **Imagine that you work for a university. What data products might you create?**

# Chapter Topics

## Data Science Overview

- What is data science?
- The growing need for data science
- The role of a data scientist
- Review questions
- **Essential points**
- Conclusion

## Essential Points

---

- Data science is the combination of several skills, including mathematics, software engineering, communications, and domain experience
- Data products are built from data and produce new data as they're used, allowing them to be further improved
- Defining differences between data analysts and data scientists are the latter's ability to work with massive data sets and to develop new tools to collect, transform and analyze data

# Chapter Topics

## Data Science Overview

- What is data science?
- The growing need for data science
- The role of a data scientist
- Review questions
- Essential points
- **Conclusion**

# Data Science Overview

---

## In this chapter you have learned

- **What data science is**
- **What data products they help to create**
- **Why data scientists are in demand**
- **Which skills a successful data scientist must possess**

# Bibliography

---

The following offer more information on topics discussed in this chapter

- **The Rise of the Data Scientist**

- <http://tiny.cloudera.com/dscc02a>

- **Building Data Science Teams**

- <http://tiny.cloudera.com/dscc02b>

- **Data Scientist: The Sexiest Job of the 21st Century**

- <http://tiny.cloudera.com/dscc02c>

- **Quora: How do I become a data scientist?**

- <http://tiny.cloudera.com/dscc02d>

- **Seismic Data Science**

- <http://tiny.cloudera.com/dscc02e>

# Use Cases

---

## Chapter 3



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases**
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Use Cases

---

## In this chapter you will learn

- How data science is being applied in different industries
- How data science is used to increase revenue
- How data science is used to reduce costs
- How data science is used to save lives

# Chapter Topics

## Use Cases

- **Finance**
- Retail
- Advertising
- Defense and Intelligence
- Telecommunications and Utilities
- Healthcare and Pharmaceuticals
- Review questions
- Essential points
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

“How can we protect our company  
against loss from criminals?”

# Finance: Fraud Detection

---

- **There are many kinds of financial fraud, including**
  - Securities trading
  - Insurance
  - Credit card
- **Credit card fraud detection is perhaps most widely known**
  - “Please confirm unusual activity we’ve detected on your account”
- **The specific techniques differ and are proprietary**
  - They usually have one thing in common: access to lots of data
- **Solution overview**
  - Log data from multiple sources to HDFS, perhaps via Flume
  - Use machine learning to classify typical customer behavior
    - And to identify deviant behavior worthy of further investigation

“How likely is it that this customer  
will pay as agreed?”

# Finance: Customer Risk Analysis

---

- **It's more profitable to price according to risk than to deny a sale**
  - Higher-risk customers pay more for products
  - Examples include auto loans, credit cards, and insurance policies
- **An accurate risk assessment relies on disparate data**
  - And lots of it!
  - Past purchases, payment history, clickstream data, call logs, etc.
- **Solution overview**
  - Ingest data from many sources into Hadoop storage (HDFS)
  - Correct data errors and use consistent representation of attributes
  - Merge data into a single view of the customer
  - Create initial risk model based on historic data
  - Continually refine the model based on actual risk encountered

# Chapter Topics

## Use Cases

- Finance
- **Retail**
- Advertising
- Defense and Intelligence
- Telecommunications and Utilities
- Healthcare and Pharmaceuticals
- Review questions
- Essential points
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

“Which products will this customer enjoy?”

# Product Recommendations

---

- **Many sites use collaborative filtering (CF) to suggest products**
  - Increases sales, attracts customers and improves satisfaction
  - Well-known examples include Amazon, Netflix, and Last.fm
- **We are going to study this in depth during the course**
  - Collaborative filtering has many practical applications
- **Solution overview**
  - Capture customer activity and preference data
  - Find other customers with similar preferences
  - Determine which products these customers rated highly
  - Recommend these products to the customer

“Which products will this customer likely buy?”

“How will a discount  
improve the odds of purchase?”

# Predict and Incentivize Purchases

---

- **Instead of simply recommending products, motivate purchases**
  - Retailers can increase sales using highly-targeted offers
- **Could use a variety of historic data for customers, including**
  - Products customer has purchased
  - Which products customer later returned or exchanged
- **Solution overview**
  - Purchase history is an implicit source of preference data
    - Buying something is an upvote
    - Returning or exchanging something is a downvote
  - Use CF to predict future purchases based on similar customers
  - Offer customer coupons to incentivize these purchases
  - Use redemption data to improve future predictions

# Chapter Topics

## Use Cases

- Finance
- Retail
- **Advertising**
- Defense and Intelligence
- Telecommunications and Utilities
- Healthcare and Pharmaceuticals
- Review questions
- Essential points
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

“How can we make our advertisements  
more relevant to this user?”

# Serving More Effective Web Ads

---

- **Advertising is a significant source of revenue for Web properties**
  - Pay-per-click is much more valuable than pay-per-impression
  - You can therefore boost revenue by increasing clickthrough rate
- **Machine learning techniques are common for ad click prediction**
- **Data typically analyzed includes**
  - Relevance of ad to the search query or site visited
  - Overall historical effectiveness of a given ad
  - Time of day and day of the week
- **However, all of these factors are independent of the user**
  - They essentially predict what an average person would click
  - Collaborative filtering can help predict a *specific* person's clicks
  - This technique increased Yahoo's prediction accuracy > 10%

# Chapter Topics

## Use Cases

- Finance
- Retail
- Advertising
- **Defense and Intelligence**
- Telecommunications and Utilities
- Healthcare and Pharmaceuticals
- Review questions
- Essential points
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

“How can we detect and report information  
that helps keep our personnel safe?”

# Forecasting Insurgent Attacks

---

- **Predicting insurgent attacks is extremely difficult**
  - Insurgents tend to be loosely organized and informal
  - They blend in with the local non-combatant population
  - Detecting their movements and activities is a challenge
- **Past attack data can reveal patterns that predict future events**
- **A team of researchers demonstrated how effective this can be**
  - Analyzed Afghan War Diary documents from WikiLeaks
  - Used data from 2004-2009 events to train a computer model
  - Forward prediction of 2010 events was surprisingly accurate
    - Expected a 128% increase in events for Baghlan province
    - A 120% increase was actually observed in the 2010 data
- **Could be used to provide decision support for mission planning**

# Chapter Topics

## Use Cases

- Finance
- Retail
- Advertising
- Defense and Intelligence
- **Telecommunications and Utilities**
- Healthcare and Pharmaceuticals
- Review questions
- Essential points
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

## Telecommunications and Utilities

---

“How can we actively prevent service outages,  
instead of simply reacting to them?”

# Predicting Equipment Failure

---

- **Aging power, water, and telecom infrastructure in many locations**
  - Too expensive to replace components that still work
  - Cost of failure can be even more expensive
- **Solution overview**
  - Install sensors to gather data (vibration, temperature, etc.)
  - Store this data in Hadoop (using Flume)
  - Analyze to determine leading indicators of failure
  - Use machine learning to classify components likely to fail soon
  - Replace with new components proactively

# Chapter Topics

## Use Cases

- Finance
- Retail
- Advertising
- Defense and Intelligence
- Telecommunications and Utilities
- **Healthcare and Pharmaceuticals**
- Review questions
- Essential points
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

“How can we protect patients from dangerous drug interactions?”

# More Accurate Medication Reconciliation

---

- **A physician needs to know what medicine the patient uses**
  - Drugs can negate one another or cause deadly interactions
- **Unfortunately, this data is not always accurate**
  - Some patients may not recall all medications when asked
  - Others might intentionally omit others (e.g. illegal drugs)
  - A 2007 study documented discrepancies in 80.4% of patients
- **Overall, quite similar to product recommendation use case**
  - Recommenders can be agnostic to what's being evaluated
    - Customer = patient
    - Product = medication
  - Result is a list of medications that similar patients report using
  - Physician can then specifically ask about / test for use of these

# Chapter Topics

## Use Cases

- Finance
- Retail
- Advertising
- Defense and Intelligence
- Telecommunications and Utilities
- Healthcare and Pharmaceuticals
- **Review questions**
- Essential points
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

## Review Questions

---

- **What are some common themes you've observed in these use cases?**
- **Imagine you've been hired by an airline to retain its most profitable customers**
  - What questions would you want to be able to answer?
  - What data would you need to do this?

# Chapter Topics

## Use Cases

- Finance
- Retail
- Advertising
- Defense and Intelligence
- Telecommunications and Utilities
- Healthcare and Pharmaceuticals
- Review questions
- **Essential points**
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

## Essential Points

---

- **Data Science is applicable to many industries**
  - Although industries may differ, techniques are often quite similar
- **Key themes include**
  - Access to large amounts of data
  - Acquisition of several types of data from different sources
  - The ability to analyze this data at scale to find interesting patterns
  - Problem solving focuses on a specific and actionable result

# Chapter Topics

## Use Cases

- Finance
- Retail
- Advertising
- Defense and Intelligence
- Telecommunications and Utilities
- Healthcare and Pharmaceuticals
- Review questions
- Essential Points
- **Hands-On Exercise: Parsing Log Data with Python**
- Conclusion

# Hands-on Exercise: Parsing Log Data with Python

---

- In this Hands-On Exercise, you will learn the basics of Python and the command line utility for accessing Hadoop's Distributed Filesystem (HDFS).
  - You'll use regular expressions in Python to parse a Web server log file – an important source of information for data science projects
  - Please refer to the Hands-On Exercise Manual for instructions on exercise #0

# Chapter Topics

## Use Cases

- Finance
- Retail
- Advertising
- Defense and Intelligence
- Telecommunications and Utilities
- Healthcare and Pharmaceuticals
- Review questions
- Essential points
- Hands-On Exercise: Parsing Log Data with Python
- Conclusion

# Use Case

---

**In this chapter you have learned**

- **How data science is being applied in different industries**
- **How data science is used to increase revenue**
- **How data science is used to reduce costs**
- **How data science is used to save lives**

# Bibliography

---

The following offer more information on topics discussed in this chapter

- **Fraudsters, Outliers, and Big Data**
  - <http://tiny.cloudera.com/dscc03a>
- **Learning Causality for News Event Prediction**
  - <http://tiny.cloudera.com/dscc03b>
- **Personalized Prediction for Sponsored Search**
  - <http://tiny.cloudera.com/dscc03c>
- **Mining Event Logs in Large Scale Systems**
  - <http://tiny.cloudera.com/dscc03d>
- **Medication Reconciliation**
  - <http://tiny.cloudera.com/dscc03e>

# Project Lifecycle

## Chapter 4



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- **Project Lifecycle**
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Project Lifecycle

---

## In this chapter you will learn

- How a data scientist approaches a problem
- Which steps comprise the lifecycle of a data science problem
- About the data science problem we'll address through hands-on exercises

# Chapter Topics

## Project Lifecycle

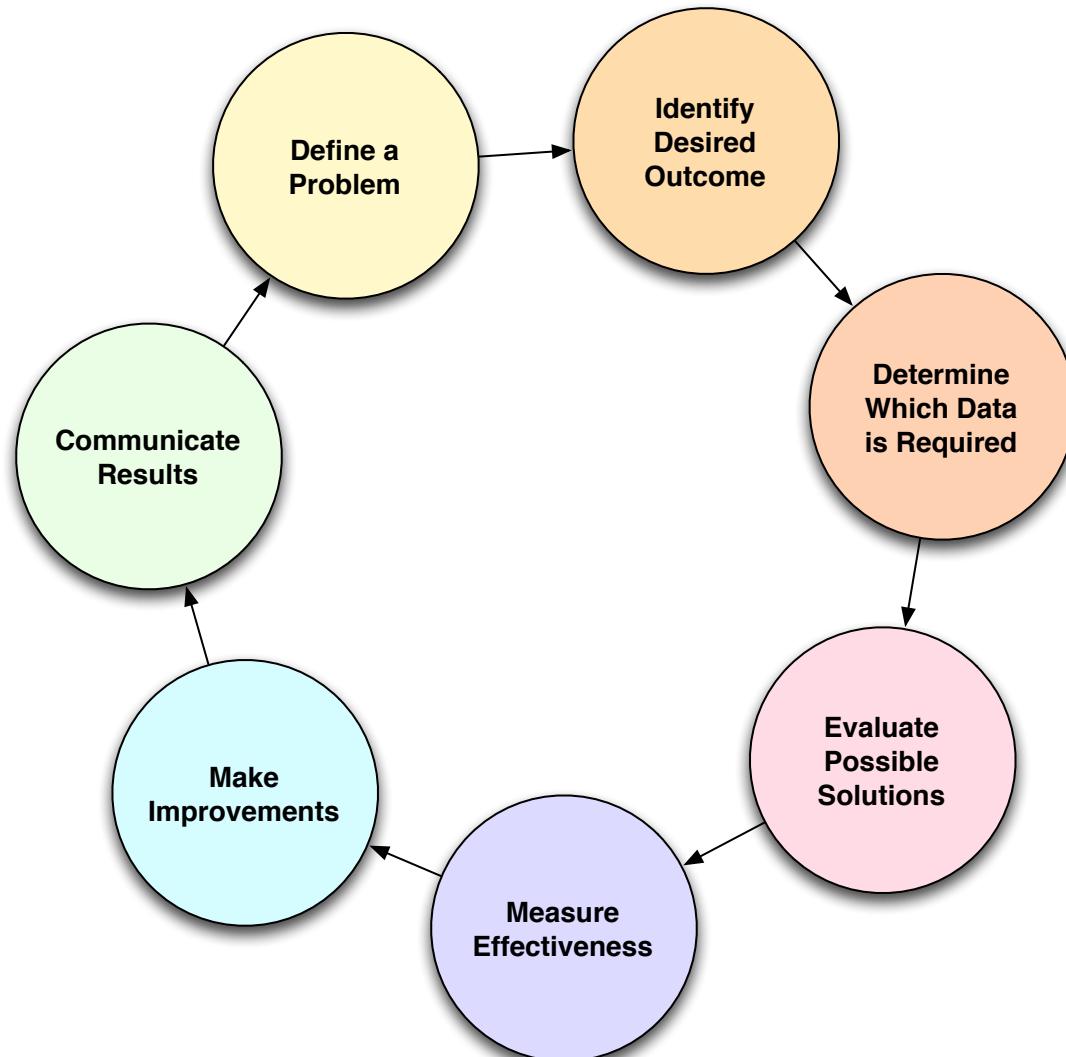
- **Steps in the project lifecycle**
- Hands-On Exercises: scenario explanation
- Review questions
- Essential points
- Conclusion

# Overview of the Project Lifecycle

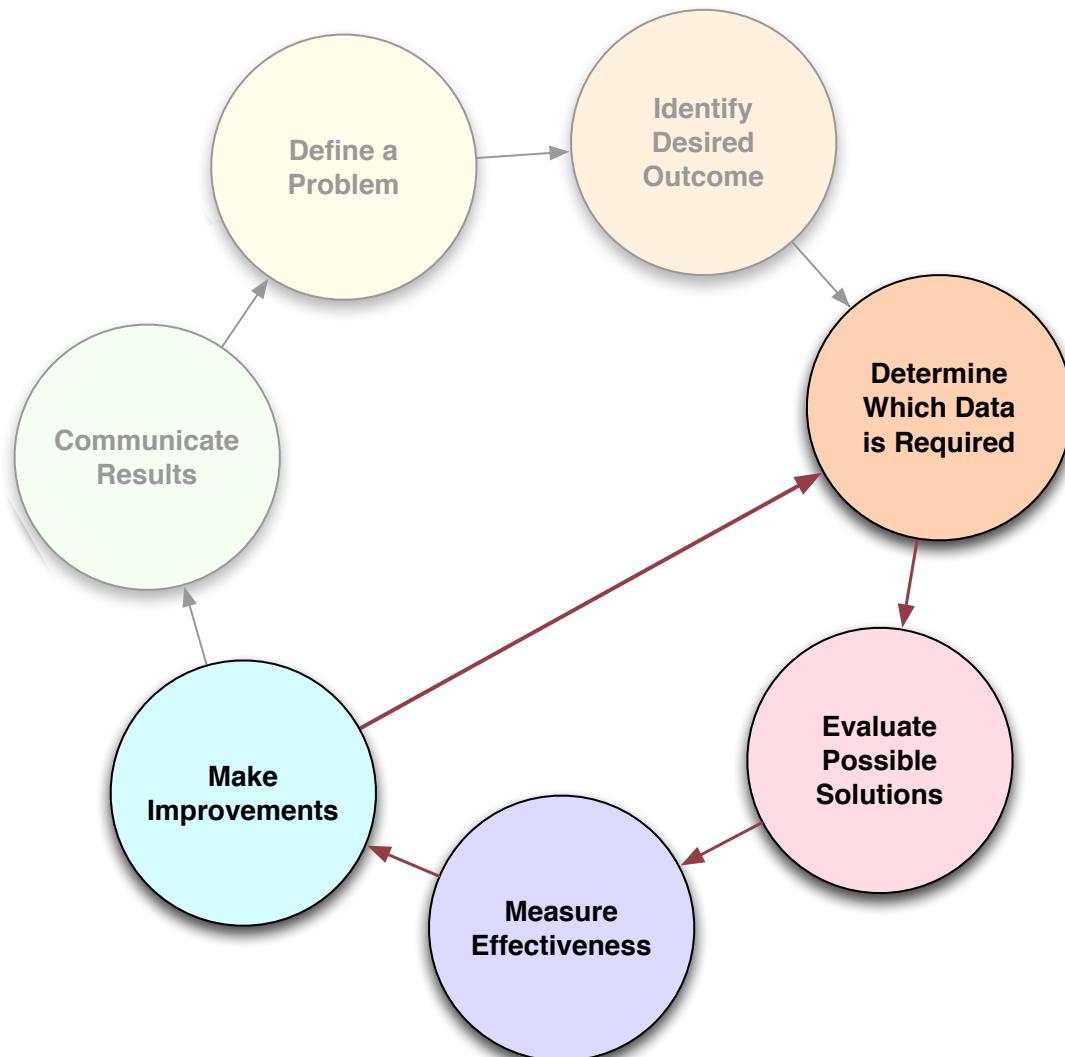
---

- A typical data science project should follow these steps
  - Define a problem
  - Identify the desired outcome
  - Determine which data is needed
  - Evaluate possible solutions
  - Measure effectiveness
  - Make improvements
  - Communicate results
- We'll discuss each of these in a moment...

# The Entire Project Lifecycle is Iterative



# Parts of the Project Lifecycle are Also Iterative



# Scale Affects the Solution

---

- **Scale is central to the iterative approach**
  - Validate an approach with a small sample of data
  - Test implementation with a moderate amount
  - Use a large amount of data to refine a proven solution
- **Implementations and even algorithms themselves may hit limits**
- **A given algorithm may work fine on 25MB of data**
  - But may require changes in implementation to work with 25GB
  - And may be impossible to scale to handle 25TB
- **Each iteration may involve a change to the amount of data used**
  - And to the code you use to analyze it

# Defining a Problem

---

- **The process begins by first clearly stating a problem**
- **This is often directly related to revenue**
  - “People browse our site but don’t buy anything”
  - “Too many customers abandon their shopping carts”
  - “Subscribers aren’t renewing their service”
  - “Sponsors are donating less than ever before”
- **In other cases, the problem is related to costs**
  - “Our employees spend too much time searching for documents”
  - “An increase in support calls cost us \$400,000 last year”

## Defining a Problem (cont'd)

---

- **Sometimes the approach is more exploratory**
  - “What can I learn about our users from this clickstream data?”
  - “Why are customers abandoning purchases before checkout?”
  - “How many more finish checkout when offered free shipping?”
  - “Would offering free shipping on all orders increase profits?”

# Identifying the Desired Outcome

---

- **Given the problem, what's the preferred resolution?**
- **Again, these are often tied to revenue or costs**
  - “Increase subscription renewals by 5% within two months”
  - “Decrease shopping cart abandonment by 10% in Q3”
  - “Reduce support call volume by 25% within one year”
- **Be careful what you wish for**
  - **Problem:** “An increase in support calls cost \$400,000 last year”
  - **Goal:** “Reduce support call volume by 25% within one year”
  - A reduction in support calls may not mean fewer problems
    - Could indicate customer frustration due to poor support

# Determining Which Data is Needed

---

- **What data must you capture to solve the problem you've defined?**
  - And to determine if your solution meets the goals identified
- **Further refinements may require additional data**
- **Consider the source, format, and quality of this data**
  - Does your organization have everything you need?
  - If not, is it available from external sources?

# Evaluate Possible Solutions

---

- Consider all solutions that could match desired outcome
- This typically involves a hypothesis about the root cause
  - What prompted the recent increase in support calls?
  - Why are customers abandoning their carts?
  - What causes customers to not renew subscriptions?
- Given several possible solutions, which should you invest in?
  - Most can be discounted quickly
  - Small-scale tests can help you choose
- The simplest solution is usually the best one to pursue first

# Measuring Effectiveness

---

- **Measuring effectiveness requires two things**
  - Metrics: properties to measure
  - Method: a process for comparing these metrics
- **Key point: have you achieved the desired outcome?**
  - Focus on measuring what really matters

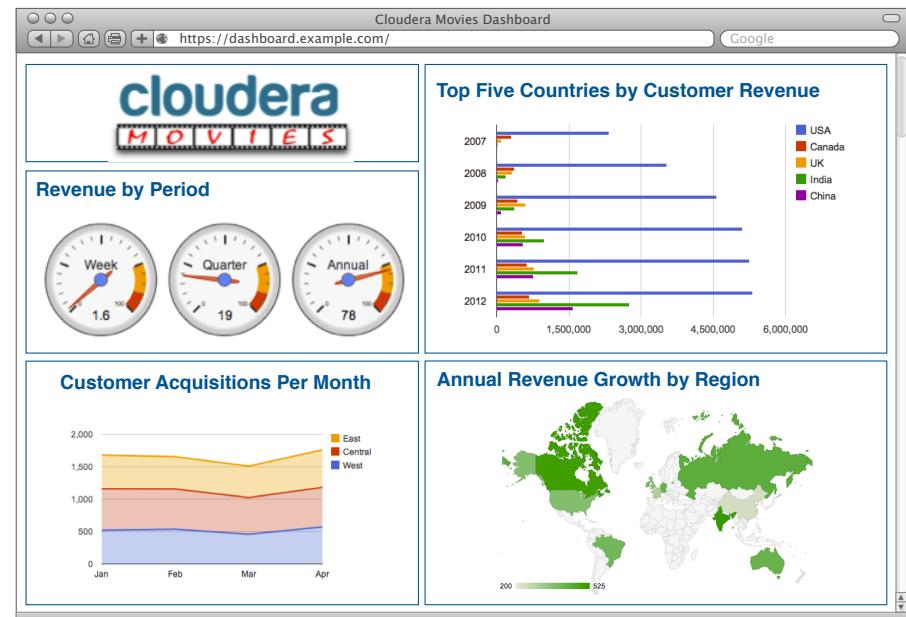
# Making Improvements

---

- **Measurement will illustrate how much improvement is required**
- **Consider what you might change**
  - Was your hypothesis about the root cause correct?
  - Could adding an additional data set give more insight?
  - Should you try one of the solutions you originally discarded?
- **Once you've implemented your improvements, test it again**
  - Comparing measurements can help to refine your solution

# Communicating Results

- **Communication is an essential part of data science**
- **A data scientist must tell the story found within the data**
  - Like any good story, it must be compelling
  - Be concise and focus on what's important for the audience
- **Dashboards are a common tool for communicating results**
  - Statistics
  - Summaries
  - Visualizations



# Chapter Topics

## Project Lifecycle

- Steps in the project lifecycle
- **Hands-On Exercises: scenario explanation**
- Review questions
- Essential points
- Conclusion

# Hands-On Exercises: Scenario Explanation

---

- **We'll practice what we've learned with hands-on exercises**
  - We will work through a complete data science problem during the course
- **Cludera Movies is a successful online movie streaming service**
  - Our 14 million customers pay a monthly subscription fee
- **Unfortunately, our success has started to wane**
  - Revenues dropped last quarter by 11%
  - Projections show revenues decreasing this quarter by 17%

## Hands-On Exercises: Scenario Explanation (cont'd)

---

- **Why are revenues down?**
  - Existing customers aren't renewing their subscriptions
  - Prospective customers aren't joining our service
- **Customers were surveyed when they called to cancel**
  - Reason cited by 79% of customers
    - "You have lots of movies, just none that interest *me*."
- **Cloudera Movies has hired you to help solve this problem**

# Hands-On Exercises: Scenario Explanation (cont'd)

---

- **Problem Definition**

- “Many customers are choosing not to renew their subscriptions”

- **Desired Outcome**

- “Decrease cancellation rate by 35% during next quarter”

- **Possible solutions**

- Decrease subscription cost (discard: price is not the problem)
  - Social media integration (discard: may violate privacy laws)
  - Improve movie recommendations (we'll pursue this one)

# Hands-On Exercises: Overview

---

- **There are many prerequisites to building a recommender system**
  - Acquiring the input data from various sources
  - Identifying and correcting errors in the input data
  - Transforming the data into the desired format for analysis
- **The work isn't finished even after you've built the recommender**
  - You need to test it
  - You'll likely find ways to improve it
    - These may require additional data sources
    - The project lifecycle begins again
- **Here's an overview of hands-on exercises you'll complete in this course...**

# Hands-On Exercises: Overview (cont'd)

---

- **Lab #0: Tool Basics**

- You'll become acquainted with the HDFS command line utility and Python, two tools we'll use extensively in later labs. You'll explore the use of Python's regular expressions for parsing a Web server log file, a valuable source of information for data scientists.

- **Lab #1: Data Import**

- You'll acquire data from different sources that will be used to build our recommender system

- **Lab #2: Evaluating Input Data**

- You'll identify errors in the input data and correct them using Hadoop MapReduce jobs written in Python

- **Lab #3: Data Transformation**

- You'll use Apache Hive to filter and join input data from earlier labs into a single object that represents all information about each customer

# Hands-On Exercises: Overview (cont'd)

---

- **Lab #4: Data Analysis**
  - You'll learn the basics of the R statistical language, then use it to analyze social media data that will later be used to test an improvement to our recommender
- **Lab #5: Basic Recommender**
  - You'll build a simple user-based recommender system in Python
- **Lab #6: Building a Recommender with Mahout**
  - You'll use Apache Mahout to build a scalable item-based recommender system using data you've acquired in earlier labs
- **Lab #7: Analyzing Your Recommender**
  - You'll evaluate data gathered from two versions of the recommender system in order to determine which is more effective

# Chapter Topics

## Project Lifecycle

- Steps in the project lifecycle
- Hands-On Exercises: scenario explanation
- **Review questions**
- Essential points
- Conclusion

## Review Questions

---

- Imagine that you're a data scientist working for an online brokerage firm. What metrics do you think would be important?
- Let's say that a major electronics retailer has hired you to decrease the bounce rate on their Web site. What kinds of data would be most helpful to analyze?

# Chapter Topics

## Project Lifecycle

- Steps in the project lifecycle
- Hands-On Exercises: scenario explanation
- Review questions
- **Essential points**
- Conclusion

## Essential Points

---

- The lifecycle of a data science project is iterative
- It's important to clearly state a problem
- The problem helps to establish the desired outcome
- Success is determined by measuring results

# Chapter Topics

## Project Lifecycle

- Steps in the project lifecycle
- Hands-On Exercises: scenario explanation
- Review questions
- Essential points
- **Conclusion**

# Project Lifecycle

---

**In this chapter you have learned**

- **How a data scientist approaches a problem**
- **Which steps comprise lifecycle of a data science problem**
- **About the data science problem we'll address through hands-on exercises**

# Bibliography

---

**The following offer more information on topics discussed in this chapter**

- **A Taxonomy of Data Science**

- <http://tiny.cloudera.com/dscc04a>

- **Setting Expectations in Data Science Projects**

- <http://tiny.cloudera.com/dscc04b>

# Data Acquisition

## Chapter 5



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- **Data Acquisition**
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Data Acquisition

---

**In this chapter you will learn**

- **What types of data are used in analysis**
- **Where you can find these data sets**
- **What are some common methods of accessing this data**

# Chapter Topics

## Data Acquisition

- **Where to source data**
- Acquisition techniques
- Review questions
- Essential points
- Hands-On Exercise: Acquiring Data
- Conclusion

# Data Quality and Provenance

---

- These are important issues to consider prior to acquisition
- Establish the trustworthiness of the original source
  - Is the organization collecting the data reputable?
  - Can I trust that the data itself is accurate?
- If acquired through a third party, determine lineage
  - Did they get the data directly from the *original* source?
  - Was the data modified from the original source?
- Repeatable results matter
  - Use a source control system for your code
  - Keep track of all changes to your data

# Internal Data Sources

---

- **Most valuable information comes from your own organization**
- **There are many sources of data available internally**
  - Application databases (OLTP)
  - Data warehouses (OLAP)
  - Log files (Web, e-mail, and other applications)
  - Documents (file servers, intranet, Web site)
  - Sensors and network events

# Freely Available Data Sources

---

- **External data is often used to augment a solution**
  - Geolocation for IP addresses in Web server logs
  - Demographic information about those locations
- **There are many sources of data available at no cost**
  - Some are public domain and some are copyrighted
  - Be sure to check the license to verify that your use is allowed

## Freely Available Data Sources (cont'd)

U.S. Census Bureau	<a href="http://factfinder2.census.gov/">http://factfinder2.census.gov/</a>
U.S. Executive Branch	<a href="http://www.data.gov/">http://www.data.gov/</a>
U.K. Government	<a href="http://data.gov.uk/">http://data.gov.uk/</a>
E.U. Government	<a href="http://publicdata.eu/">http://publicdata.eu/</a>
The World Bank	<a href="http://data.worldbank.org/">http://data.worldbank.org/</a>
Freebase	<a href="http://www.freebase.com/">http://www.freebase.com/</a>
Wikidata	<a href="http://meta.wikimedia.org/wiki/Wikidata">http://meta.wikimedia.org/wiki/Wikidata</a>
Amazon Web Services	<a href="http://aws.amazon.com/datasets">http://aws.amazon.com/datasets</a>
InfoChimps *	<a href="http://www.infochimps.com/marketplace">http://www.infochimps.com/marketplace</a>

\* Most data sets are available at no cost, but some have a fee

# Commercial Data Sources

---

- **Many companies also offer data**
  - Usually for a fee, but sometimes available at no cost
  - Always be sure to check the license terms

Gnip	Social Media	<a href="http://gnip.com/">http://gnip.com/</a>
AC Nielsen	Media Usage	<a href="http://www.nielsen.com/">http://www.nielsen.com/</a>
Rapleaf	Demographic	<a href="http://www.rapleaf.com/">http://www.rapleaf.com/</a>
ESRI	Geographic (GIS)	<a href="http://www.esri.com/">http://www.esri.com/</a>
eBay	Auction	<a href="https://developer.ebay.com/">https://developer.ebay.com/</a>
D&B	Business Entity	<a href="http://www.dnb.com/">http://www.dnb.com/</a>
Trulia	Real Estate	<a href="http://www.trulia.com/">http://www.trulia.com/</a>
Standard & Poor's	Financial	<a href="http://standardandpoors.com/">http://standardandpoors.com/</a>

# Chapter Topics

## Data Acquisition

- Where to source data
- **Acquisition techniques**
- Review questions
- Essential points
- Hands-On Exercise: Acquiring Data
- Conclusion

# Database Integration

- Data internal to an organization is often kept in a database
- To access small samples, just export a subset to a local file
  - Can do this programmatically or manually via query tool
  - Can also do this on command line, as shown below

```
$ cat 10k_customers.sql
select id, firstname, lastname, email, zipcode
into outfile '/user/jsmith/cust10k.csv'
fields terminated by ','
optionally enclosed by """
escaped by '\\'
lines terminated by '\n'
from customers limit 10000"

$ mysql -u jsmith -p mysecret < 10k_customers.sql
```

- Invocation details will vary depending on database used

## Database Integration (cont'd)

---

- **This approach isn't appropriate for large data sets**
  - HDFS (Hadoop) is a better choice once you reach terabyte range
  - HDFS is scalable, resilient, and offers high performance I/O
- **Sqoop exchanges data between an RDBMS and HDFS**
  - Import all tables, a single table, or a partial table into HDFS
  - Data can be imported in delimited text or Avro file format
  - Sqoop can also export data from HDFS to a database
- **Sqoop is compatible with almost any database**
  - Some also support high-performance custom connectors
- **Sqoop supports incremental imports**
  - Can bring in all existing data into HDFS during initial import
  - Then import just new records in subsequent imports

## Other Internal Sources

---

- Systems that produce data in the form of files are easily handled
  - For a few small files, just copy them to a local filesystem
  - Larger file sets should be copied to HDFS instead

```
$ hadoop fs -put myfile.txt /bigdata/project/myfile.txt
```

- This can be done manually or as part of a script
- HDFS also supports a REST API through WebHDFS
- Alternative: Use Flume to add data to HDFS as it's generated
  - Can “tail” log files to capture lines as soon as they're written
  - Other sources: program execution, network port, and syslog
  - Write custom sources to integrate with legacy systems

# Data Archive Downloads

---

- External data sources are sometimes in the form of archives
  - Delimited and fixed-width textfiles are most common type
  - Usually compressed to save storage space and bandwidth
- These are usually hosted on Web or FTP sites
  - Downloading is easy with your browser for small number of files
- How do you automate download of many files?
  - Use the curl or wget command line utilities

```
$ curl -i list_of_urls.txt  
  
$ curl -O http://www.example.com/xyz[001-999].zip  
  
$ curl -u jsmith:mysecret ftp://ftp.example.com/archive/bigfile.gz  
  
$ wget --mirror http://www.example.com/data/ -o /home/jsmith
```

# Data APIs

---

- **Many organizations offer data as services rather than downloads**
  - Some APIs are read-only, while others support data modification
  - Authentication is often required (register for an ID to use in calls)
- **Data APIs have several advantages over archive downloads**
  - The service maintains the data and can keep it updated
  - Usually cross-platform and cross-language (REST or SOAP)
  - Price may be based on only what you use
- **Access to data through APIs also has disadvantages**
  - Price or terms of service may change
  - Your application's availability depends on service availability
- **Data returned by an API is typically in XML or JSON format**

# Data API Example

- Here's an example request to the Twitter API using curl

```
$ curl https://api.twitter.com/1/users/show.json?screen_name=cloudera
```

- Here's the JSON response (formatted and excerpted)

```
{  
  "id":16134540,  
  "name":"Cloudera",  
  "screen_name":"cloudera",  
  "location":"Palo Alto, CA",  
  "url":"http:\/\/www.cloudera.com\/",  
  "description":"Cloudera is the leading provider of Apache  
              Hadoop-based software and services.",  
  "followers_count":11359,  
  "created_at":"Thu Sep 04 20:10:22 +0000 2008",  
  "time_zone":"Pacific Time (US & Canada)",  
}
```

# Screen Scraping

---

- Sometimes the data is only available within a Web site
  - You don't have access to the database powering the site
  - You only have access to the rendered pages themselves
- You can acquire the data by “screen scraping”
  - Programmatic access and parsing of page content
  - Fragile: your script may break when page changes
  - Should be viewed as a last resort

```
$ cat scraper.py
import urllib
from BeautifulSoup import BeautifulSoup

txt = urllib.urlopen("http://www.example.com/")
soup = BeautifulSoup(txt)

headings = soup.findAll("h2")
for heading in headings:
    print heading.string
```

# Chapter Topics

## Data Acquisition

- Where to source data
- Acquisition techniques
- **Review questions**
- Essential points
- Hands-On Exercise: Acquiring Data
- Conclusion

## Review Questions

---

- **How do you currently track changes to your data?**
- **Where would you look for information on languages spoken by residents of a given ZIP code?**
- **Imagine that you work for an e-Commerce company**
  - How would you fetch the first 1000 lines of log data from your Web servers each day?
  - How would this approach change as traffic increased and you wanted to analyze 5 TB of data each day?

# Chapter Topics

## Data Acquisition

- Where to source data
- Acquisition techniques
- Review questions
- **Essential points**
- Hands-On Exercise: Acquiring Data
- Conclusion

## Essential Points

---

- **The most valuable information is found within your organization**
- **There's a variety of data available from external sources that can help augment your solution**
- **External data is usually accessed as an archive or via an API**
  - Screen scraping is another option, but best avoided

# Chapter Topics

## Data Acquisition

- Where to source data
- Acquisition techniques
- Review questions
- Essential points
- **Hands-On Exercise: Acquiring Data**
- Conclusion

## Hands-on Exercise: Acquiring Data

---

- In this Hands-On Exercise, you will gain practice acquiring data using several of the methods we've discussed
  - The result will be examined, processed, and analyzed in upcoming labs
- Please refer to the Hands-On Exercise Manual for instructions on exercise #1

# Chapter Topics

## Data Acquisition

- Where to source data
- Acquisition techniques
- Review questions
- Essential points
- Hands-On Exercise: Acquiring Data
- **Conclusion**

# Data Acquisition

---

**In this chapter you have learned**

- **What types of data are used in analysis**
- **Where you can find these data sets**
- **What are some common methods of accessing this data**

# Bibliography

---

**The following offer more information on topics discussed in this chapter**

- **Data Provenance: Some Basic Issues**
  - <http://tiny.cloudera.com/dscc05a>
- **Data Integration: A Theoretical Perspective**
  - <http://tiny.cloudera.com/dscc05b>
- **Creating a Bioinformatics Nation**
  - <http://tiny.cloudera.com/dscc05c>
- **Programmable Web's Data API Index**
  - <http://tiny.cloudera.com/dscc05d>

# Evaluating Input Data

---

## Chapter 6



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- **Evaluating Input Data**
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Evaluating Input Data

---

## In this chapter you will learn

- Which file types are commonly used for input and output
- What are the advantages and disadvantages of these file types
- Several ways to examine data on the command line and at scale
- How sampling and filtering data can improve your processing
- What data quality problems are typical and how you can fix them

# Chapter Topics

## Evaluating Input Data

- **Data formats**
- Data quantity
- Data quality
- Review questions
- Essential points
- Hands-On Exercise: Evaluating Input Data
- Conclusion

# Data Formats

---

- **Data comes in many formats**
  - Format: the structure and encoding used to represent information
- **Format is primarily important at three points in the process**
  1. Format of the data provided to you or collected by you
  2. Format used as input to the analysis
  3. Format produced as output from the analysis
- **Some formats are better suited to certain uses than the others**
- **Data is sometimes converted to other formats during processing**
- **Some formats map to a relational model more than others**

# Log Files

- **Log files are generated by applications and devices**
  - Examples: Web servers, mail servers, Hadoop, cell phones
- **Data scientists view logs as a valuable source of information**
  - Contain data that's too expensive to store in a transactional DB
  - Data is available immediately – no need to wait for ETL process
  - Log analysis does not require putting load on production system

```
$ head -n1 httpd.log
192.168.5.137 - - [17/Aug/2012:21:18:36 -0600]
"GET /products/widget.jsp?sku=16879 HTTP/1.1" 200 8472
"http://www.example.com" "Mozilla/5.0 (Windows NT 5.1; rv:2.0) Gecko/
20100101 Firefox/4.0" "uid=jsmith;usertype=Customer;region=midwest"

$ cut -f1 -d' ' httpd.log | sort | uniq -c | sort -rn | head -n2
283 192.168.5.137
79 10.9.8.47
```

# Fixed-Width and Delimited Text Files

---

- **Data is sometimes provided as fields in text files**
  - Common for data exported from databases or spreadsheets
  - Typically one record per line
- **Two main variants**
  - Fixed-width: field starts at position M and occupies N characters
  - Delimited: fields separated by characters such as comma or tab
- **CSV files can be deceptively difficult to parse**
  - There is a specification, but few follow it exactly
  - Variations on quoting, embedded commas, missing fields, etc.

```
$ cut -c10-14 fixedwidth.txt  
$ cut -f3,5 mydata.tsv  
$ cut -d, -f2 mydata.csv | sed -e 's//"/g'
```

# XML and HTML

---

- **Data is commonly made available in XML or HTML format**
  - However, neither is an ideal format for analysis at scale
- **XML is a self-describing hierarchical text format**
  - XML is well-formed and can be validated for compliance
  - Verbose format consumes much storage and memory
- **HTML is a closely related type used for Web pages**
  - Likelier to deviate from spec and have less structure than XML
  - Content and formatting intertwined, especially in older documents

```
$ head -n4 customers.xml
<customer>
  <id>1234</id>
  <name type="display">Smith, Jane</name>
</customer>

$ perl -ne 'print m|<id>\s*(\d+)\s*</id>(\n)| ;' < customers.xml
```

# JSON

---

- **JSON is an alternative to XML**
  - It offers many of the benefits, but with fewer drawbacks
  - Format is also hierarchical and self-describing
  - Much less verbose than XML
- **Despite its JavaScript origins, it's supported by many languages**

```
{  
  "id":573698,  
  "name": "John Smith",  
  "address": "123 Hadoop Drive",  
  "zipcode": "90210",  
  "email": "jsmith@example.com",  
  "phone_numbers": [  
    { "type": "mobile", "number": "(213) 555-1953" },  
    { "type": "work", "number": "(310) 555-2752" },  
    { "type": "home", "number": "(310) 555-7419" }  
  ],  
}
```

# Binary Input Formats

---

- **Not all data collected is text-based**
  - Images
  - Spreadsheets
  - Word processor and PDF documents
  - Audio and video
- **These formats are not necessarily ideal for analysis**
  - Not natively supported by Hadoop or ecosystem tools
  - Analysis typically involves format-specific custom code
- **Often better to convert to a text-based format before processing**
  - For example, convert Excel to CSV, or PDF to text
  - This may not be possible for some formats (such as images)
- **In some cases, only the metadata is actually needed for analysis**

# SequenceFiles

---

- **SequenceFiles are a Hadoop-specific format**
  - Flat binary file consisting of key-value pairs
  - You are unlikely to receive data in this format, but may produce it
  - Hadoop, Hive and Pig all support this format well
- **SequenceFiles offer better performance than text-based formats**
  - No need to convert native types to and from String values
  - Compression (optional) may offer still better performance
- **Unfortunately, SequenceFiles are closely tied to Java**
  - Cannot access data easily from other languages
  - Even changing your Java classes can break compatibility
- **SequenceFiles are beneficial for intermediate data**
  - Limitations described above affect use for long-term storage

# Avro

---

- **Avro is an Apache project for data serialization**
  - It addresses many of the shortcomings of SequenceFiles
  - Binary data format is concise and offers good performance
  - Language support (C, C++, C#, Java, perl, Python, Ruby, PHP)
- **Avro works with Hadoop MapReduce**
  - In Java via AvroMapper and AvroReducer classes
  - In other languages via AvroAsTextInputFormat and AvroKeyValueOutputFormat in a Streaming job
- **Avro also works with both Hive and Pig**
- **Data can be easily collected in Avro format using Flume**

# Chapter Topics

## Evaluating Input Data

- Data formats
- **Data quantity**
- Data quality
- Review questions
- Essential points
- Hands-On Exercise: Evaluating Input Data
- Conclusion

# Data Quantity Considerations

---

- **Data quantity is a defining characteristic of data science**
- **However, preliminary analysis is often best done with less data**
  - Smaller data sets mean faster execution times
  - Faster execution times allow for more iterations
  - More iterations provides more opportunities to refine solution

# Filtering

---

- **The goal of filtering is to limit the amount of data**
  - Include only certain records
  - Exclude only certain records
  - Isolate only those fields relevant for analysis
  - Can also combine any of these approaches
- **This can have profound impact on performance**
  - Eliminating data before it is processed will usually improve performance more than any optimization of your analysis code

```
$ gunzip -c logfile.gz | grep jsmith > only_jsmith.log  
$ gunzip -c logfile.gz | grep -v 'GET /img/' > no_images.log  
$ egrep '^63[0-3][0-9]{2}' clients.txt | egrep -vi 'smith|johnson' | less  
$ cut -f1,3,9 mydata.tsv > mydata-3cols.txt
```

## Filtering (cont'd)

- Using grep is convenient, but it doesn't scale well
- How do you filter terabytes of data quickly?
  - Hadoop lets you divide and conquer across many nodes
  - Easy to do in Python, using Hadoop Streaming

```
$ cat mapper.py
#!/usr/bin/env python

import sys

for line in sys.stdin:
    line = line.strip()

    if "jsmith" in line.lower():
        print '%s' % line

$ hadoop jar \
  /usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-stream*.jar \
  -mapper mapper.py -file mapper.py \
  -input /user/training/input \
  -output /user/training/output
```

# Sampling

- **Sampling allows you to capture a subset of your data**
  - It's easier to examine and explore this sample than huge data set
  - Differs from filtering in that eventually you'll want all the data
- **There are different sampling strategies**
  - Extracting a set of similar records
  - Choosing values at random
  - Intentionally selecting extreme values

```
$ head -n100 purchases.log  
  
$ gunzip -c httpd.log.gz | tail -n300  
  
$ dd bs=1M count=25 if=/home/training/foo.dat of=/tmp/25_megs_of_foo.dat  
  
$ split -C5000 httpd-big.log httpd_
```

## Sampling (cont'd)

- How can we get a random sample at scale?
  - Also easy to do with Hadoop Streaming and Python
  - The example below has a ~ 1% chance of selecting any record
  - Job is submitted as in the previous example

```
$ cat mapper.py
#!/usr/bin/env python

import sys
import random

include_pct = 0.01

for line in sys.stdin:
    line = line.strip()

    if random.random() < include_pct:
        print '%s' % line
```

# Measuring Input Data with Counters

---

- You might want to track types of records seen while processing
  - How many bad records did you encounter?
  - How many requests were for JPG versus PNG?
  - How many users accessed a certain URL?
- Hadoop supports counters and groups of counters
  - Can be incremented during processing
  - In Streaming, counters are updated by printing to STDERR
  - Result is shown in logs and Web UI once job is complete

## Measuring Input Data with Counters (cont'd)

- General format: **reporter:counter:<group>,<counter>**
- Python excerpt below shows how to count image types
  - Group name is FILE\_TYPE
  - Counter name is one of JPG, PNG, or OTHER

```
if ".jpg" in line.lower():
    sys.stderr.write("reporter:counter:FILE_TYPE,JPG\n")
elif ".png" in line.lower():
    sys.stderr.write("reporter:counter:FILE_TYPE,PNG\n")
else:
    sys.stderr.write("reporter:counter:FILE_TYPE,OTHER\n")
```

- There are limits on the number of counters and counter groups

# Chapter Topics

## Evaluating Input Data

- Data formats
- Data quantity
- **Data quality**
- Review questions
- Essential points
- Hands-On Exercise: Evaluating Input Data
- Conclusion

# Data Quality Overview

---

- **Quality problems are inevitable in a sufficiently large data set**
- **Three main types of problem**
  - Inconsistent: correct but with minor formatting variations
  - Invalid: incorrect but conforms to expected format
  - Corrupt: doesn't conform to expected format at all

# Common Problems

---

- **Inconsistencies in case of string values**
  - CA versus ca
  - Recommendation: always convert to one case
- **Inconsistencies in date formats**
  - 12/31/2012 versus Dec. 31, 2012
  - Recommendation: always convert to one format
  - A string like 20121231 occupies less space and sorts correctly
- **Inconsistencies in times**
  - Is 12:00:00 noon or midnight? What time zone?
  - Recommendation: use a 24-hour format
  - Recommendation: use a consistent time zone, such as GMT

## Common Problems (cont'd)

- Differences in how missing values are represented
  - Does it use NULL or N/A or zero or spaces or an empty string?
  - Recommendation: use as few representations as possible
    - May need one for strings and another for numeric fields
- Variations in free-form input
  - CA versus California (might also be misspelled in various ways)
    - Recommendation: limit free-form input if possible
    - Recommendation: scan to find all variations, then normalize

```
$ cut -f5 data.tsv | sort | uniq -c | sort -rn
9887 CA
 8 California
 3 CA.
 1 Cailfornia
 1 Caleefornya
```

# Identifying Bad Data

---

- **Small scale strategies**

- Examine columns with UNIX tools like `head`, `cut`, and `awk`
  - Write custom code that inspects records

- **Large scale strategies**

- Use counters in a Hadoop job to count bad records
  - Use logging in a Hadoop job to log unexpected data
    - Exercise caution with this approach!
    - Only log *bad* records, not *all* records

# Resolution Techniques

---

- **How do you fix the bad data once you've identified it?**
  - Fix the data upstream to avoid the issue altogether
  - Pre-process the data to fix the problem before analysis
  - Correct bad data on the fly during analysis
  - Ignore bad data as you find it during analysis
- **Which is best depends on a few factors**
  - Can you control how data is generated?
  - How valuable is the data?
  - Will you analyze this data more than once?

# Chapter Topics

## Evaluating Input Data

- Data formats
- Data quantity
- Data quality
- **Review questions**
- Essential points
- Hands-On Exercise: Evaluating Input Data
- Conclusion

## Review Questions

---

- Why are SequenceFiles a poor choice for archiving data?
- Imagine that you're a data scientist working for a consumer marketing company. The US Census Bureau has just released the data for the 2010 census and you plan to add it to your Hadoop cluster. At what point would you check for inconsistencies in the data and what would you do to correct them?

# Chapter Topics

## Evaluating Input Data

- Data formats
- Data quantity
- Data quality
- Review questions
- **Essential points**
- Hands-On Exercise: Evaluating Input Data
- Conclusion

## Essential Points

---

- Some input formats are best suited for input, some for intermediate data and others for final output
- Filtering irrelevant data can significantly improve performance
- Data sampling provides a subset that's easier to work with
- Issues with data quality are inevitable at scale – you must identify and correct them

# Chapter Topics

## Evaluating Input Data

- Data formats
- Data quantity
- Data quality
- Review questions
- Essential points
- **Hands-On Exercise: Evaluating Input Data**
- Conclusion

## Hands-on Exercise: Evaluating Data

---

- In this Hands-On Exercise, you will gain practice evaluating the quality of the input data acquired during the previous exercise, determining what problems it contains, and then correcting these problems
  - The result of this will be high-quality data that you'll use in upcoming exercises
- Please refer to the Hands-On Exercise Manual for instructions on exercise #2

# Chapter Topics

## Evaluating Input Data

- Data formats
- Data quantity
- Data quality
- Review questions
- Essential points
- Hands-On Exercise: Evaluating Input Data
- **Conclusion**

# Evaluating Input Data

---

In this chapter you have learned

- Which file types are commonly used for input and output
- What are the advantages and disadvantages of these file types
- Several ways to examine data on the command line and at scale
- How sampling and filtering data can improve your processing
- What data quality problems are typical and how you can fix them

# Bibliography

---

The following offer more information on topics discussed in this chapter

- **Data Interoperability with Apache Avro**

- <http://tiny.cloudera.com/dscc06a>

- **Executing Data Quality Projects**

- <http://tiny.cloudera.com/dscc06b>

- **The Data Wrangler Project**

- <http://tiny.cloudera.com/dscc06c>

- **Best Practices in Data Cleaning**

- <http://tiny.cloudera.com/dscc06d>

- **Exploratory Data Analysis by John Tukey**

- <http://tiny.cloudera.com/dscc06e>

# Data Transformation

---

## Chapter 7



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- **Data Transformation**
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Data Transformation

---

## In this chapter you will learn

- Why you might wish to convert file formats prior to analysis
- How you can join both small and large data sets
- What anonymization is and why it's important
- How re-identification can expose an organization to liability

# Chapter Topics

## Data Transformation

- **File format conversion**
- Joining data sets
- Anonymization
- Review questions
- Essential points
- Hands-On Exercise: Transforming Data
- Conclusion

# File Format Conversion

---

- **Sometimes data isn't provided in the same format you require**
  - The format might be suitable for data collection but not analysis
  - It might not be appropriate at expected scale
  - It might not be supported by the tool you need
  - Another format might offer better performance
  - Another format might be better for long-term storage
- **The solution is often to convert data to another format**

# File Format Conversion

---

- **Approaches to file format conversion for small data sets**
  - UNIX command-line (`tr`, `join`, `paste`, `sed`, `awk`, etc.)
  - Use conversion utilities like ImageMagick, tidy or poppler-utils
  - Use an application's export or “Save As” feature
  - Write script or small program to run on a single machine

# File Format Conversion (cont'd)

---

- **Approaches to large-scale conversion**

- Distributed conversion with custom code in Map-only Hadoop job
  - Use Hadoop to write records in new format
    - SequenceFileOutputFormat
    - AvroOutputFormat
    - Custom subclasses of FileOutputFormat

# Brief Introduction to Apache Hive

---

- **Another way of converting file formats involves using Apache Hive**
  - Let's first briefly cover what Hive is and what it can do
- **Hive is an alternative to writing low-level MapReduce code**
  - Users can analyze data stored in Hadoop data via HiveQL
    - HiveQL is a declarative language very similar to SQL
- **Hive does *not* turn your Hadoop cluster into a database**
  - Instead, the Hive interpreter turns HiveQL into MapReduce jobs
  - Hive tables are simply directories of data stored in HDFS
    - The `create table` statement instructs Hive how to parse it

## Brief Introduction to Apache Hive (cont'd)

- Hive is especially useful for joining data
  - We'll cover this later in the chapter, but here's an example in HiveQL

```
SELECT customer.id, customer.name, sum(order.cost)
      FROM customer INNER JOIN order
        ON (customer.id = order.customer_id)
     WHERE customer.zipcode = '63105'
GROUP BY customer.id, customer.name;
```

# Hive SerDes

---

- **Hive can read and write data in many file formats**
  - Via implementations of the Serializer/Deserializer (SerDe) API
- **There are many SerDes available for Hive, including**
  - Delimited text file
  - RegexSerde
  - JSON
  - Avro
- **It's also possible to implement your own custom SerDe**

# Regex Serde Example

- Load a log file into a three column table
  - Sample input file

```
30-Nov-2012 23:15:21 "Unusual event detected"  
01-Dec-2012 01:33:02 "System shutdown"  
01-Dec-2012 01:34:59 "System restarted"
```

- Create table example

```
CREATE TABLE LOGDATA (date_str STRING, time_str STRING, msg STRING)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'  
WITH SERDEPROPERTIES (  
    "input.regex" = "^(\d{2})-(\w{3})-(\d{4}) \s+(\d{2}):\d{2}:\d{2})  
    \s+(\w+) \s+\"(.+)\\" \s*"  
)
```

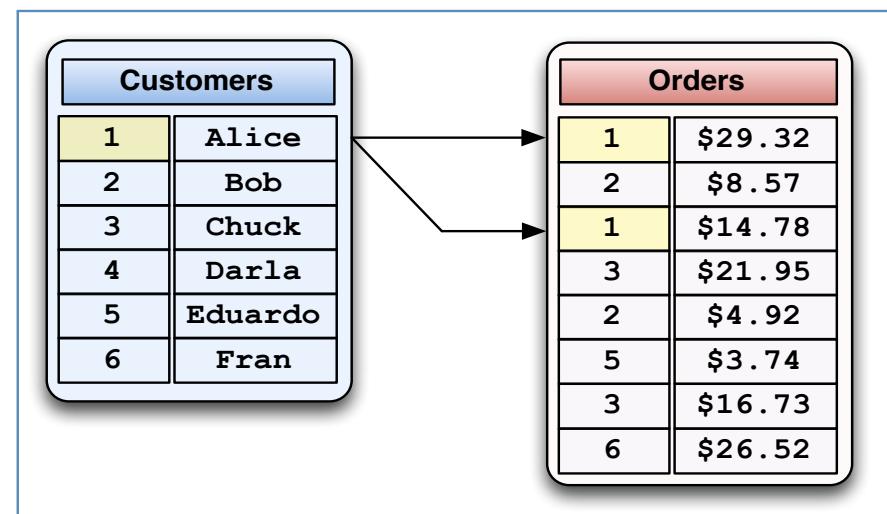
# Chapter Topics

## Data Transformation

- File format conversion
- **Joining data sets**
- Anonymization
- Review questions
- Essential points
- Hands-On Exercise: Transforming Data
- Conclusion

# Joining Data Sets

- **Joins are a common operation with relational data**
  - Two distinct data sets are joined based on a common key field
- **Data scientists use joins to connect disparate data sets**
  - This provides insight that no single data set could
- **Joining data is an expensive operation**
  - It's important to amortize the cost of doing joins
  - Do joins once, up front



## Joining Data Sets (cont'd)

- The UNIX **join** command can be used for small data sets
  - Will join on key field in first column by default
  - Options allow key field in a different column for each file

```
$ cat customers.txt
1 Alice
2 Bob
3 Chuck

$ cat orders.txt
1 $3.97
1 $19.34
2 $8.55
3 $8.22

$ join customers.txt orders.txt
1 Alice $3.97
1 Alice $19.34
2 Bob $8.55
3 Chuck $8.22
```

# Joining Larger Data Sets

---

- **Doing joins in a relational database may be an option**
  - If the data originated in the RDBMS...
  - Do the join in a SQL statement
  - Export the result to a file for analysis
- **It's also possible to join data with Hadoop using MapReduce**

# Joining Data Sets with Hive

---

- **Hive is an alternative to writing low-level MapReduce code**
- **Joining data sets with Hive is easy**
  - Usually preferable to writing MapReduce code to do joins
- **Benefits of using Hive for joins**
  - Far less code
  - Much quicker to write
  - Less chance for error
  - Requires far less skill, so it's accessible to more people
- **Disadvantages of using Hive**
  - Slightly less control

## Joining Data Sets with Hive (cont'd)

- The following is an example of a join in Hive
  - This is equivalent to several dozen lines of MapReduce code

```
SELECT customer.fname, customer.lname, customer.email,  
order.date, order.amount  
FROM customer  
JOIN order ON (customer.cid = order.cid)  
WHERE order.amount >= 50;
```

# Joining Data Sets with Hive (cont'd)

---

- **Recall two important points mentioned previously**
  - Joining disparate data sets yields insight
  - Joins are expensive – amortize the cost by doing them only once
- **Hive is very scalable**
  - Joins may produce dozens or hundreds of columns
- **Common to output a huge single file from several data sets**
  - Everything a hospital knows about a patient
    - Name, contact info, insurance info, medical history, etc.
  - Everything a company knows about a customer
    - Name, demographics, past orders, Web site sessions, etc.
  - Everything a manufacturer knows about a product
    - Configurations, part numbers, suppliers, defect history, etc.

# Chapter Topics

## Data Transformation

- File format conversion
- Joining data sets
- **Anonymization**
- Review questions
- Essential points
- Hands-On Exercise: Transforming Data
- Conclusion

# Anonymization

---

- **Anonymization is the process for removing PII from data**
  - IDs
  - Names
  - Addresses
  - Phone numbers
  - Potentially many other kinds of information
- **Why anonymize data?**
  - Laws may require it, particularly for finance and healthcare data
  - Industry standards
  - Company policies
  - Protects against attack

PII: Personally Identifiable Information

## Anonymization (cont'd)

---

- **Limiting access to non-anonymized data is essential**
  - Original data should be readable by as few people as possible
- **Typical anonymization process is to remove identifying columns**
  - First join all data sets
  - Next remove all ID fields needed only for the join
  - Finally, suppress fields which contain any PII

# Re-Identification

---

- **Several companies have been affected by re-identification**
  - For example: Researchers found Netflix prize data could re-identify people
    - Past ratings may expose political and sexual orientation
    - \$5 billion class action suit filed and later settled
- **Include only the minimum data required for the intended purpose**
  - A single field might not be PII, but a combination of them might
  - Research has shown that 87% of U.S. population can be uniquely identified from gender, ZIP code and date of birth
- **Using partial values can further anonymize data**
  - Use only first three digits of ZIP code
  - Retain the year of birth but exclude the month and day

# Chapter Topics

## Data Transformation

- File format conversion
- Joining data sets
- Anonymization
- **Review questions**
- Essential points
- Hands-On Exercise: Transforming Data
- Conclusion

## Review Questions

---

- **Which laws, regulations, or policies mandate anonymization?**
  - What are some other reasons you may want to anonymize data?
- **How would you convert a single Excel spreadsheet to CSV?**
  - How would you convert 100,000 Excel spreadsheets to CSV?
- **Why should you join data sets early in the project and why would you store the result of data sets you've joined?**

# Chapter Topics

## Data Transformation

- File format conversion
- Joining data sets
- Anonymization
- Review questions
- **Essential points**
- Hands-On Exercise: Transforming Data
- Conclusion

## Essential Points

---

- Anonymization removes personally identifiable information (PII) from your data and may be required by laws, regulations, or company policies
- Data isn't always provided in the format you need, so you may have to convert it
- You should join data sets once – as early as possible – and store the result to allow you to amortize the cost of this operation

# Chapter Topics

## Data Transformation

- File format conversion
- Joining data sets
- Anonymization
- Review questions
- Essential points
- **Hands-On Exercise: Transforming Data**
- Conclusion

## Hands-on Exercise: Transforming Data

---

- In this Hands-On Exercise, you will gain practice using Hive to join the disparate data sets you've previously acquired
  - This will produce a JSON object for each user to be further analyzed in upcoming exercises
- Please refer to the Hands-On Exercise Manual for instructions on exercise #3

# Chapter Topics

## Data Transformation

- File format conversion
- Joining data sets
- Anonymization
- Review questions
- Essential points
- Hands-On Exercise: Transforming Data
- **Conclusion**

# Data Transformation

---

## In this chapter you have learned

- Why you might wish to convert file formats prior to analysis
- How you can join both small and large data sets
- What anonymization is and why it's important
- How re-identification can expose an organization to liability

# Bibliography

---

The following offer more information on topics discussed in this chapter

- **Introduction to Data Anonymization**
  - <http://tiny.cloudera.com/dscc07a>
- **Fast Data Anonymization With Low Information Loss**
  - <http://tiny.cloudera.com/dscc07b>
- **Resisting Re-Identification in Anonymized Social Networks**
  - <http://tiny.cloudera.com/dscc07c>
- **The Regular Expressions Cheat Sheet**
  - <http://tiny.cloudera.com/dscc07d>
- **A Comparison of Join Algorithms in MapReduce**
  - <http://tiny.cloudera.com/dscc07e>

# Data Analysis and Statistical Methods

## Chapter 8



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- **Data Analysis and Statistical Methods**
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Data Analysis and Statistical Methods

---

## In this chapter you will learn

- How statistics and probability are related
- How scatterplots can help you identify numeric errors
- How to evaluate data distribution
- How extreme values can mislead you
- How regression analysis can help to predict missing values
- Which types of variables are important in regression analysis

# Chapter Topics

## Data Analysis and Statistical Methods

- Relationship between statistics and probability
- Descriptive statistics
- Inferential statistics
- Review questions
- Essential points
- Conclusion

# Comparison between Probability and Statistics

---

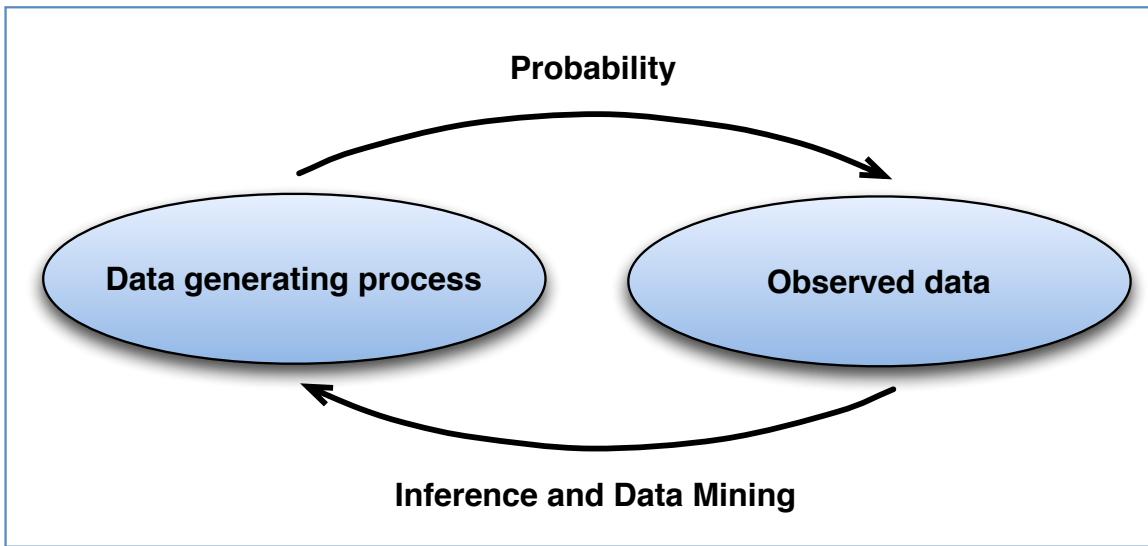
- **Probability deals with the prediction of future events**
  - This is inherently theoretical, as it's based on an *ideal* world
  - Example: “Which movies is this user likely to enjoy?”
  
- **Statistics deals with measurements from past events**
  - This is inherently more practical, as it's based on the *real* world
  - Example: “Which movies did this user rate highest?”

# The Cycle of Prediction and Measurement

---

- **Data scientists regularly employ both prediction and measurement**
  - Predict what's likely to happen given a particular set of circumstances
  - Then, conduct an experiment to measure the accuracy of prediction
- **This approach is cyclical**
  - Results from past experiments influence future predictions
- **Ultimately, statistics and probability are closely related**
  - They can be viewed as two sides of the same coin

# Relationship of Probability and Statistics



*All of Statistics: A Concise Course in Statistical Inference*  
Larry Wasserman, 2003

- **Probability: given a data generating process...**
  - What are the properties of the outcomes?
- **Statistics: given the outcomes...**
  - What can we say about the process that generated the data?

# Chapter Topics

## Data Analysis and Statistical Methods

- Relationship between statistics and probability
- **Descriptive statistics**
- Inferential statistics
- Review questions
- Essential points
- Conclusion

# Descriptive Statistics

---

- **Descriptive statistics answer questions about data distribution**
  - What is the range of values (distance from min to max)?
  - Where are values concentrated within this range?
- **Understanding data distribution is a preliminary step**
  - Helps to expose “dirty” data that you should correct or remove
  - Assists you in finding interesting patterns within the data
- **Distribution may affect how further analysis is performed**
- **Visualization is an essential tool for analyzing distributions**
  - Scatterplots and histograms are particularly helpful

# Exposing Data Errors with Scatterplots

---

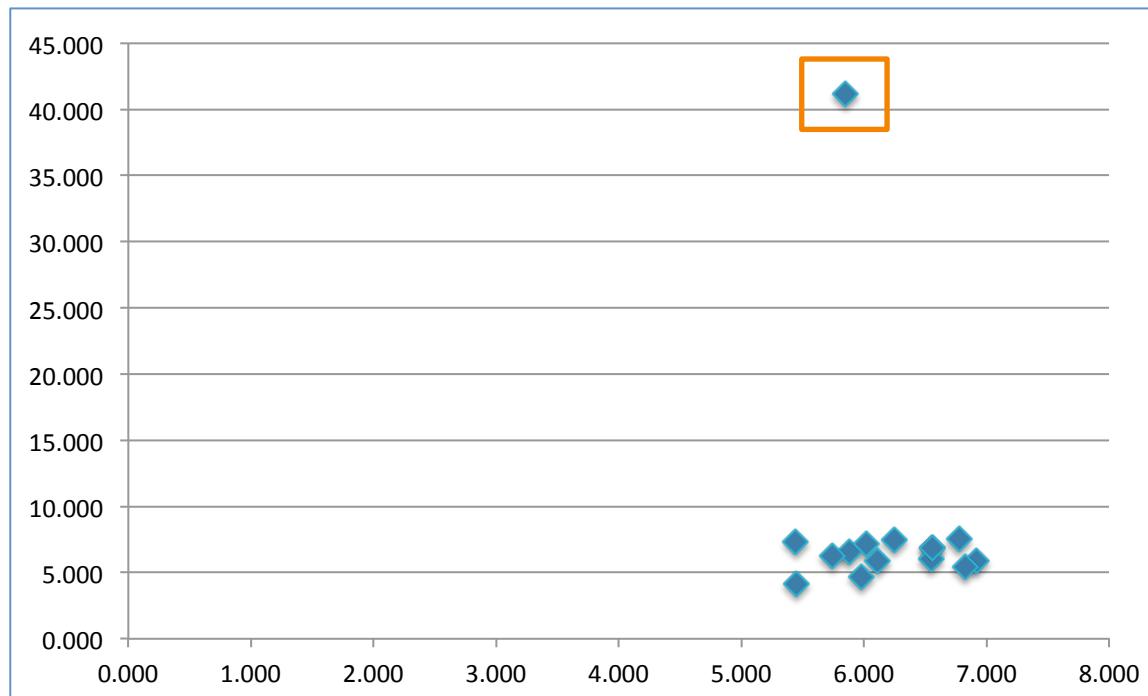
- **Data may contain “bugs” just as software does**
  - These can be difficult to find and may lead to invalid conclusions
  - Looking at a set of numbers seldom uncovers the problem

6.112	5.871
6.917	5.892
6.547	6.020
6.823	5.418
5.879	6.574
6.554	6.877
5.741	6.297
5.447	4.179
5.847	41.17
5.974	4.713
6.247	7.474
6.551	6.874
5.441	7.347
6.774	7.514
6.018	7.142

## Exposing Data Errors with Scatterplots (cont'd)

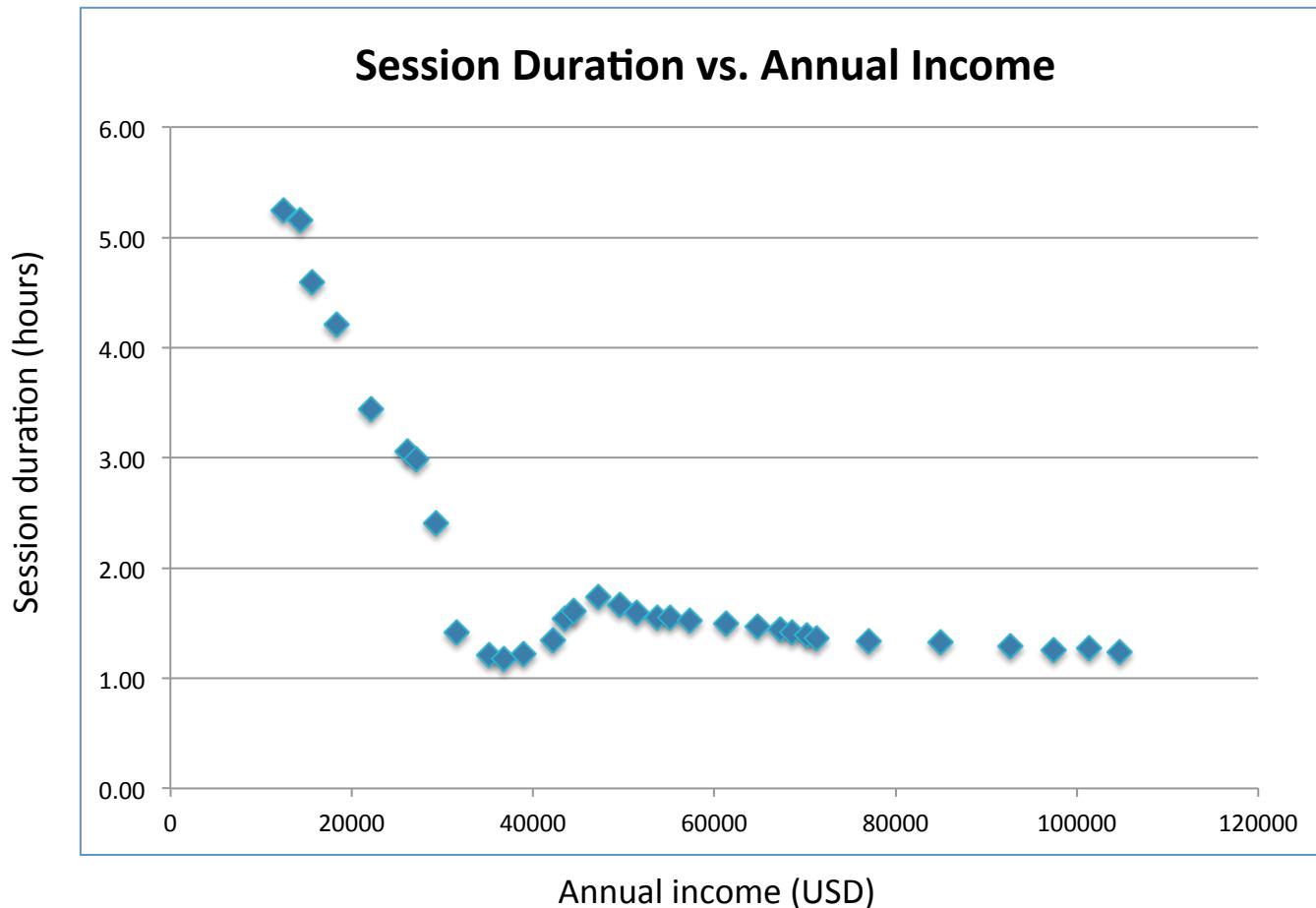
- Scatterplots help to expose potentially invalid data
  - The latent problem in this data set is obvious when visualized

6.112	5.871
6.917	5.892
6.547	6.020
6.823	5.418
5.879	6.574
6.554	6.877
5.741	6.297
5.447	4.179
5.847	41.17
5.974	4.713
6.247	7.474
6.551	6.874
5.441	7.347
6.774	7.514
6.018	7.142



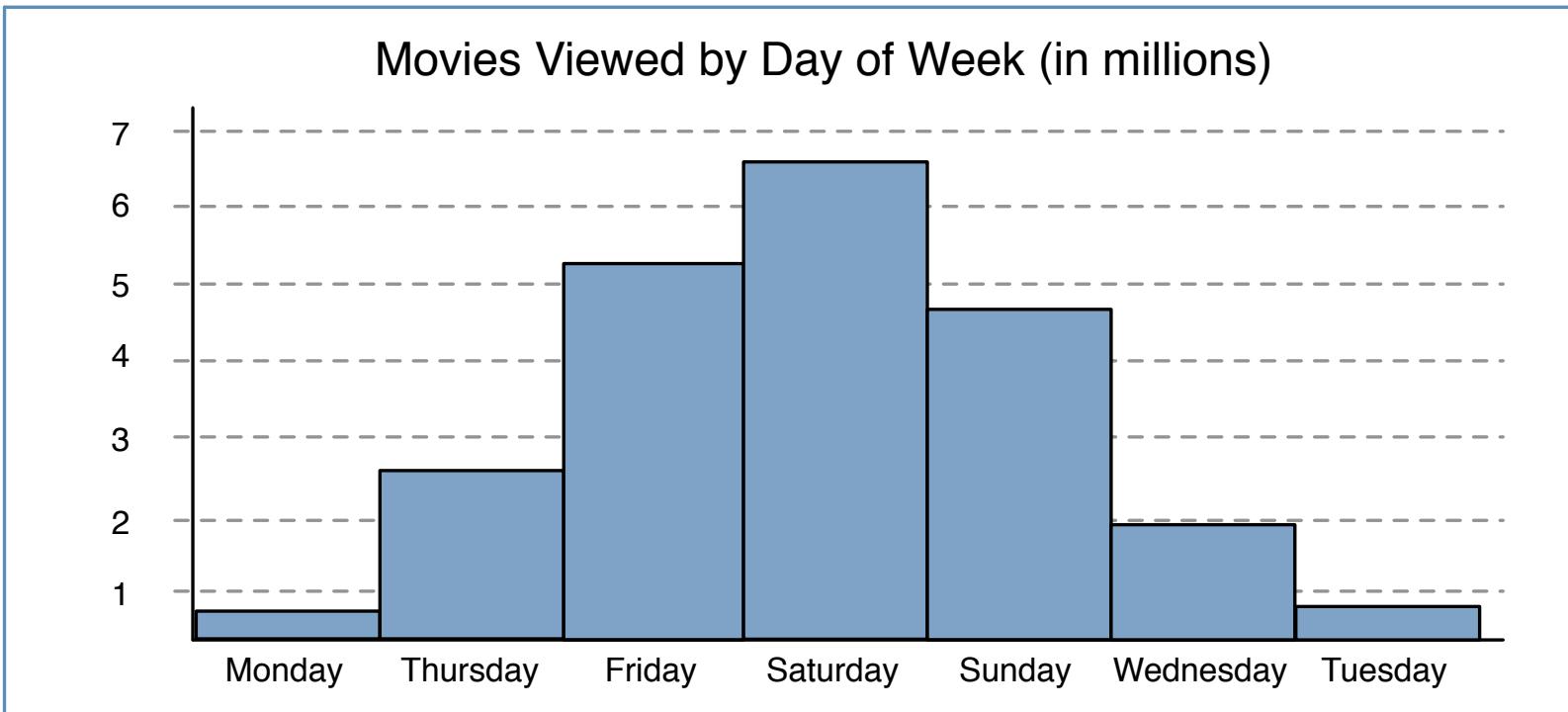
# Finding Interesting Patterns with Scatterplots

- Scatterplots also help point out relationships in data



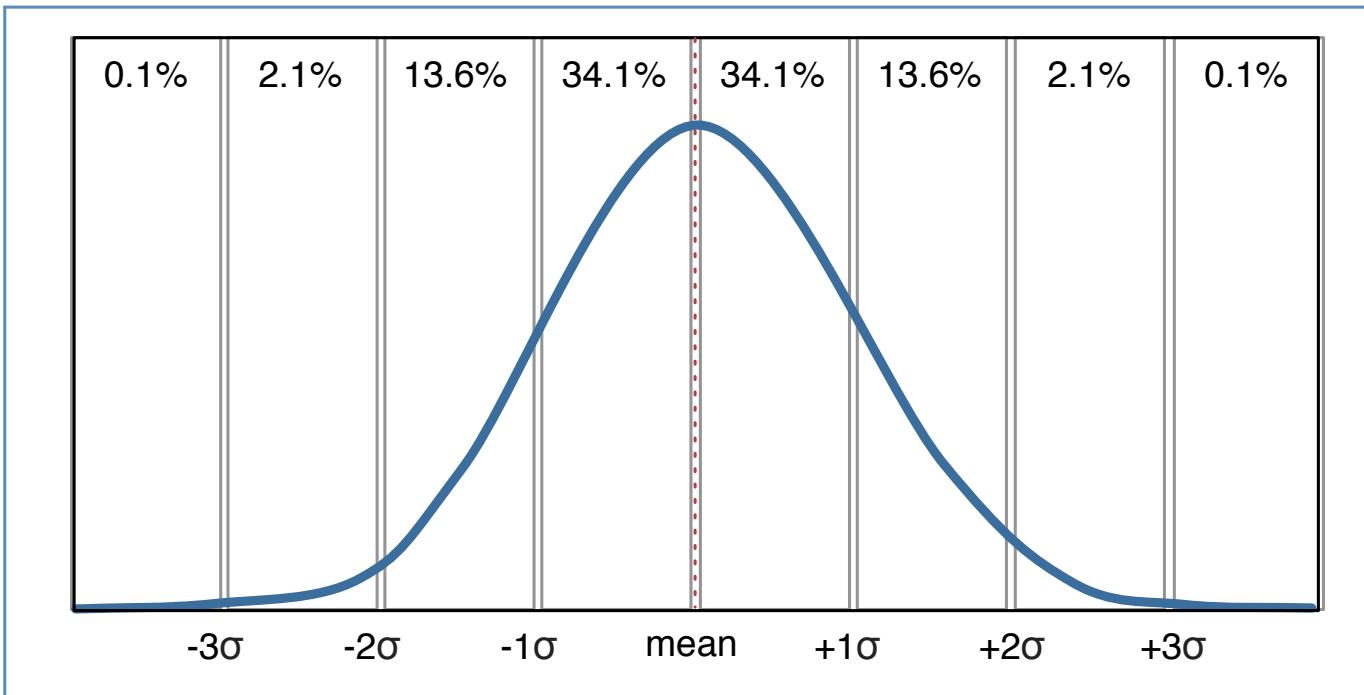
# Histograms

- A histogram illustrates the distribution of data
  - This helps you to compare relative frequency



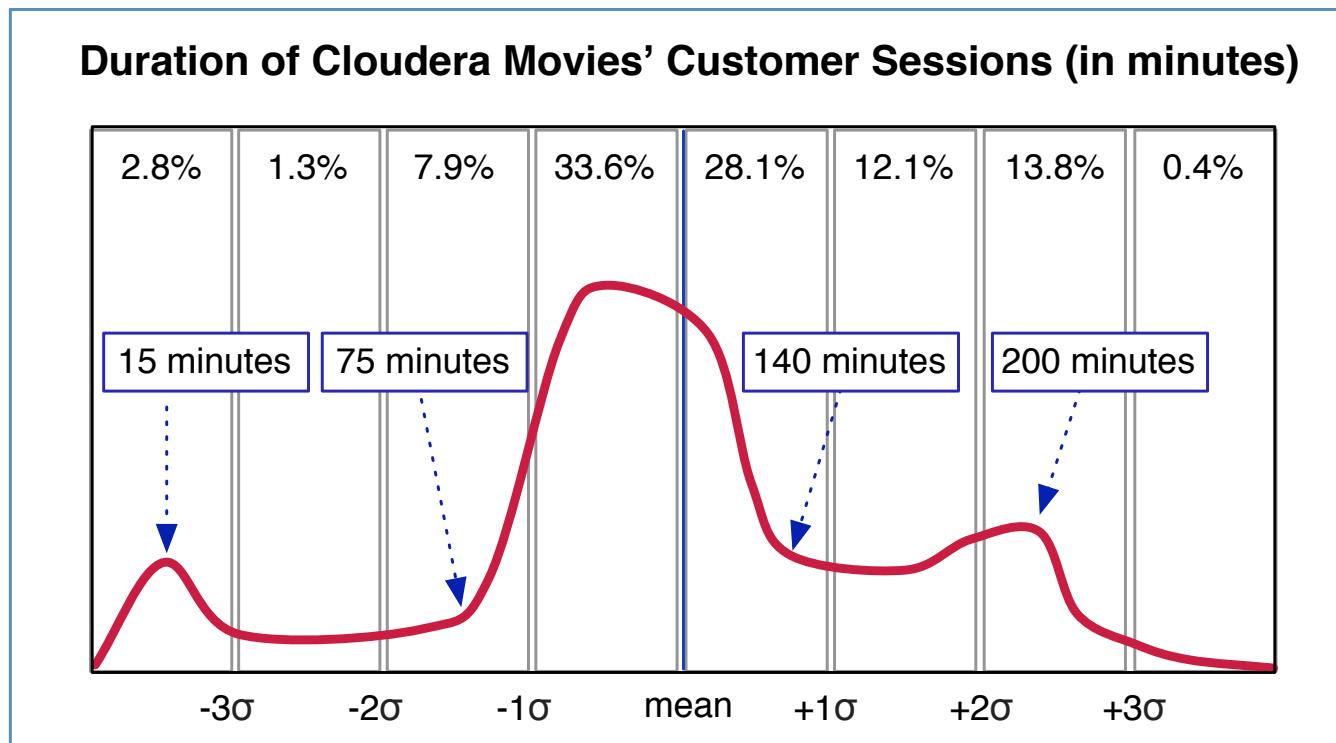
# Normal Distribution

- The normal distribution of data is often called the “bell curve”
  - Distribution is symmetrical about the mean (average)
  - More than two-thirds lies within one standard deviation
    - Standard deviation is a measure of dispersion from the mean



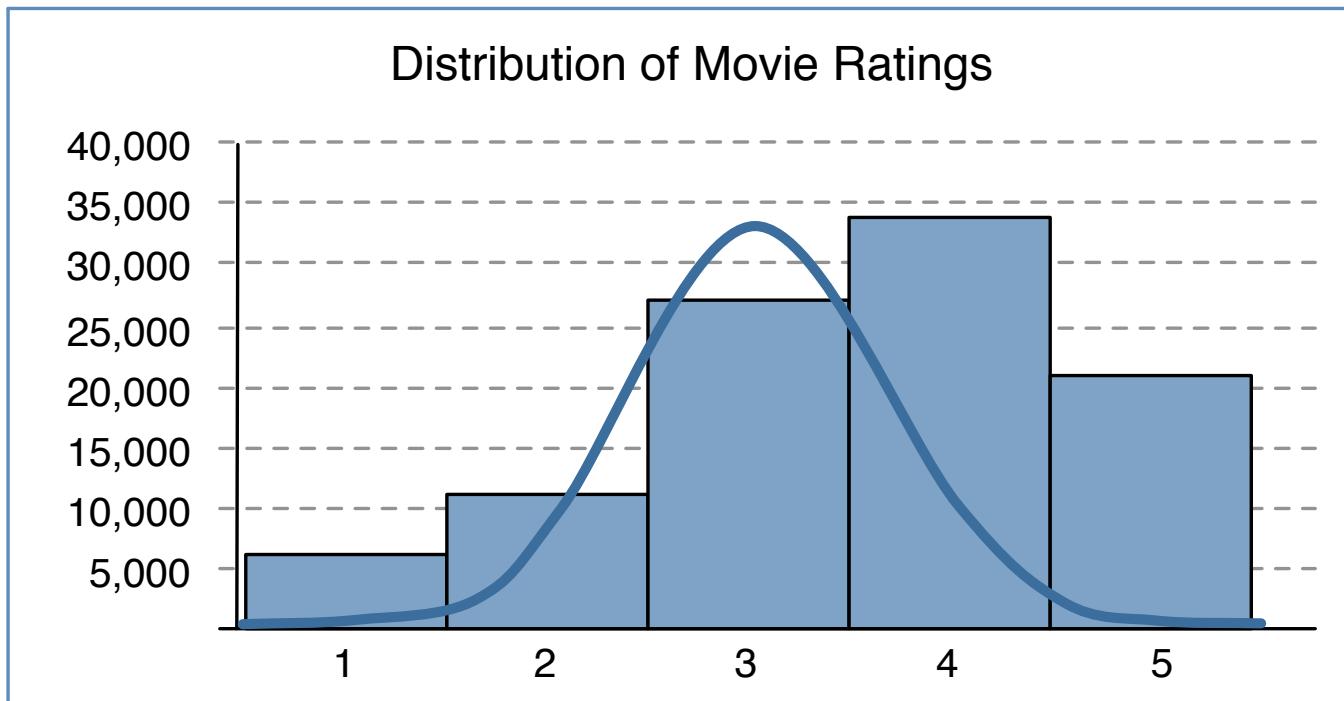
# Skewed Distributions

- Data often deviates from the normal distribution
  - Skewed data is asymmetrically concentrated away from mean
- Skew is natural, but you should understand the reason for it



## Skewed Distributions (cont'd)

- Our set of movie ratings are also skewed
  - The mean (average) value *should* be 3, but is actually 3.53
  - This is because the mode (most common value) is 4
- What's the cause of the inflated ratings?



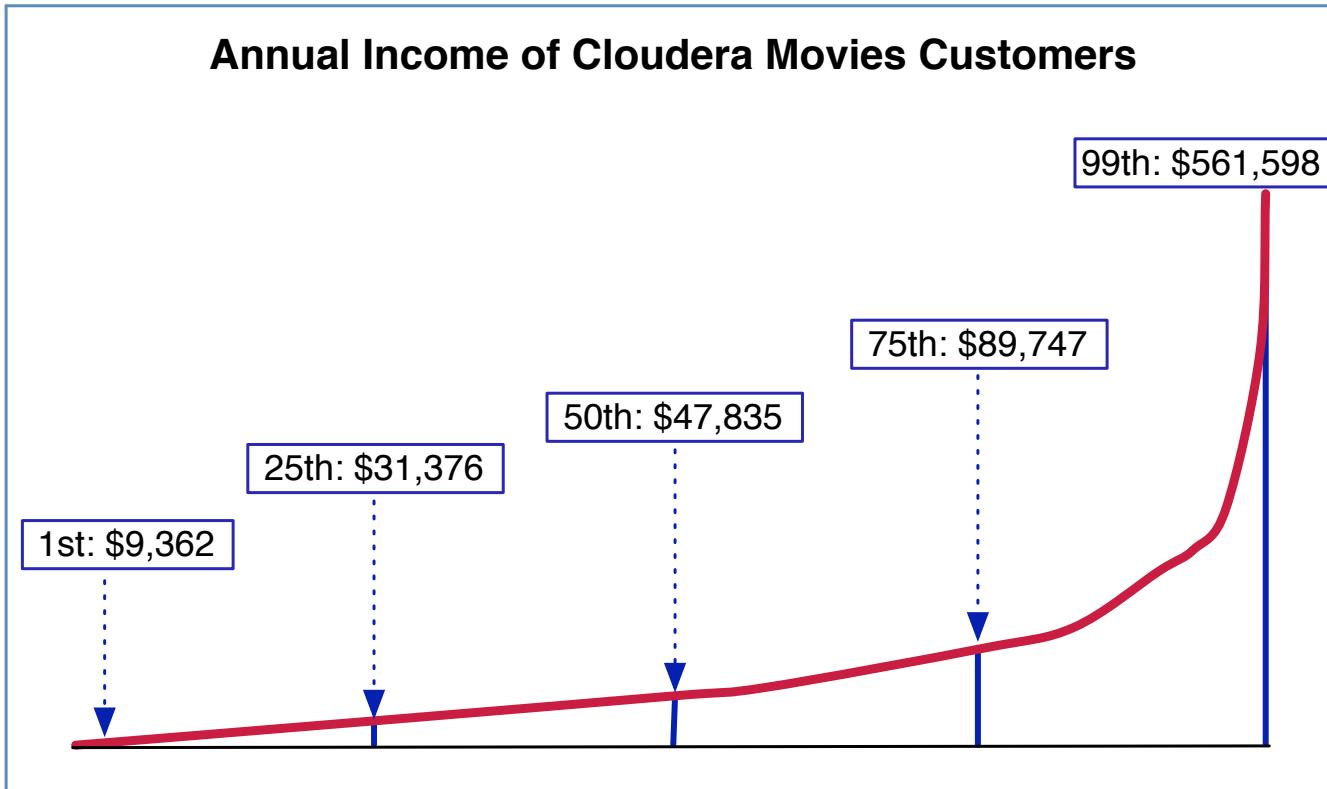
# The Effects of Extremal Values

---

- **Extremal values are common in real-world data sets**
  - These are very large, very small, or very rare values
- **These values make averages (mean) misleading**
  - In such cases, the median is a better measure of what's typical
- **Example: Cloudera Movies customer's annual income**
  - Mean household income of our customers is \$127,396
  - The mean is skewed by a few very affluent customers
  - Our customer's median household income is only \$47,835

# Using Percentiles to Detect Extremal Values

- Percentiles can help to identify extremal values
  - They represent the point below which a percentage of values fall
  - The “typical customer” lies between the 25<sup>th</sup> and 75<sup>th</sup> percentiles



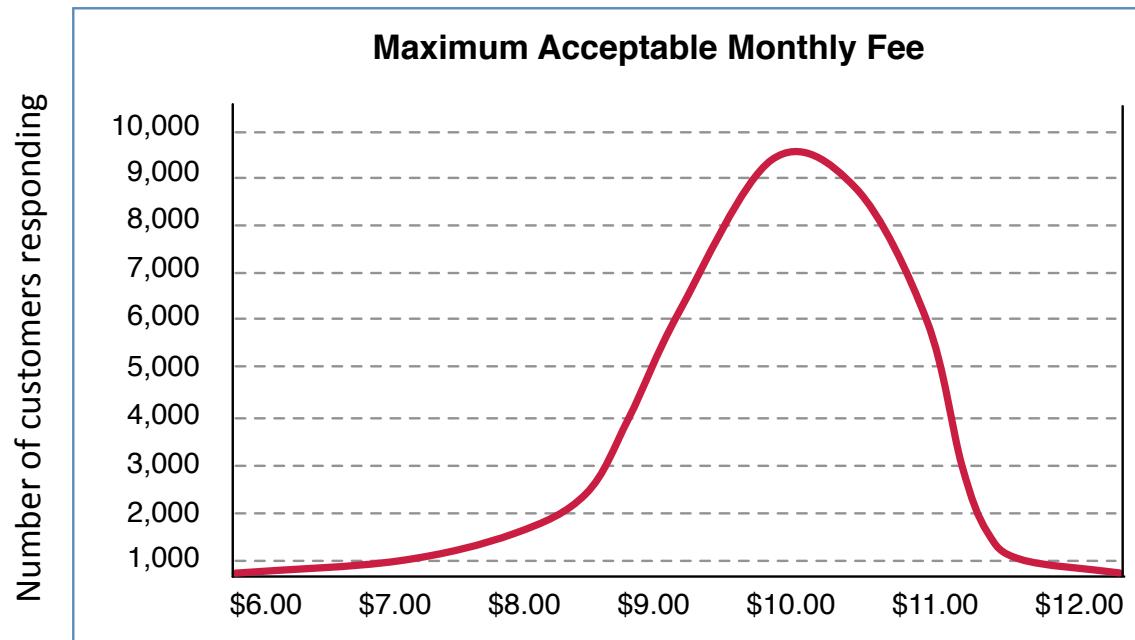
# Chapter Topics

## Data Analysis and Statistical Methods

- Relationship between statistics and probability
- Descriptive statistics
- **Inferential statistics**
- Review questions
- Essential points
- Conclusion

# Inferential Statistics

- **Inferential statistics attempts to draw conclusions based on data**
  - You're estimating the parameters that led to output observed
  - This will help you to determine what to do next in your analysis
- **Consider the results of this Cloudera Movies customer survey**
  - Inferential statistics can help us evaluate *why*



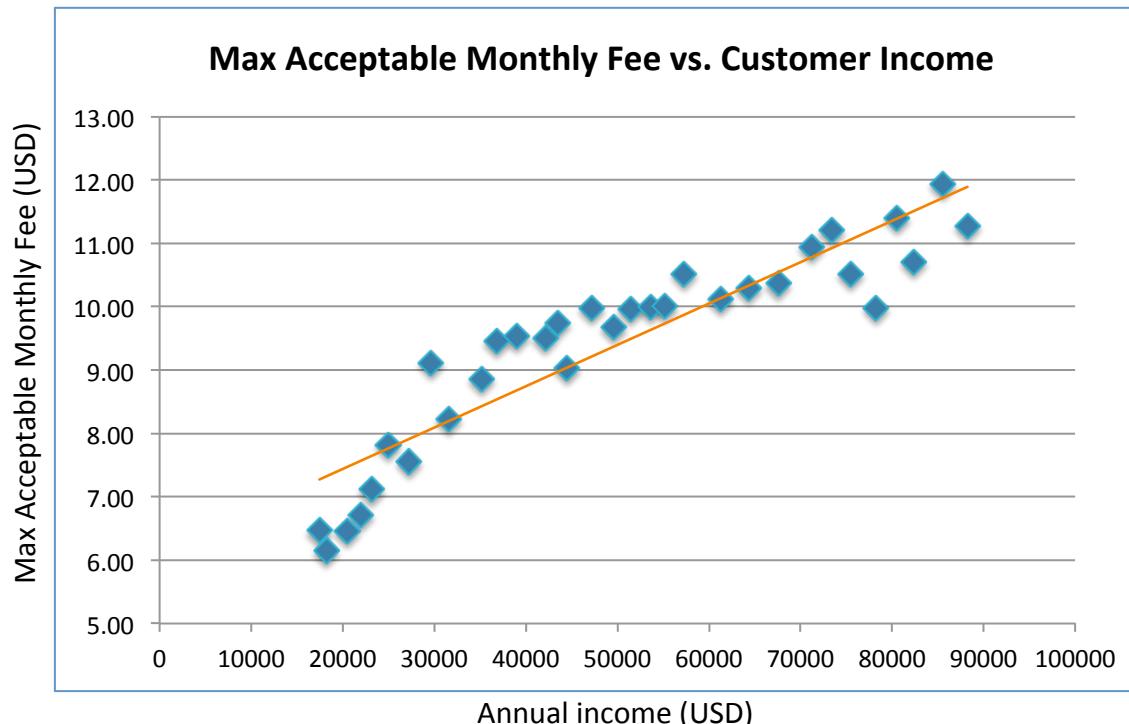
# Dependent and Independent Variables

---

- **There are two main variables to consider**
  - Dependent
    - The output result we're interested in measuring
  - Independent
    - Input parameter(s) we're testing for its effect on output
- **You must account for covariates**
  - These are input parameters that might also affect result
  - They aren't tested, but must be controlled to avoid interference

# Regression Analysis

- A statistical technique for analyzing relationship between these variables
  - How much does the dependent variable change, given a corresponding change in only one of the independent variables?
- For example, how does income affect the fee a customer is willing to pay?



# Regression Analysis and Variable Types

---

- **Several factors might affect what a customer is willing to pay**
  - Which regression analysis technique is appropriate depends on the type of *dependent* variable you need to analyze
- **Continuous variables have an unbounded range of values**
  - Customer's annual income
  - Customer's age
- **Categorical variables have a finite set of values**
  - Movie rating on a scale of 1 to 5
  - Customer's state of residence
- **Binary variables are a subset of categorical variables with only two values**
  - Gender (male or female)
  - Do you subscribe to cable television (yes or no)

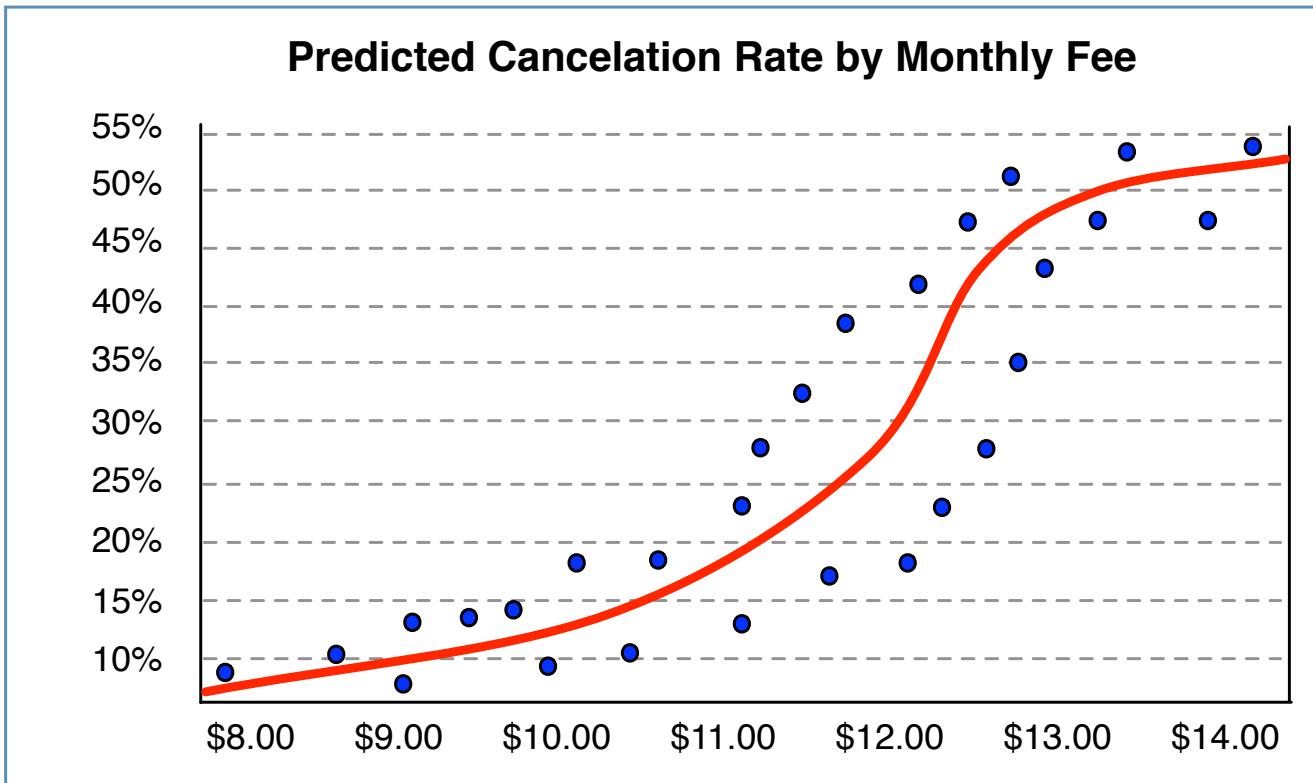
# Linear Regression

---

- **Appropriate for continuous dependent variables**
  - Such as income, height, duration, age, speed, or temperature
- **Basic formula for linear regression:  $Y = X\beta + \epsilon$** 
  - Y is the dependent variable
  - X is an independent variable
  - Beta is a coefficient that show change in Y per change in X
  - Epsilon represents a random distribution of error
- **Observations for X must be independent of one another**
  - Number of DVDs in our catalog one year influences next year
  - Multiple ratings from the same user are correlated

# Logistic Regression

- Appropriate for binary dependent variables
  - Such as “is or is not spam” or “did or did not click on ad”
  - Can be used to model likelihood of a boolean action occurring



# Chapter Topics

## Data Analysis and Statistical Methods

- Relationship between statistics and probability
- Descriptive statistics
- Inferential statistics
- **Review questions**
- Essential points
- Conclusion

## Review Questions

---

- **What might you do to identify extreme data for testing?**
- **Consider the following hypothesis: “A customer is more likely to assign a higher rating to a given movie when other customers in the same ZIP code gave the same movie a higher rating than it received elsewhere.”**
  - What is the dependent variable?
  - What is the independent variable?
  - What are the covariates?
- **What variable type (continuous or binary) are the following?**
  - Whether or not customer subscribes to cable TV
  - Length of time at job
  - Value of customer’s residence

# Chapter Topics

## Data Analysis and Statistical Methods

- Relationship between statistics and probability
- Descriptive statistics
- Inferential statistics
- Review questions
- **Essential points**
- Conclusion

## Essential Points

---

- Scatterplots show both numeric errors and interesting patterns
- Data distribution is a key first step in analysis
- Skewed distributions and extreme values can mislead you
- Testing edge cases is important and biased sampling can help
- Correlation does not imply causation

# Chapter Topics

## Data Analysis and Statistical Methods

- Relationship between statistics and probability
- Descriptive statistics
- Inferential statistics
- Review questions
- Essential points
- Conclusion

# Data Analysis and Statistical Methods

---

## In this chapter you have learned

- How statistics and probability are related
- How scatterplots can help you identify numeric errors
- How to evaluate data distribution
- How extreme values can mislead you
- How regression analysis can help to predict missing values
- Which types of variables are important in regression analysis

# Bibliography

---

The following offer more information on topics discussed in this chapter

- **Causal inference in statistics: An overview (Judea Pearl)**
  - <http://tiny.cloudera.com/dscc08a>
- **Head First Data Analysis**
  - <http://tiny.cloudera.com/dscc08b>
- **The Art of R Programming**
  - <http://tiny.cloudera.com/dscc08c>
- **The Future of Data Analysis**
  - <http://tiny.cloudera.com/dscc08d>
- **Regression Modeling Strategies**
  - <http://tiny.cloudera.com/dscc08e>

# Fundamentals of Machine Learning

---

## Chapter 9



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- **Fundamentals of Machine Learning**
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Fundamentals of Machine Learning

---

## In this chapter you will learn

- What machine learning is
- What are three common machine learning techniques
- How organizations are applying these techniques
- What is the relationship between algorithms and data volume
- How the Naïve Bayes classification algorithm uses probabilities

# Chapter Topics

## Fundamentals of Machine Learning

### ■ Overview

- The three C's of machine learning
- Importance of data and algorithms
- Spotlight: Naïve Bayes classifiers
- Review questions
- Hands-On Exercise: Analysis of social media
- Essential points
- Conclusion

# Fundamentals of Computer Programming

- Let's first consider how a typical program works
  - Hardcoded conditional logic
  - Predefined reactions when those conditions are met

```
$ cat spam-filter.py
#!/usr/bin/env python

import sys

for line in sys.stdin:
    if "Make MONEY Fa$t At Home!!!" in line:
        print "This message is likely spam"

    if "Happy Birthday from Aunt Betty" in line:
        print "This message is probably OK"
```

- The programmer must consider all possibilities at design time
- An alternative technique is to have computers *learn* what to do

# What is Machine Learning

---

- **Machine learning is a field within artificial intelligence (AI)**
  - AI: “the science and engineering of making intelligent machines”
- **Machine learning focuses on automated knowledge acquisition**
  - Primarily through the design and implementation of algorithms
  - These algorithms require empirical data as input
- **Machine learning algorithms “learn” based on input provided**
  - Amount of data is often more important than the algorithm itself

# What is Machine Learning (cont'd)

---

- **The output produced varies by application**
  - Product recommendations
  - Items grouped based on similarity
  - Possible diagnosis of a disease
- **These are examples of “The Three C’s” of machine learning**

# Chapter Topics

## Fundamentals of Machine Learning

- Overview
- **The three C's of machine learning**
- Importance of data and algorithms
- Spotlight: Naïve Bayes classifiers
- Review questions
- Hands-On Exercise: Analysis of social media
- Essential points
- Conclusion

# The ‘Three Cs’

---

- **Three established categories of machine learning techniques:**
  - Collaborative filtering (recommendations)
  - Clustering
  - Classification
- **This course will focus on collaborative filtering**
  - Though we'll also cover a simple classification algorithm

# Collaborative Filtering

---

- Collaborative filtering is a technique for recommendations
  - It's one primary type of recommender system
  - We'll cover it in detail during the next several chapters
- Helps users find items of relevance
  - Among a potentially vast number of choices
  - Based on comparison of preferences between users



# Applications Involving Collaborative Filtering

- Collaborative filtering is domain agnostic
- Can use the same algorithm to recommend practically anything
  - Movies (Cloudera Movies... oh, and Netflix too)
  - Television (TiVO Suggestions)
  - Music (Several popular music download and streaming services)
- Amazon uses CF to recommend a variety of products

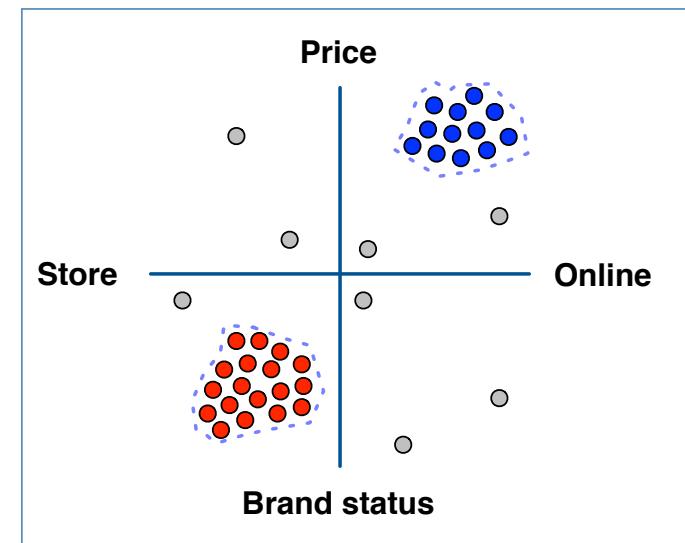
Continue Shopping: Customers Who Bought Items in Your Recent History Also Bought

Page 11 of 13 (Start over)

 C Programming Language (2nd Edition) by Brian W. Kernighan ★★★★★ (323) Paperback \$45.99 <a href="#">Fix this recommendation</a>	 Grand Theft Auto Vice City ★★★★★ (1,368) \$7.28 <a href="#">Fix this recommendation</a>	 Louder Than Love by Soundgarden ★★★★★ (58) Audio CD \$7.04 <a href="#">Fix this recommendation</a>	 Scotch Desktop Tape Dispenser... ★★★★★ (9) \$6.01 <a href="#">Fix this recommendation</a>
--------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------

# Clustering

- **Clustering algorithms discover structure in collections of data**
  - Where no formal structure previously existed
- **They discover what clusters ('groupings'), naturally occur in data**
  - By examining various properties of the input data
- **Clustering is often used for exploratory analysis**
  - Divide huge amount of data into smaller groups
  - Can then tune analysis for each group



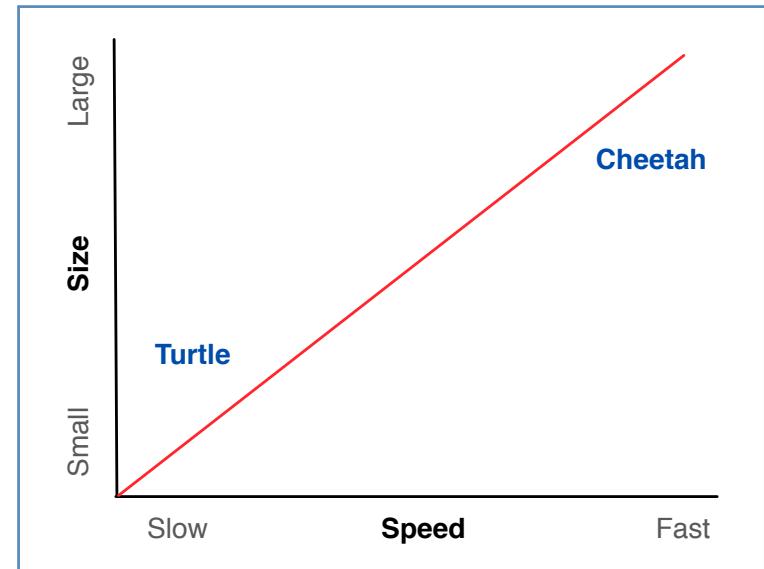
# Applications Involving Clustering

---

- **Market segmentation**
  - Group similar customers in order to target them effectively
- **Finding related news articles**
  - Google News
- **Epidemiological studies**
  - For example, identifying “cancer cluster” and finding root cause
- **Computer vision (groups of pixels that cohere into objects)**
  - Related pixels clustered to recognize faces or license plates

# Classification

- The previous two techniques are *unsupervised learning*
  - The algorithm discovers recommendations or groups
- Classification is a form of ‘supervised’ learning
  - Requires training with data that has known labels
    - These are healthy cells, those are cancerous
  - Learns how to label new records based on that information



# Applications Involving Classification

---

- **Spam filtering**

- Train using a set of spam and non-spam messages
  - System will eventually learn to detect unwanted e-mail

- **Oncology**

- Train using images of benign and malignant tumors
  - System will eventually learn to identify cancer

- **Risk Analysis**

- Train using financial records of customers who do/don't default
  - System will eventually learn to identify risk customers

# Chapter Topics

## Fundamentals of Machine Learning

- Overview
- The three C's of machine learning
- **Importance of data and algorithms**
- Spotlight: Naïve Bayes classifiers
- Review questions
- Hands-On Exercise: Analysis of social media
- Essential points
- Conclusion

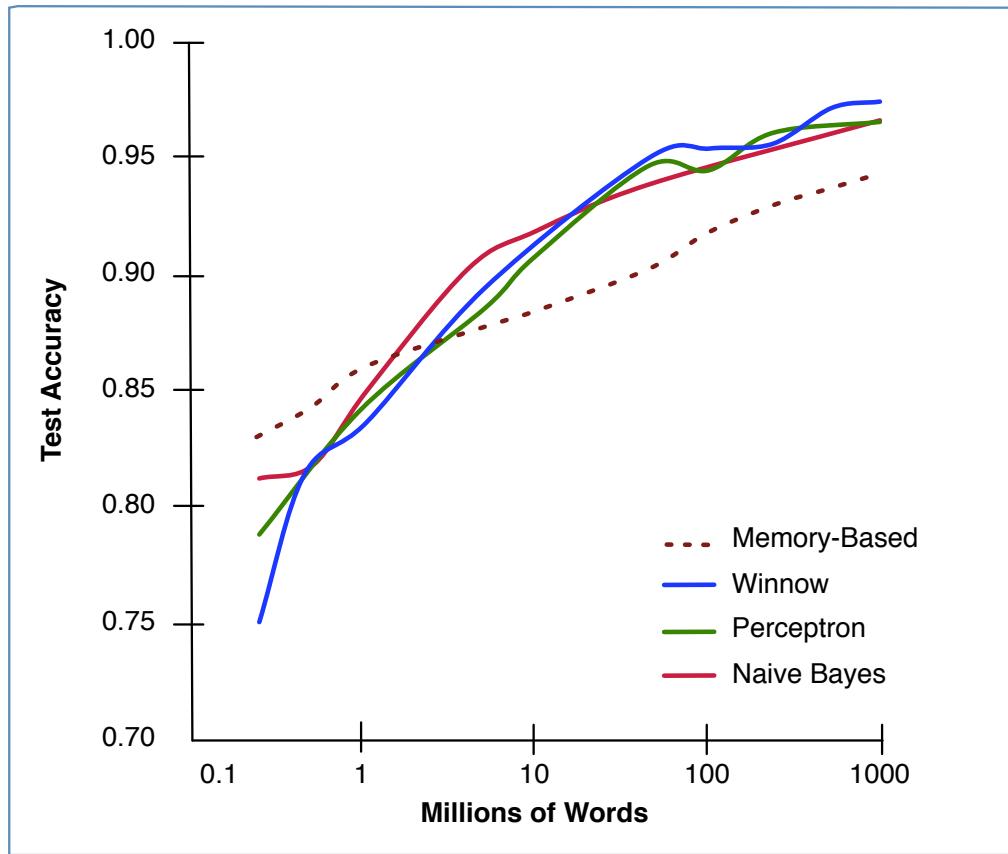
# Relationship of Algorithms and Data Volume

---

- **There are many algorithms for each type of machine learning**
  - There's no overall “best” algorithm
  - Each algorithm has advantages and limitations
- **Algorithm choice is often related to data volume**
  - Some scale better than others
- **Most algorithms offer better results as volume increases**
  - Best approach = simple algorithm + lots of data

## Relationship of Algorithms and Data Volume (cont'd)

“It’s not who has the best algorithms that wins.  
It’s who has the most data.” [Banko and Brill, 2001]



# Chapter Topics

## Fundamentals of Machine Learning

- Overview
- The three C's of machine learning
- Importance of data and algorithms
- **Spotlight: Naïve Bayes classifiers**
- Review questions
- Hands-On Exercise: Analysis of social media
- Essential points
- Conclusion

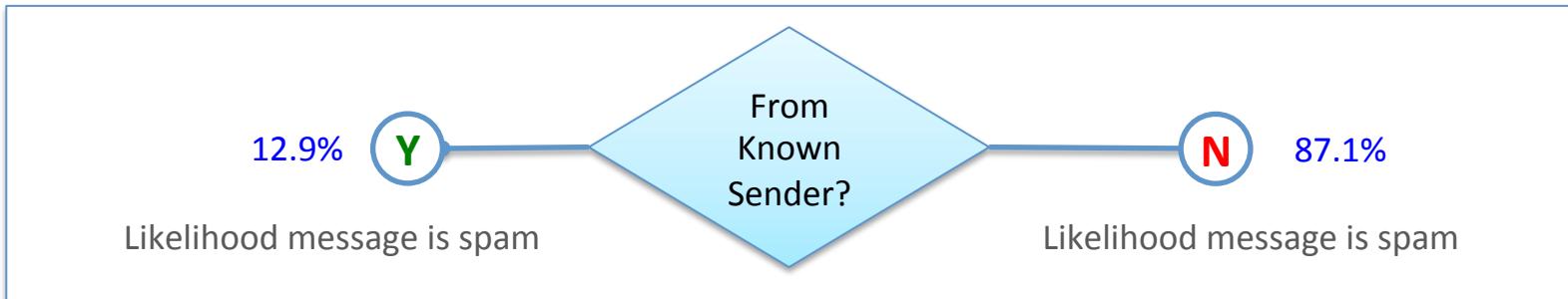
# Naïve Bayes Classifiers

---

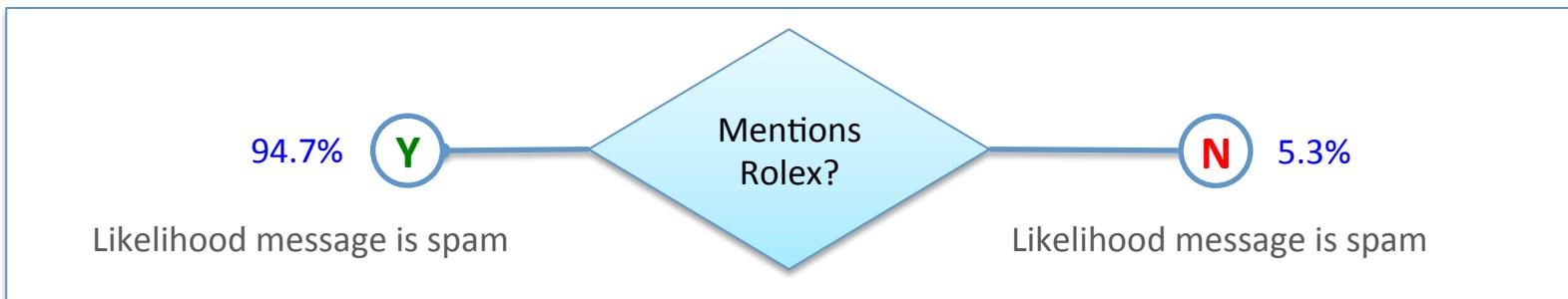
- **Naïve Bayes is a simple – but effective – classification algorithm**
- **Based on the concept of conditional probability**
  - How likely is outcome Z, given conditions X and Y
  - Each condition is evaluated independently
- **Each condition must satisfy two constraints**
  - It must be independent of every other condition
  - All of the independent variables must be binary – no continuous variables allowed
- **Spam filtering is a classic example of naïve Bayes classification**

## Naïve Bayes Classifiers (cont'd)

- We've analyzed our inbox and found the following
  - 87.1% of messages from unknown senders are spam

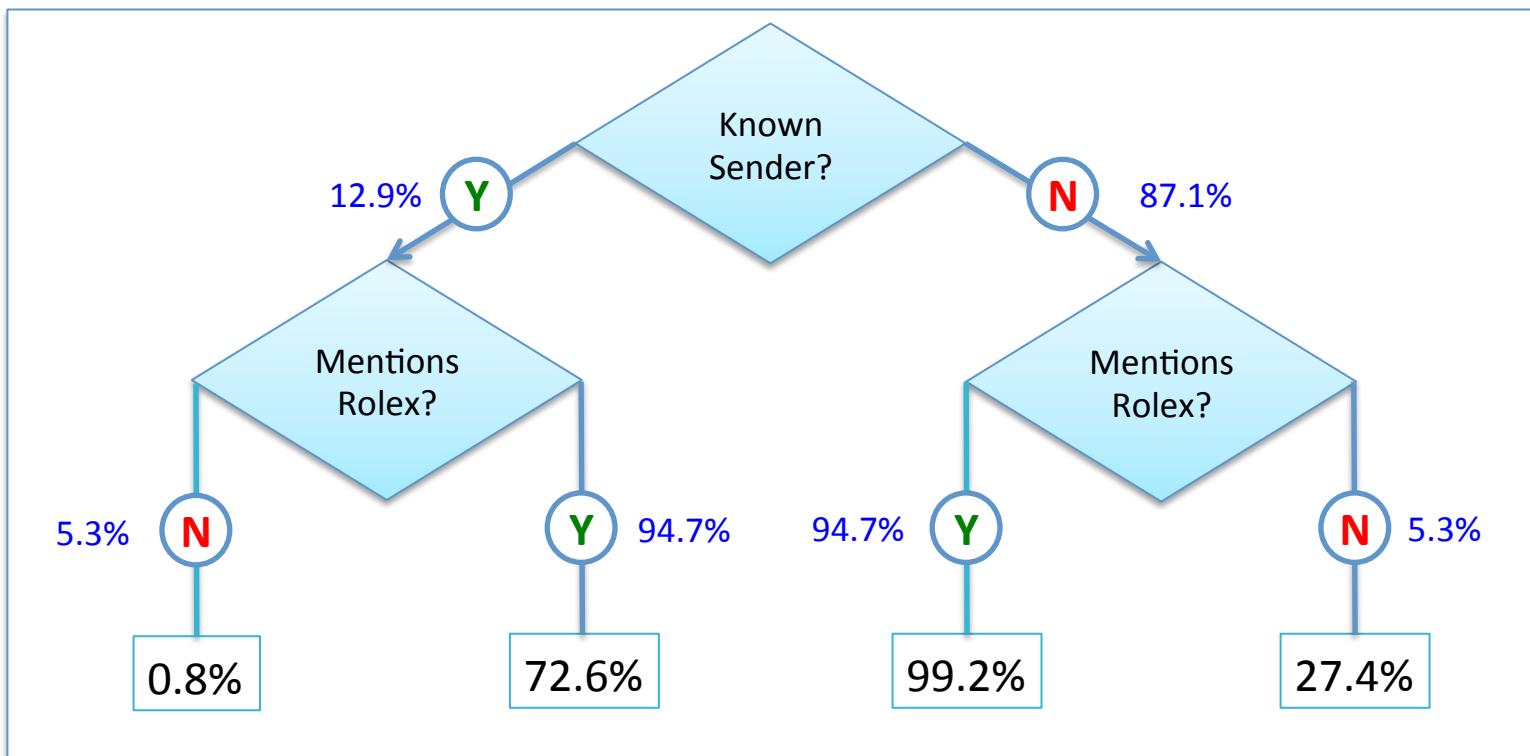


- 94.7% of messages mentioning Rolex are spam



## Naïve Bayes Classifiers (cont'd)

- Applying the result of multiple tests improves overall detection
  - We can predict how likely a message from an unknown sender that mentions Rolex is likely to be spam



# Applications of Bayesian Classifiers

---

- **Bayesian classification can do a lot more than filter spam**
  - Medical diagnosis
  - Root cause analysis
  - Prediction of loan default

# Chapter Topics

## Fundamentals of Machine Learning

- Overview
- The three C's of machine learning
- Importance of data and algorithms
- Spotlight: Naïve Bayes classifiers
- **Review questions**
- Hands-On Exercise: Analysis of social media
- Essential points
- Conclusion

## Review Questions

---

- **What are the three C's of machine learning?**
- **Classification algorithms like Naïve Bayes are used in many areas of everyday life. What are some ways it might help Cloudera Movies?**

# Chapter Topics

## Fundamentals of Machine Learning

- Overview
- The three C's of machine learning
- Importance of data and algorithms
- Spotlight: Naïve Bayes classifiers
- Review questions
- **Hands-On Exercise: Analysis of social media**
- Essential points
- Conclusion

# Hands-on Exercise: Analysis of Social Media

---

- In this Hands-On Exercise, you will gain practice performing statistical analysis on the Cloudera Movie customer data
  - You will use R and Python to analyze the social media data collected about our customers. This information provides you insight into which movies they prefer and will be used to improve our recommender in a subsequent hands-on exercise.
- Please refer to the Hands-On Exercise Manual for instructions on exercise #4

# Chapter Topics

## Fundamentals of Machine Learning

- Overview
- The three C's of machine learning
- Importance of data and algorithms
- Spotlight: Naïve Bayes classifiers
- Review questions
- Hands-On Exercise: Analysis of social media
- **Essential points**
- Conclusion

## Essential Points

---

- Machine learning algorithms “learn” based on data provided
- Collaborative filtering recommends items
- Clustering discovers how to group a set of items into subsets
- Classification is supervised learning that can identify item types
- More data is usually preferable to a better algorithm

# Chapter Topics

## Fundamentals of Machine Learning

- Overview
- The three C's of machine learning
- Importance of data and algorithms
- Spotlight: Naïve Bayes classifiers
- Review questions
- Hands-On Exercise: Analysis of social media
- Essential points
- **Conclusion**

# Fundamentals of Machine Learning

---

## In this chapter you have learned

- What machine learning is
- What are three common machine learning techniques
- How organizations are applying these techniques
- What is the relationship between algorithms and data volume
- How the Naïve Bayes classification algorithm uses probabilities

# Bibliography

---

The following offer more information on topics discussed in this chapter

- **Programming Collective Intelligence**

- <http://tiny.cloudera.com/dscc09a>

- **Andrew Ng's Online Course on Machine Learning at Coursera**

- <http://tiny.cloudera.com/dscc09b>

- **Learning With Large Datasets**

- <http://tiny.cloudera.com/dscc09c>

- **The Elements of Statistical Learning**

- <http://tiny.cloudera.com/dscc09d>

- **Machine Learning: A Probabilistic Perspective**

- <http://tiny.cloudera.com/dscc09e>

# Recommender Overview

---

## Chapter 10



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- **Recommender Overview**
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Recommender Overview

---

## In this chapter you will learn

- What is the difference between content-based and collaborative filtering recommender systems
- Which limitations recommender systems frequently encounter
- How collaborative filtering can identify similar users and items
- How Tanimoto and Euclidean distance similarity metrics work

# Chapter Topics

## Recommender Overview

- **What is a recommender system?**
- Types of collaborative filtering
- Limitations of recommender systems
- Fundamental concepts
- Review questions
- Hands-On Exercise: Implementing a Basic Recommender
- Essential points
- Conclusion

# What is a Recommender System?

---

- **Recommenders are a type of filter**
- **They help users find relevant items within a huge selection**
  - How do you find an interesting movie among 95,000 choices?
  - They help you find things you didn't know to look for
- **Recommenders use preferences to predict preferences**
  - Input is feedback about likes and/or dislikes
  - Output is a list of suggested items based on feedback received
- **Two main types of recommenders**
  - Content-based
  - Collaborative filtering

# Content-Based Recommenders

---

- **Content-based recommenders consider an item's attributes**
  - These attributes describe the item
- **Examples of item attributes**
  - Movies: actor, director, screenwriter, producer, and location
  - Music: songwriter, style, musicians, vocalist, meter, and tempo
  - Books: author, publisher, subject, illustrations, and page count
- **A user's taste defines values and weights for each attribute**
  - These are supplied as input to the recommender

# Content-Based Recommenders (cont'd)

---

- **Content-based recommenders are domain-specific**
  - Because attributes don't transcend item types
- **Examples of content-based recommendations**
  - You like 1980's action films starring Chuck Norris, try *Delta Force*
  - You like abstract rock from the 1970's, try *Dark Side of the Moon*

# Collaborative Filtering

---

- **Collaborative filtering is an inherently social system**
  - It recommends items based on preferences of similar users
- **It's similar to how you get recommendations from friends**
  - Query those people who share your interests
  - They'll know movies you haven't seen and would probably like
    - And you'll be able to recommend some to them
- **This approach is not domain-specific**
  - System doesn't “know” anything about the items it recommends
  - The same algorithm can be used to recommend any type of product
- **We'll discuss collaborative filtering in detail during this chapter**

# Hybrid Recommenders

---

- **Content-based and collaborative filtering are two approaches**
- **Each has advantages and limitations**
  - We'll discuss these in a moment
- **It's also possible to combine these approaches**
  - For example, predict rating using content-based approach
  - Then predict rating using collaborative filtering
  - Finally, average these values to create a hybrid prediction
- **Research demonstrates that this can offer better results than using either system on its own**
  - Netflix and other companies use hybrid recommenders

# Chapter Topics

## Recommender Overview

- What is a recommender system?
- **Types of collaborative filtering**
- Limitations of recommender systems
- Fundamental concepts
- Review questions
- Hands-On Exercise: Implementing a Basic Recommender
- Essential points
- Conclusion

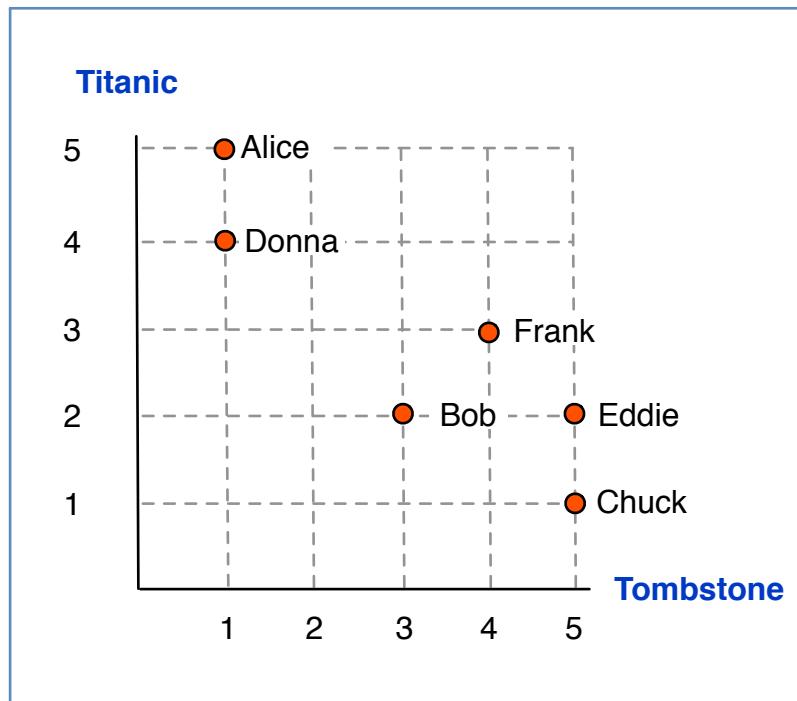
# Types of Collaborative Filtering

---

- **Collaborative filtering can be subdivided into two main types**
- **User-based: “What do users similar to you like?”**
  - For a given user, find other people who have similar tastes
  - Then, recommend items based on past behavior of those users
- **Item-based: “What is similar to other items you like?”**
  - Given items that a user likes, determine which items are similar
  - Make recommendations to the user based on those items

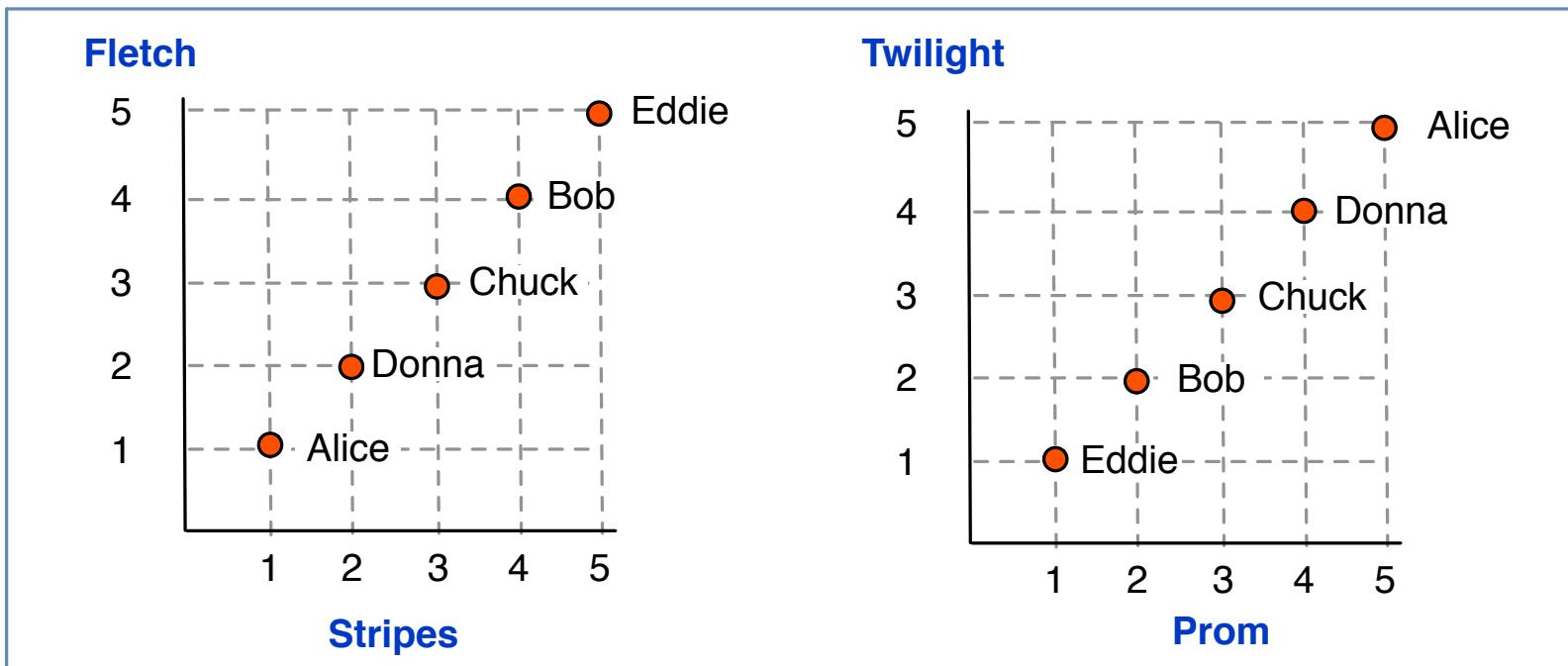
# User-Based Collaborative Filtering

- User-based collaborative filtering is social
  - It takes a “people first” approach, based on common interests
- In this example, Alice and Donna have similar tastes
  - Each is likely to enjoy a movie that the other rated highly



# Item-Based Collaborative Filtering

- After examining more of these ratings, patterns emerge
  - Strong correlations between movies suggest they're similar



## Item-Based Collaborative Filtering (cont'd)

---

- **The item-based approach was popularized by Amazon**
  - Given previous purchases, what would you be likely to buy?
- **Cloudera Movies could also use item-based filtering**
  - Suggest *Stripes* after customer adds *Fletch* to the queue
- **Item-based CF usually scales better than user-based**
  - Successful companies have more users than products

# Chapter Topics

## Recommender Overview

- What is a recommender system?
- Types of collaborative filtering
- **Limitations of recommender systems**
- Fundamental concepts
- Review questions
- Hands-On Exercise: Implementing a Basic Recommender
- Essential points
- Conclusion

# Limitations

---

- **The “cold start problem” is a limitation of collaborative filtering**
  - CF finds recommendations based on actions of similar users
  - So what do you do for a startup?
    - A new service has no users, similar or otherwise!
  - One workaround is to use content-based filtering at first
    - Eventually you’ll have enough data for collaborative filtering
    - You can transition via a hybrid approach as you add users
- **Performance of sparse matrix operations**
  - Cloudera Movies has 14 million customers and 100,000 movies
  - A matrix representation will have 1.4 trillion elements
    - Even active customers have only seen a few hundred movies
    - And they haven’t rated all of these

## Limitations (cont'd)

---

- **People aren't very good at rating things**
  - You may need to identify and correct for individual biases
  - Observe user behavior instead of asking for ratings
- **Individual tastes aren't always predictable**
  - One person may love *Halloween*, *Friday the 13<sup>th</sup>*, and *Saw*
  - Unlike similar users, this person may also love *Mary Poppins*
  - As always, using more input data will likely produce better results
- **A single account may correspond to multiple users**
  - Does the account holder like *Bambi*? Or is it her daughter?

## Limitations (cont'd)

---

- **Item-based CF may predict previously-satisfied needs**
  - The goal of item-based CF is to identify similar products
  - More helpful with pre-purchase suggestions than post-purchase
    - If I bought a toaster, ads for other toasters aren't helpful
    - But ads for bagels and jam might be helpful
  - Not an issue for some products (like movies or music)

# Chapter Topics

## Recommender Overview

- What is a recommender system?
- Types of collaborative filtering
- Limitations of recommender systems
- **Fundamental concepts**
- Review questions
- Hands-On Exercise: Implementing a Basic Recommender
- Essential points
- Conclusion

# Input Data

---

- **The recommender accepts preference data as input**
  - These preferences represent what users like and dislike
  - Content-based recommenders also use attributes about an item
- **Input preferences can be collected in two ways**
  - Explicit: we ask users to rate items that they like or dislike
    - Netflix “star” ratings
    - TiVO “thumbs up” ratings
    - “How would you rank these items?”
  - Implicit: we observe user behavior to determine their preferences
    - Which movies does a customer watch?
    - Does customer move a movie up or down in the queue?
    - Does the customer finish the movie?

# Evaluating Input

---

## ■ How does collaborative filtering work?

- Create a matrix of users and items, populated with preferences
- For a given user, identify other users with similar tastes
- Find items new to this user, but rated highly by similar users

	Alice	Bob	Chuck	Donna	Eddie	Frank	Gina
Airplane	1	4			5		
Bambi	4			5		2	
Caddyshack		4	3		4		5
Dracula			5			4	
Eat Pray Love		2		5	1		1
Friday		4					5
Gunsmoke						4	5
Hang 'Em High			5			4	5
Iron Man			3	1	4		5
Jane Eyre	5						
The Karate Kid	4		5	5	3		

## Evaluating Input (cont'd)

- Donna has preferences similar to Alice

	Alice	Bob	Chuck	Donna	Eddie	Frank	Gina
Airplane	1	4			5		
Bambi	4			5		2	
Caddyshack		4	3		4		5
Dracula			5			4	
Eat Pray Love		2		5	1		1
Friday		4					5
Gunsmoke						4	5
Hang 'Em High			5			4	5
Iron Man			3	1	4		5
Jane Eyre	5						
The Karate Kid	4		5	5	3		

## Evaluating Input (cont'd)

- Based on this, we could recommend *Eat Pray Love* to Alice

	Alice	Bob	Chuck	Donna	Eddie	Frank	Gina
Airplane	1	4			5		
Bambi	4			5		2	
Caddyshack		4	3		4		5
Dracula			5			4	
Eat Pray Love		2		5	1		1
Friday		4					5
Gunsmoke						4	5
Hang 'Em High			5			4	5
Iron Man			3	1	4		5
Jane Eyre	5						
The Karate Kid	4		5	5	3		

## Evaluating Input (cont'd)

- Similarly, we could recommend *Jane Eyre* to Donna

	Alice	Bob	Chuck	Donna	Eddie	Frank	Gina
Airplane	1	4			5		
Bambi	4			5		2	
Caddyshack		4	3		4		5
Dracula			5			4	
Eat Pray Love		2		5	1		1
Friday		4					5
Gunsmoke						4	5
Hang 'Em High			5			4	5
Iron Man			3	1	4		5
Jane Eyre	5						
The Karate Kid	4		5	5	3		

## Evaluating Input (cont'd)

- More users mean stronger signals and better recommendations
  - Whose preferences are similar to Bob?

	Alice	Bob	Chuck	Donna	Eddie	Frank	Gina
Airplane	1	4			5		
Bambi	4			5		2	
Caddyshack		4	3		4		5
Dracula			5			4	
Eat Pray Love		2		5	1		1
Friday		4					5
Gunsmoke						4	5
Hang 'Em High			5			4	5
Iron Man			3	1	4		5
Jane Eyre	5						
The Karate Kid	4		5	5	3		

## Evaluating Input (cont'd)

- Both Eddie and Gina's preferences are similar to Bob
  - Ratings they share produce better recommendations for Bob

	Alice	Bob	Chuck	Donna	Eddie	Frank	Gina
Airplane	1	4			5		
Bambi	4			5		2	
Caddyshack		4	3		4		5
Dracula			5			4	
Eat Pray Love		2		5	1		1
Friday		4					5
Gunsmoke						4	5
Hang 'Em High				5		4	5
Iron Man				3	1	4	
Jane Eyre	5						
The Karate Kid	4			5	5	3	

## Evaluating Input (cont'd)

- We could recommend **Gunsmoke**, **Karate Kid**, or **Iron Man** to Bob
  - Highest confidence about Iron Man, based on stronger signal

	Alice	Bob	Chuck	Donna	Eddie	Frank	Gina
Airplane	1	4			5		
Bambi	4			5		2	
Caddyshack		4	3		4		5
Dracula			5			4	
Eat Pray Love		2		5	1		1
Friday		4					5
Gunsmoke						4	5
Hang 'Em High				5		4	5
Iron Man			3	1	4		5
Jane Eyre	5						
The Karate Kid	4		5	5	3		

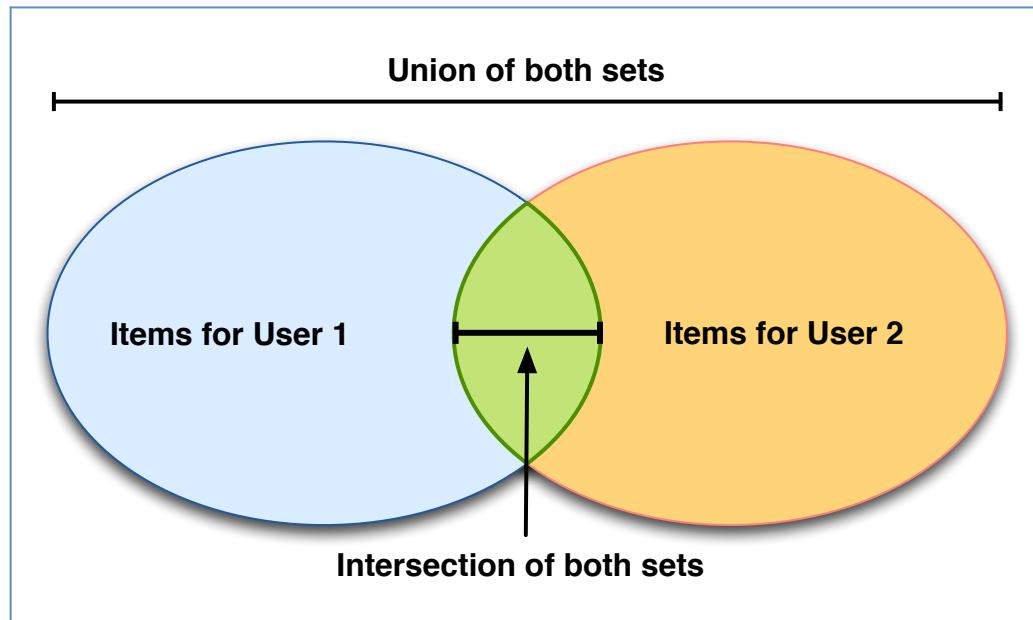
# Basic Similarity Metrics

---

- **It's easy for humans to see similarities between users**
  - But how can a computer find these similarities?
  - More importantly, how we can measure them?
- **There are many similarity metrics**
  - We'll briefly cover two now, and discuss several in depth later
- **Choosing one involves several factors, including**
  - The type of preference data available
  - Performance at scale
- **They work by comparing vectors of data**
  - The elements could be users or items
  - You need to calculate metrics for every pair

# Tanimoto Coefficient

- Tanimoto coefficient is applicable when you have binary (boolean) data
  - “Did customer watch a given movie or not?”
  - “Did customer finish this movie or not?”
- Also known as the Jaccard coefficient, Tanimoto compares two sets
  - Based on the ratio of union (all items) and intersection (common items)



## Tanimoto Coefficient (cont'd)

- The Tanimoto coefficient is easy to compute in Python

```
def tanimoto(set_a, set_b):
    intersection = set_a.intersection(set_b)

    len_a = len(set_a)
    len_b = len(set_b)
    len_i = len(intersection)

    return float(len_i) / (len_a + len_b - len_i)
```

- The value ranges between 0.0 and 1.0
  - A value of 1.0 indicates both sets exactly match one another
  - Value moves towards 0.0 as number of common items decreases

## Tanimoto Coefficient (cont'd)

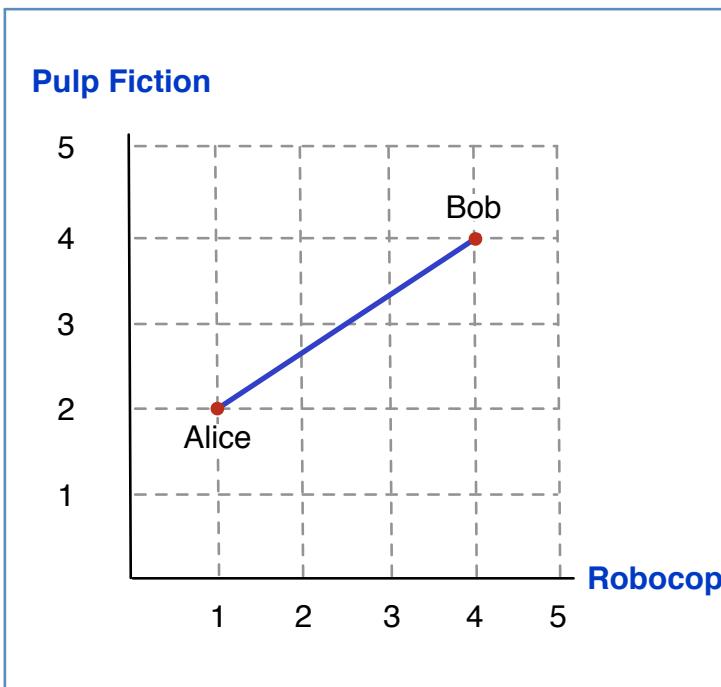
- Consider the following input
  - An 'X' in the matrix below indicates customer watched the movie

	Alice	Frank	Gina
Airplane		X	X
Bambi	X	X	
Caddyshack		X	X
Eat Pray Love	X		
Gunsmoke		X	X
Hang 'Em High		X	X

- Frank and Gina share similar taste (value = 0.8)
- But Alice and Gina don't (value = 0.0)

# Euclidean Distance

- Euclidean distance is a measure of similarity for numeric data
  - “How many stars did the customer give this movie?”
  - “How many times did the customer watch this movie?”
- Effectively the same as plotting it and measuring with a ruler



## Euclidean Distance (cont'd)

- Euclidean distance is also easy to calculate in Python
  - Simple calculation based on parallel elements from each list

```
def euclidean(list_a, list_b):  
    dist = 0.0  
    for i in range(len(list_a)):  
        rate_a = list_a[i]  
        rate_b = list_b[i]  
  
        dist = dist + pow((rate_a - rate_b), 2)  
    return sqrt(dist)
```

- A lower number indicates a stronger similarity
  - Though this is often inverted to provide a value in the 0.0 – 1.0 range

## Euclidean Distance (cont'd)

---

- Consider the following input
  - Each element in the matrix below is the user's rating of a movie

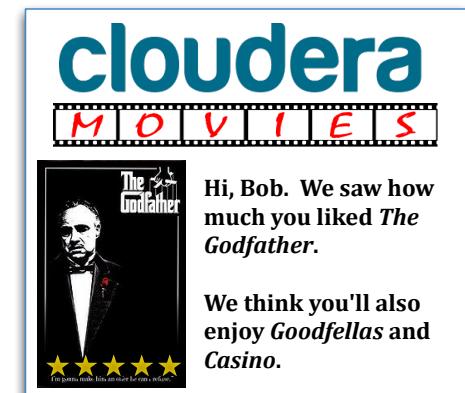
	Alice	Frank	Gina
Airplane	1	4	5
Bambi	4	2	1
Caddyshack	2	4	5
Eat Pray Love	5	1	1
Gunsmoke	1	5	5
Hang 'Em High	1	4	5

- Frank and Gina's preferences are close (distance of 2.0)
  - Alice and Gina's preferences aren't (distance of 9.05)

# Recommender Output

---

- Quick recap of how a user-based recommender works
  - Takes preference data as input
  - It finds similar users based on similarity metrics
- What does a recommender produce as output?
  - A list of items along with the predicted ratings for each
- What do we do with this output?
  - Remove items known to be of little value
  - Sort remaining items in descending order of predicted rating
  - Present this to the user in the application



# Chapter Topics

## Recommender Overview

- What is a recommender system?
- Types of collaborative filtering
- Limitations of recommender systems
- Fundamental concepts
- **Review questions**
- Hands-On Exercise: Implementing a Basic Recommender
- Essential points
- Conclusion

## Review Questions

---

- **What are some ways in which Cloudera Movies might gather implicit preferences from our customers?**
- **How might Cloudera Movies learn which movie attributes are important to a customer?**

# Chapter Topics

## Recommender Overview

- What is a recommender system?
- Types of collaborative filtering
- Limitations of recommender systems
- Fundamental concepts
- Review questions
- **Hands-On Exercise: Implementing a Basic Recommender**
- Essential points
- Conclusion

## Hands-on Exercise: Basic Recommender

---

- In this Hands-On Exercise, you will build a simple recommender system in Python using the techniques you've just learned
- Please refer to the Hands-On Exercise Manual for instructions on exercise #5

# Chapter Topics

## Recommender Overview

- What is a recommender system?
- Types of collaborative filtering
- Limitations of recommender systems
- Fundamental concepts
- Review questions
- Hands-On Exercise: Implementing a Basic Recommender
- **Essential points**
- Conclusion

## Essential Points

---

- **Recommenders are filtering systems**
- **Content-based recommenders consider item attributes**
- **Collaborative filters consider actions of other users**
- **Preferences can be collected implicitly or explicitly**
- **Similarity metrics are chosen, in part, based on data type**

# Chapter Topics

## Recommender Overview

- What is a recommender system?
- Types of collaborative filtering
- Limitations of recommender systems
- Fundamental concepts
- Review questions
- Hands-On Exercise: Implementing a Basic Recommender
- Essential points
- **Conclusion**

# Recommender Overview

---

In this chapter you have learned

- What is the difference between content-based and collaborative filtering recommender systems
- Which limitations recommender systems frequently encounter
- How collaborative filtering can identify similar users and items
- How Tanimoto and Euclidean distance similarity metrics work

# Bibliography

---

**The following offer more information on topics discussed in this chapter**

- **Recommender Systems: An Introduction**

- <http://tiny.cloudera.com/dscc10a>

- **Amazon's Original Item-Item Recommendations Paper**

- <http://tiny.cloudera.com/dscc10b>

# Introduction to Apache Mahout

---

## Chapter 11



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- **Introduction to Apache Mahout**
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Introduction to Apache Mahout

---

## In this chapter you will learn

- **What Apache Mahout is**
- **How it was developed**
- **What's required to run Apache Mahout**
- **How you can use Mahout for Collaborative Filtering**

# Chapter Topics

## Introduction to Apache Mahout

- **What Apache Mahout is (and is not)**
- Brief history
- Availability and installation
- Review questions
- Optional Demo: Using Mahout's Item-Based Recommender
- Essential points
- Conclusion

# What is Apache Mahout?

---

- **Apache Mahout is an open source machine learning library**
  - Focuses on real world use cases, not academic ones
  - Many of its algorithms can use Hadoop for improved scalability
- **Mahout is derived from the Hindi word for “elephant driver”**
  - There's some debate about pronunciation
- **Mahout has extensive support for collaborative filtering**
  - Handles both user-based and item-based approaches
  - Includes Tanimoto, Euclidean and many more similarity metrics



# What is Apache Mahout? (cont'd)

---

- **Mahout is a collection of algorithms**
  - Mainly focused on “The Three C’s” of machine learning
- **It also provides some helpful utility classes**
  - Format conversion and basic visualization tools
- **Mahout is implemented in Java**
  - You can extend it by writing Java code that uses its API
  - But you don’t have to be a Java programmer to *use* it
    - You can invoke it from the command line (we’ll see how)
    - Could also make it accessible via a Web service

# What Apache Mahout Is Not

---

- **A turnkey solution for content-based recommenders**
  - Though you can use Mahout to *help you* build one
- **A cohesive system**
  - Mahout is an “umbrella” project with many subcomponents
    - These were contributed by different people over time
    - Developers work independently on their preferred approaches
  - Documentation isn’t always consistent
- **Always user-friendly**
  - It’s meant for developers, not end users
  - Can be finicky (leftover temp files may cause jobs to fail)

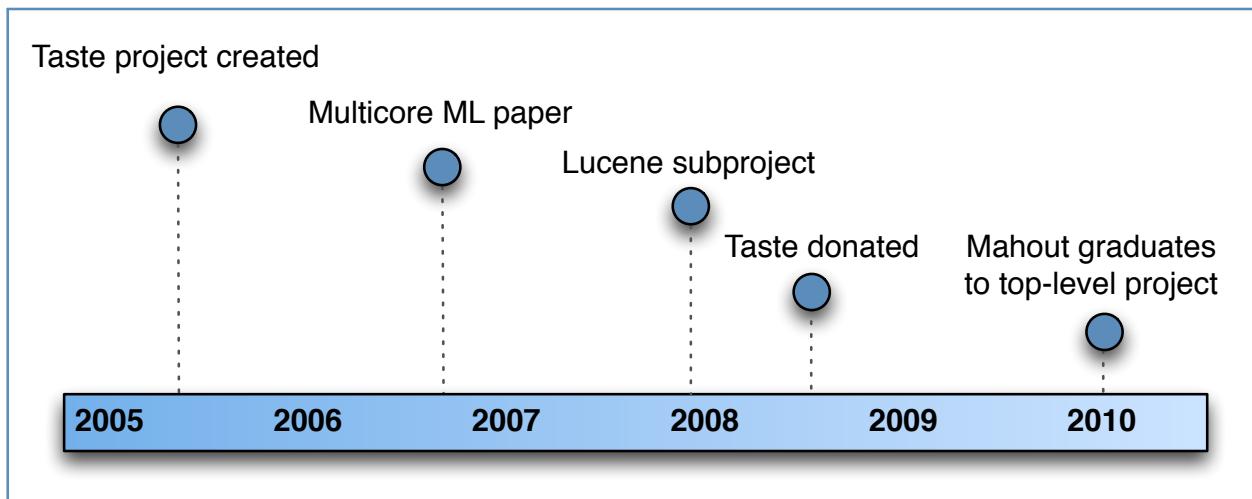
# Chapter Topics

## Introduction to Apache Mahout

- What Apache Mahout is (and is not)
- **Brief history**
- Availability and installation
- Review questions
- Optional Demo: Using Mahout's Item-Based Recommender
- Essential points
- Conclusion

# Brief History of Apache Mahout

- Sean Owen started the Taste CF project in 2005
- Separately, Lucene developers were inspired by a 2006 paper
  - *Map-Reduce for Machine Learning on Multicore*
- Mahout created as a sub-project of Lucene in January, 2008
  - Taste and Mahout “merged” in April, 2008
  - Mahout became a top-level Apache project in 2010



# Chapter Topics

## Introduction to Apache Mahout

- What Apache Mahout is (and is not)
- Brief history
- **Availability and installation**
- Review questions
- Optional Demo: Using Mahout's Item-Based Recommender
- Essential points
- Conclusion

# Prerequisites

---

- **Like Hadoop itself, Mahout minimally requires two things**
  - A Unix-like operating system (Linux is most common)
  - Java virtual machine, version 1.6
- **You almost certainly want to use Hadoop as well**
  - Strictly speaking, it's not a requirement
  - Many algorithms can run in parallel with MapReduce
  - Version of Hadoop needed varies by Mahout version

## Availability

---

- **Mahout is available via its Web site: <http://mahout.apache.org/>**
  - Need Maven and Ant to build from source
- **It's also part of Cloudera's Distribution including Apache Hadoop (CDH)**
  - CDH is both free and open source
  - We fix production issues and contribute our patches to Apache

# Installation

---

- To install the binary version downloaded from Apache's site
  - Verify checksum
  - Unpack archive
  - Move newly-created directory to desired location
  - Add its `bin` subdirectory to `PATH` environment variable
  - Make sure Hadoop's `bin` subdirectory is also in your `$PATH`
- Make sure the `JAVA_HOME` environment variable is set
- Your virtual machine already contains Mahout as part of CDH
  - Installation took only one command

```
$ sudo yum install mahout
```

## Local Mode

---

- **Mahout will attempt to distribute its workload with Hadoop, if possible**
- **For this to work, two things must be true**
  - Hadoop must be installed and properly configured
  - Hadoop's bin subdirectory must be in your \$PATH
- **If Hadoop is not available, Mahout will run in local mode**
- **You can also use local mode even when Hadoop is available**
  - Set the MAHOUT\_LOCAL environment variable to any value

```
$ export MAHOUT_LOCAL=true  
$ mahout -myoptions ... # run Mahout as desired
```

# Chapter Topics

## Introduction to Apache Mahout

- What Apache Mahout is (and is not)
- Brief history
- Availability and installation
- **Review questions**
- Optional Demo: Using Mahout's Item-Based Recommender
- Essential points
- Conclusion

## Review Questions

---

- **Which of The Three C's of Machine Learning does Mahout support?**
- **What does Mahout's early history have in common with Hadoop's?**

# Chapter Topics

## Introduction to Apache Mahout

- What Apache Mahout is (and is not)
- Brief history
- Availability and installation
- Review questions
- **Optional Demo: Using Mahout's Item-Based Recommender**
- Essential points
- Conclusion

## Optional Demo: Overview

- Input is a comma-delimited list of user IDs, movie IDs and ratings
- How to run item-based recommender from command-line

```
$ mahout recommenditembased \
  --input /clouderamovies/demoratings.csv \
  --tempDir /tmp/mahoutdemo \
  --similarityClassname SIMILARITY_EUCLIDEAN_DISTANCE \
  --output /clouderamovies/demoresults
```

- Outputs a list of movies Mahout recommends for each user
  - Format: user\_id [predicted\_rating:movieid, ...]
- Time permitting, your instructor will now demonstrate this

# Chapter Topics

## Introduction to Apache Mahout

- What Apache Mahout is (and is not)
- Brief history
- Availability and installation
- Review questions
- Optional Demo: Using Mahout's Item-Based Recommender
- **Essential points**
- Conclusion

## Essential Points

---

- **Apache Mahout is an open source machine learning library**
  - It's really a collection of various implementations and utilities
- **Mahout can use Hadoop for better performance and scalability**
  - Though not all of Mahout's algorithms currently support this
- **Mahout supports collaborative filtering through Taste**
- **Using Mahout doesn't necessarily require writing any code**

# Chapter Topics

## Introduction to Apache Mahout

- What Apache Mahout is (and is not)
- Brief history
- Availability and installation
- Review questions
- Optional Demo: Using Mahout's Item-Based Recommender
- Essential points
- **Conclusion**

# Introduction to Apache Mahout

---

## In this chapter you have learned

- **What Apache Mahout is**
- **How it was developed**
- **What's required to run Apache Mahout**
- **How you can use Mahout for Collaborative Filtering**

# Bibliography

---

**The following offer more information on topics discussed in this chapter**

- **The Apache Mahout Web site**

- <http://tiny.cloudera.com/dsccl1a>

- **Mahout in Action**

- <http://tiny.cloudera.com/dsccl1b>

# Implementing Recommenders with Apache Mahout

---

Chapter 12



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- **Implementing Recommenders with Apache Mahout**
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Implementing Recommenders with Apache Mahout

---

## In this chapter you will learn

- How several common similarity metrics work
- How data type, magnitude, and rating bias can influence your choice of a similarity metric
- What scoring is in the context of recommender systems
- How modifying your scoring factors can make your recommender more aligned with business interests

# Chapter Topics

## Implementing Recommenders with Apache Mahout

- **Overview**
- Similarity metrics for binary preferences
- Similarity metrics for numeric preferences
- Scoring
- Review questions
- Hands-On Exercise: Comparison of Similarity Metrics
- Essential points
- Conclusion

# Generating Recommendations with Mahout

- Two main ways to generate recommendations with Mahout
  - Writing Java code that invokes Mahout's APIs
  - Using the command line recommender
- We're going to focus on the latter approach
- The `mahout` command has an item-based recommender
  - The following is a basic example of how to invoke it
  - If Mahout is part of a Hadoop cluster, all file paths are in HDFS

```
$ mahout recommenditembased \
  --input /clouderamovies/ratings.csv \
  --similarityClassname SIMILARITY_EUCLIDEAN_DISTANCE \
  --output /clouderamovies/myrecs
```

# Mahout's Command-Line Recommender Input

---

- The input is a comma-delimited file of preference data
  - User ID, Item ID, and (optionally) that user's rating of the item
  - Binary preferences will not have an associated rating value

```
$ head -n5 ratings.csv
1,168,5
1,172,3
1,165,1
1,156,4
1,196,2
```

# Mahout's Command-Line Recommender Output

- **The result of your Mahout job will be a text file**
  - As with input, it will generally be in HDFS
- **Each line in this file represents ratings for a given user**
  - First field is the user ID
  - Second field is an ordered list of recommended items
    - Each element is a tuple of item ID and predicted rating

```
$ hadoop fs -getmerge /clouderamovies/myrecs results.txt

$ head -n3 results.txt
1 [399:5.0,251:4.8,159:4.3,217:4.2,356:4.1]
2 [387:4.9,332:4.7,249:4.6,282:4.5,297:4.3]
3 [345:5.0,236:4.9,253:4.7,394:4.4,427:4.3]
```

# Limiting Results

- **Collaborative filtering can involve processing a lot of data**
  - Consequently, Mahout jobs can take a long time to finish
- **What if you only want to consider certain users or items?**
  - You could edit the input data to exclude everything else
- **A better option is to use Mahout's filtering support**
  - Can limit by user, by item, or both
  - File is simply a list of IDs, one per line

```
$ mahout recommenditembased \
  --input /clouderamovies/ratings.csv \
  --usersFile /clouderamovies/userlist.txt \
  --itemsFile /clouderamovies/itemlist.txt \
  --similarityClassname SIMILARITY_EUCLIDEAN_DISTANCE \
  --output /clouderamovies/myrecs
```

# Similarity Metrics on the Command Line

- Mahout supports a number of similarity metrics
  - Which one you should select depends on your data
  - An important choice that affects accuracy and performance
- Which of these to use is specified on the command line
  - Run the mahout recommenditembased command to see a list
  - We'll now discuss several of the most common ones

```
$ mahout recommenditembased \
  --input /clouderamovies/ratings.csv \
  --usersFile /clouderamovies/userlist.txt \
  --itemsFile /clouderamovies/itemlist.txt \
  --similarityClassname SIMILARITY_EUCLIDEAN_DISTANCE \
  --output /clouderamovies/myrecs
```

# Chapter Topics

## Implementing Recommenders with Apache Mahout

- Overview
- **Similarity metrics for binary preferences**
- Similarity metrics for numeric preferences
- Scoring
- Review questions
- Hands-On Exercise: Comparison of Similarity Metrics
- Essential points
- Conclusion

# Binary Preference Data

---

- **The simplest type of preference data is binary**
  - There are no ratings at all
- **Implicit feedback often results in binary preference data**
  - Whether or not a customer bought a product
  - Whether or not someone watched a movie
  - Whether or not a user has a given connection in a social network
  - Whether or not a customer shops at a particular store

# Representing Binary Preference Data

- The input passed to Mahout is a CSV file of user and item IDs

```
$ head -n5 binaryprefs.csv  
1,217  
1,318  
1,262  
2,347  
2,294
```

# Tanimoto Coefficient

- **The Tanimoto coefficient is based on intersection of two sets**
  - It measures the proportion of shared elements to total elements
  - A value of 0.0 indicates that **no** elements are shared
  - A value of 1.0 indicates that **all** elements are shared
- **Example of how to run Mahout with Tanimoto similarity**
  - Always use the booleanData flag with binary preferences

```
$ mahout recommenditembased \
  --input /clouderamovies/binaryprefs.csv \
  --similarityClassname SIMILARITY_TANIMOTO_COEFFICIENT \
  --booleanData \
  --output /clouderamovies/myrecs
```

## Limitations of Tanimoto Coefficient

---

- **Tanimoto attempts to measure what users have in common**
- **For example, Alice and Bob have each seen five movies**
  - Alice has seen three that Bob hasn't
  - Bob has seen three that Alice hasn't
  - They have two movies in common
- **It seems Alice and Bob have slightly similar taste in movies**

## Limitations of Tanimoto Coefficient (cont'd)

---

- Let's look at Alice and Bob's movies more closely

Alice Only	Shared	Bob Only
Beaches	Forrest Gump	The Texas Chainsaw Massacre
Ghost	Back to the Future	A Nightmare on Elm Street
Pretty Woman		Saw

- Do Alice and Bob really have much in common?
  - It's more likely that this overlap is coincidental
- How can we improve our ratings in situations like this?

## Log Likelihood

---

- **The log likelihood similarity metric handles this situation**
  - It takes the statistical likelihood of coincidence into account
- **Log likelihood analyzes four key values**

	How many watched X?	How many did not watch X?
How many watched Y?	# who watched both X and Y	# who watched Y but not X
How many did not watch Y?	# who watched X but not Y	# who watched neither X nor Y

## Log Likelihood (cont'd)

- Alice and Bob will have low similarity with log likelihood
  - The fact that they saw the same two movies is inconsequential
  - Those are popular movies – everyone else saw them too

	Alice	Bob	Chuck	Donna	Eddie
Beaches	X				
Ghost	X			X	
Pretty Woman	X			X	
Forrest Gump	X	X	X	X	X
Back to the Future	X	X	X	X	X
A Nightmare on Elm Street		X			X
Texas Chainsaw Massacre		X	X		
Saw		X			X

## Log Likelihood (cont'd)

- On the other hand, Alice and Donna will have a high score
  - Their similarities are less common and more meaningful

	Alice	Bob	Chuck	Donna	Eddie
Beaches	X				
Ghost	X			X	
Pretty Woman	X			X	
Forrest Gump	X	X	X	X	X
Back to the Future	X	X	X	X	X
A Nightmare on Elm Street		X			X
Texas Chainsaw Massacre		X	X		
Saw		X			X

## Log Likelihood (cont'd)

- Log likelihood is *usually* more accurate than Tanimoto, given the same data
  - Especially when there's a lot of input data to analyze
- Example of how to run Mahout with log likelihood similarity

```
$ mahout recommenditembased \
  --input /clouderamovies/binaryprefs.csv \
  --similarityclassname SIMILARITY_LOGLIKELIHOOD \
  --booleanData \
  --output /clouderamovies/myrecs
```

# Chapter Topics

## Implementing Recommenders with Apache Mahout

- Overview
- Similarity metrics for binary preferences
- **Similarity metrics for numeric preferences**
- Scoring
- Review questions
- Hands-On Exercise: Comparison of Similarity Metrics
- Essential points
- Conclusion

# Numeric Preference Data

---

- **Numeric preferences convey signal strength**
  - How *much* did the user like (or dislike) an item?
- **This is typically based on explicit feedback**
  - How did the customer rate the movie on a scale of 1 to 5?
- **It can also be based on implicit feedback**
  - How many times did the customer watch a given movie?
  - How much did customer spend on related merchandise?

# Representing Numeric Preference Data

- The input is a comma-delimited file of preference data
  - User ID, item ID, and that user's rating of the item

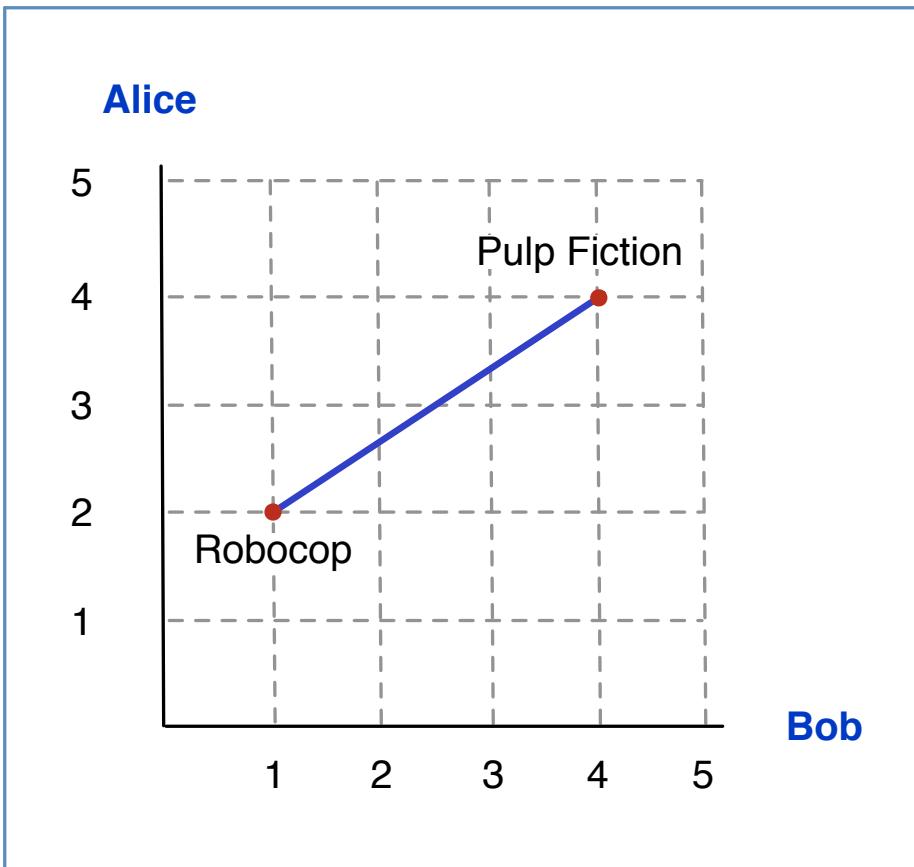
```
$ head -n3 example01.csv  
1,168,5  
1,172,3  
1,165,1
```

- The format and scale of these rating values may vary
  - Ratings may be formatted as integer or decimal values
  - Need not be limited to the range of 1-5

```
$ head -n3 example02.csv  
1,168,55.36  
1,172,37.41  
1,165,19.38
```

# Euclidean Distance

- Euclidean distance is a simple metric for numeric data
  - The same as you'd measure with a ruler and chart



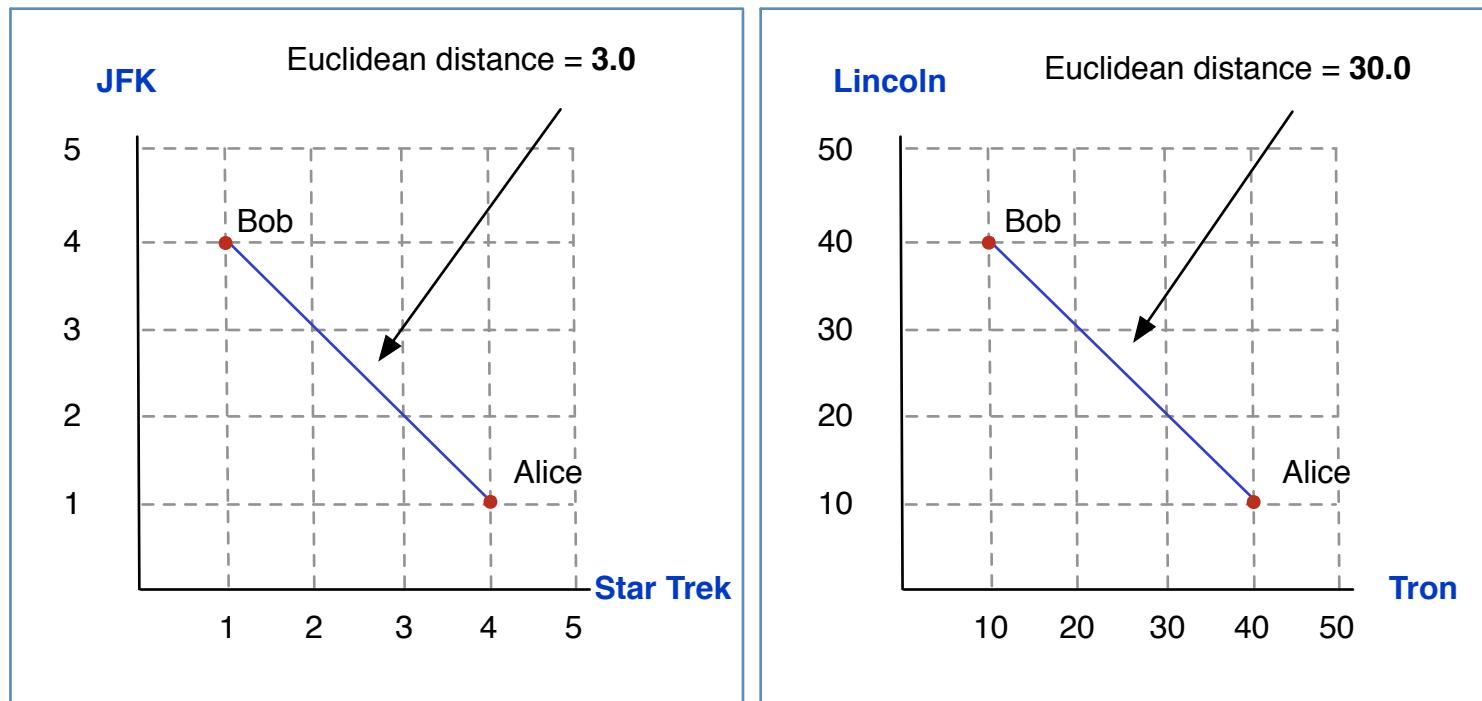
## Euclidean Distance (cont'd)

- Example of how to run Mahout with Euclidean distance similarity

```
$ mahout recommenditembased \
  --input /clouderamovies/ratings.csv \
  --similarityClassname SIMILARITY_EUCLIDEAN_DISTANCE \
  --output /clouderamovies/myrecs
```

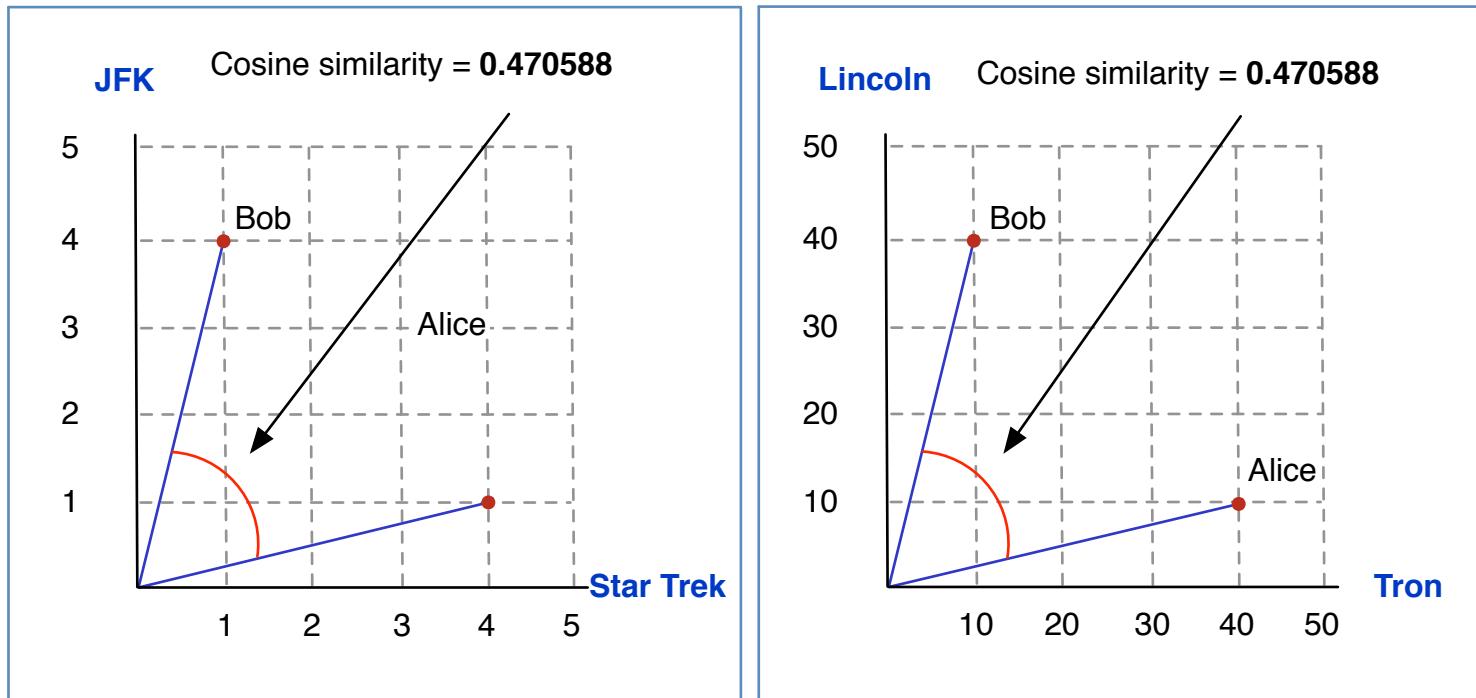
# The Importance of Magnitude

- Let's imagine that we're going to track the number of times each customer has watched a movie with a given actor
- Note that the scale is different in the second plot
  - Values are proportional, but Euclidean distance differs greatly



# Cosine-Based Similarity Metrics

- Cosine-based metrics are based on angles rather than distance
  - The points being measured form a triangle with the origin
  - This approach discounts magnitude when determining similarity
  - We'll discuss two types of cosine-based metrics



# Cosine Similarity

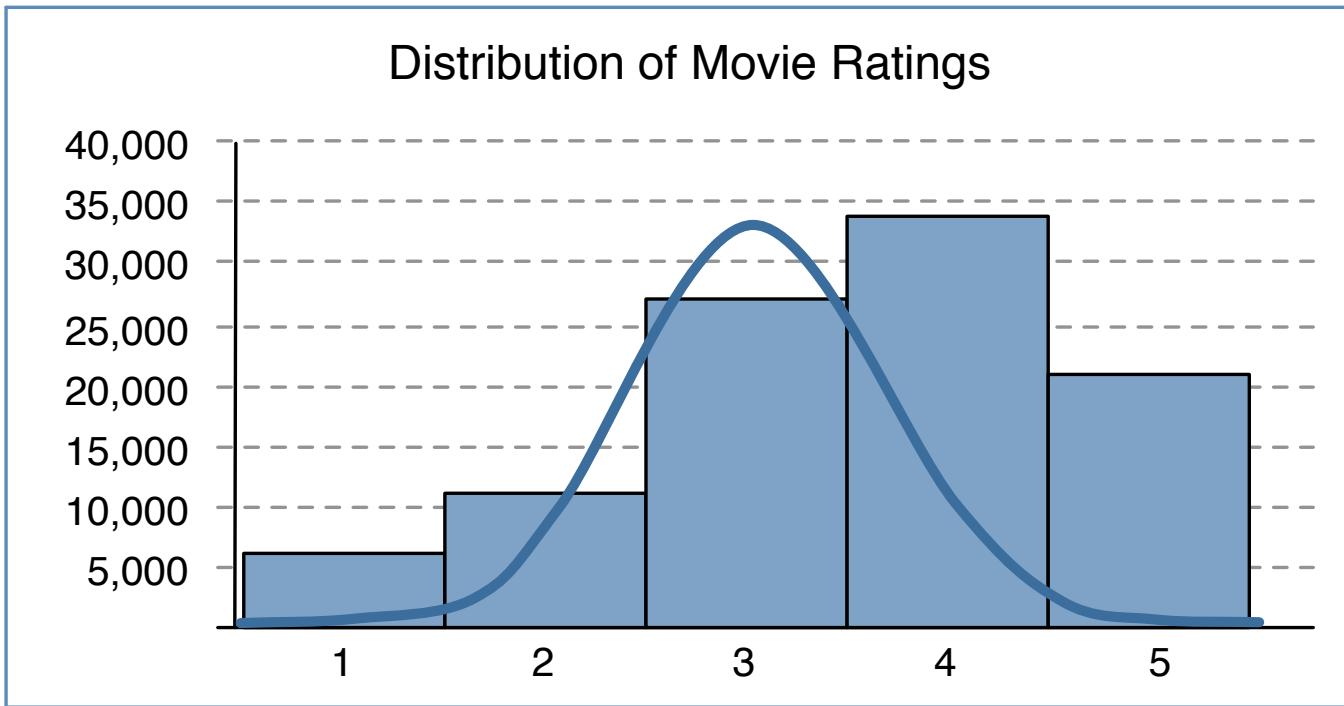
---

- The more basic of these is called cosine similarity in Mahout
  - But would more accurately be called “uncentered cosine”
  - Points are always triangulated to the origin, as we just saw
- Example of how to run Mahout with cosine similarity

```
$ mahout recommenditembased \
  --input /clouderamovies/ratings.csv \
  --similarityClassname SIMILARITY_COSINE \
  --output /clouderamovies/myrecs
```

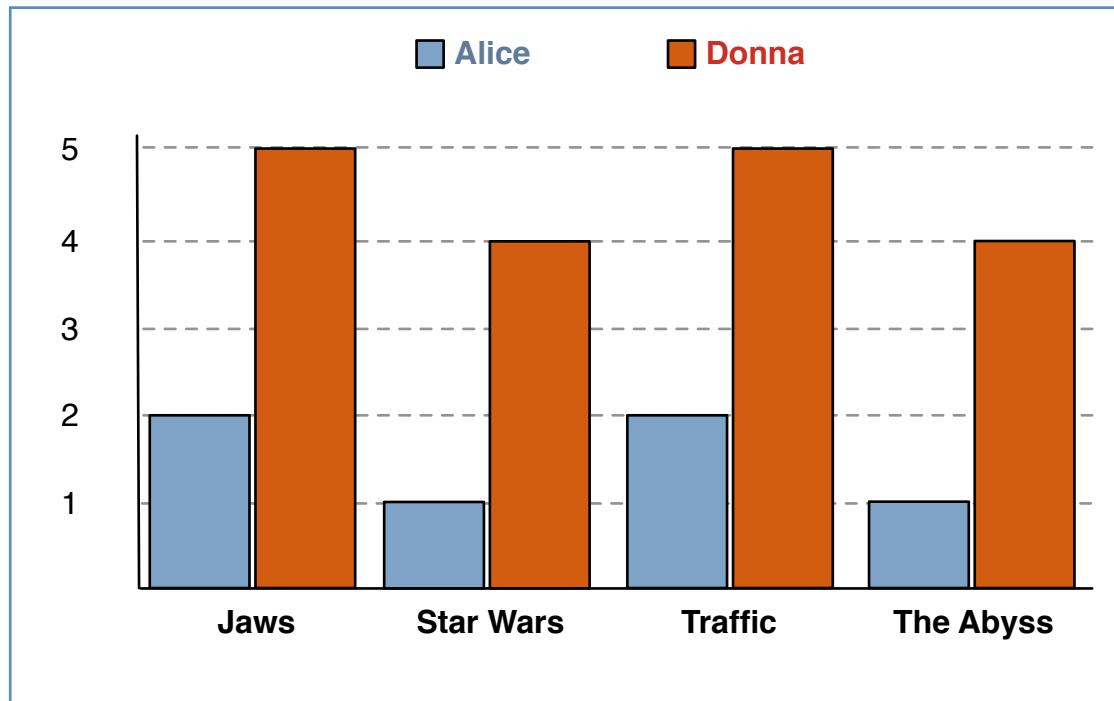
# Distribution of Ratings

- As we previously discussed, ratings are not evenly distributed



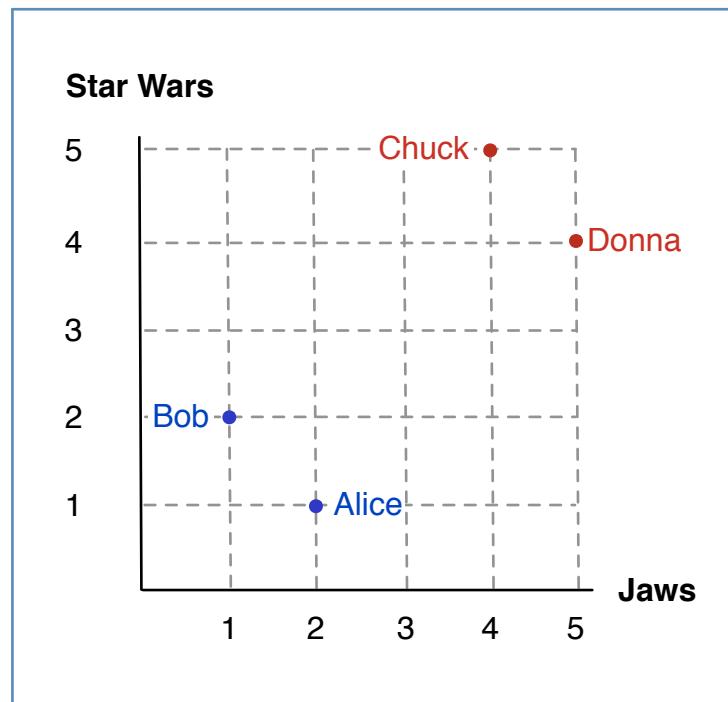
# Bias in Ratings

- Ratings are also not consistent across users
  - Alice says, “Jaws was a good movie so I will give it a 2”
    - She also says, “I didn’t like Star Wars, so I’ll give it a 1”
  - Donna assigns 5 to movies she likes and 4 to ones she doesn’t



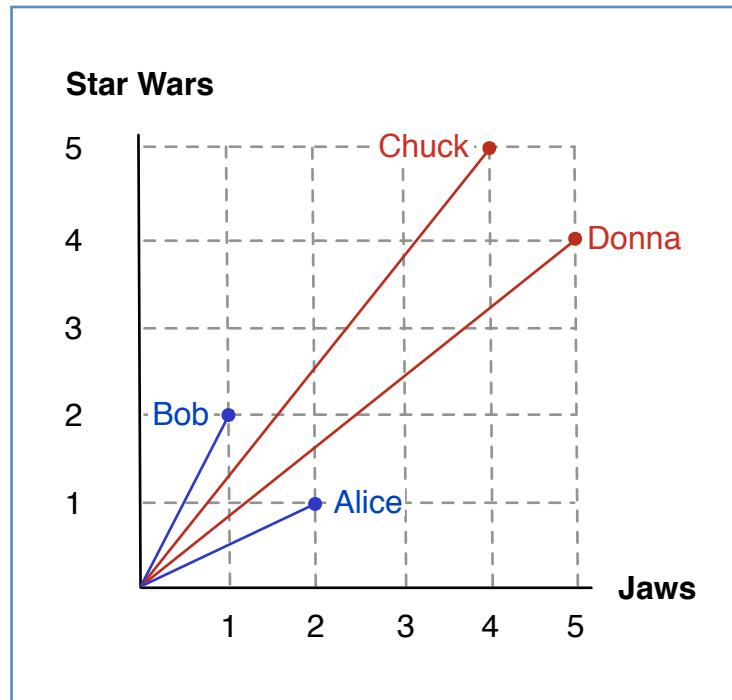
## Bias in Ratings (cont'd)

- Bias can cause misleading comparisons
- We see that Bob and Alice (who tend to rate low) are similar
  - But we might miss the similarity between Alice and Donna



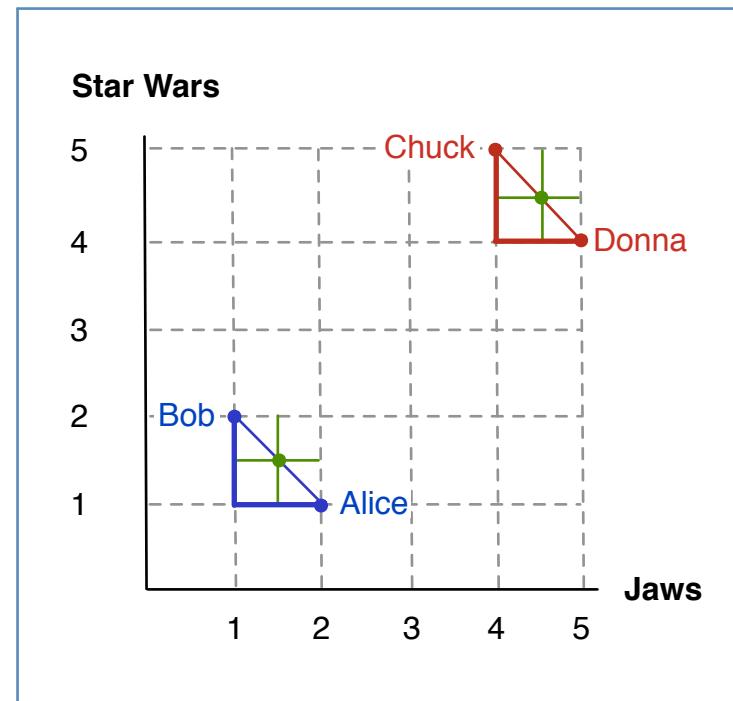
# Bias and Uncentered Cosine Similarity

- **Uncentered cosine similarity does not handle bias well**
  - Distance from the origin affects the angle
  - This makes accurate comparisons difficult



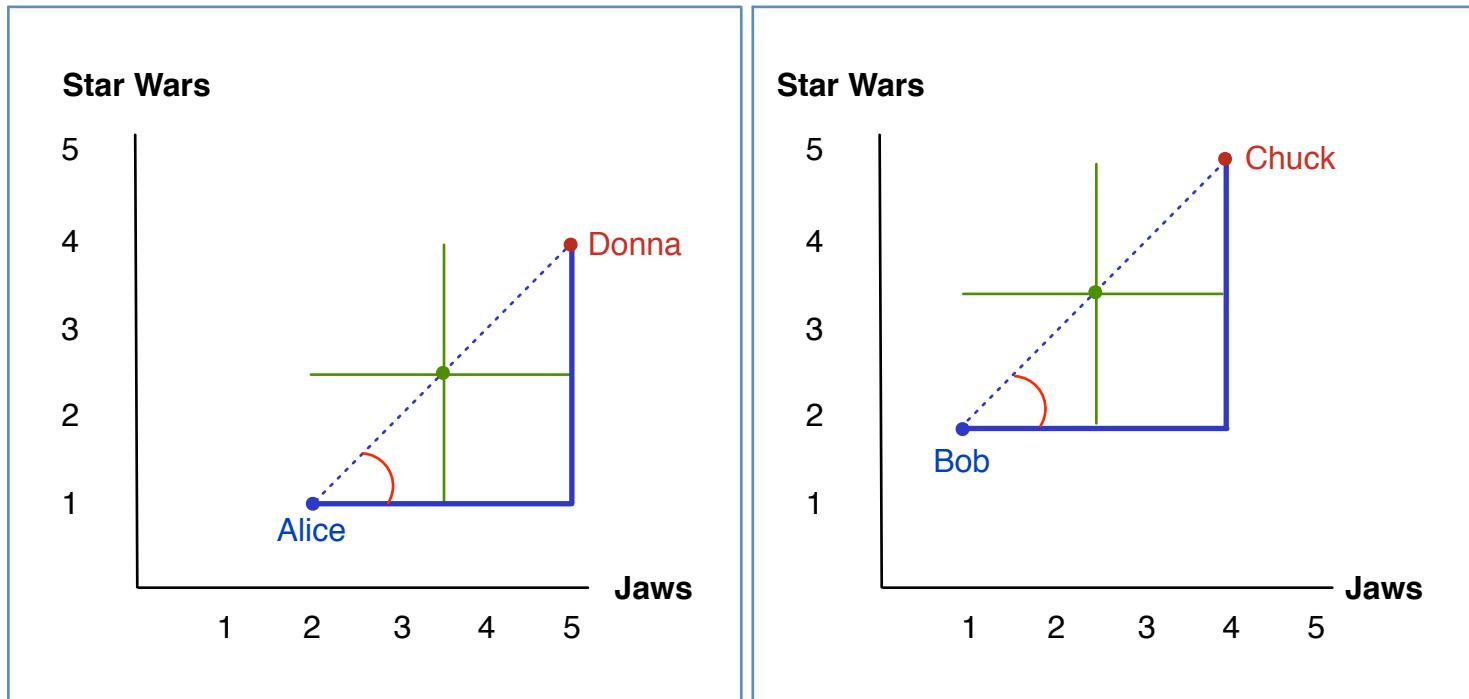
# Overcoming Bias in Ratings

- A centered cosine similarity metric can overcome this problem
- Instead of triangulating to the origin
  - Calculate the mean of the values being compared
  - Plot this mean value and then base the triangulation around it



## Overcoming Bias in Ratings (cont'd)

- This allows more accurate comparisons despite bias
  - We now see that Alice and Donna are similar
  - It's also clear that Bob and Chuck are similar too



# Pearson Correlation

- Another cosine-based similarity metric supported by Mahout
  - Pearson correlation centers about the mean to overcome bias
- Pearson correlation isn't ideal when many ratings overlap
  - It does not consider number of shared items
  - There's no standard deviation when both ratings are equal
- Example of how to run Mahout with Pearson correlation similarity

```
$ mahout recommenditembased \
  --input /clouderamovies/ratings.csv \
  --similarityClassname SIMILARITY_PEARSON_CORRELATION \
  --output /clouderamovies/myrecs
```

# Advice on Choosing a Similarity Metric

---

- **Which similarity metric should you choose?**
  - It largely depends on your preference data
- **If you have binary data, log likelihood is typically the best choice**
- **If you have numeric data...**
  - And magnitude is important, consider Euclidean distance
  - And you need to overcome ratings bias, try Pearson correlation
  - And you have many overlapping values, use uncentered cosine
- **It's usually best to experiment and compare several such metrics**
  - We'll do exactly this during the hands-on exercise

# Chapter Topics

## Implementing Recommenders with Apache Mahout

- Overview
- Similarity metrics for binary preferences
- Similarity metrics for numeric preferences
- **Scoring**
- Review questions
- Hands-On Exercise: Comparison of Similarity Metrics
- Essential points
- Conclusion

# What is Scoring?

---

- **We have thus far focused on a single piece of explicit feedback**
  - How a given user rated a given movie
- **However, the term *rating* is not synonymous with *input***
  - Recommenders often combine many types of feedback
- **Different types of feedback are signals of preference**
  - For example, a user may rate a movie after watching it
  - She might instead post a message to a social media site about it
  - She might both rate the movie and post a message about it
- **Scoring refers to the value we place on these signals**
  - It's how important we consider them relative to each other

# Considerations for Scoring

---

- **Scoring is an essential part of the value a data scientist provides**
  - It's often more important than algorithm or similarity metric
- **Scoring is an ongoing consideration for recommender systems**
  - It's the result of constant experimentation
  - Must react to changing conditions and new opportunities
- **It almost always requires knowledge of the domain**
  - We must know what's relevant to the business *and* the customer
- **Scoring effectively can have a profound effect on the bottom line**
  - It's often a balance between relevance and profits
    - Users want relevant suggestions
    - Business wants higher revenue and lower costs

# Examples of Potential Scoring Criteria

---

- **Scoring often makes extensive use of implicit feedback**
- **It may also consider revenue, costs, and other business factors**
  - Selling price
  - Wholesale cost
  - Amount of item currently in inventory
- **Cloudera Movies might score based on the following criteria**

% of score	Description of Criteria
42%	How customer rated similar movies
28%	Cost of royalty payment required to show this movie
17%	Whether the user searched for this movie by name
8%	How long this movie has been in the customer's queue
5%	Popularity of this movie among others in customer's region

# Integrating Scores into Mahout

- Recall the format of the input we provide to Mahout

```
$ head -n5 ratings.csv  
1,168,5  
1,172,3  
1,165,1  
1,156,4  
1,196,2
```

- How do we incorporate scoring into this data?
  - Simply write code that incorporates all your scoring logic
  - Your program will create output similar to what's seen above
  - Your program's output will be the input to Mahout

# Chapter Topics

## Implementing Recommenders with Apache Mahout

- Overview
- Similarity metrics for binary preferences
- Similarity metrics for numeric preferences
- Scoring
- **Review questions**
- Hands-On Exercise: Comparison of Similarity Metrics
- Essential points
- Conclusion

## Review Questions

---

- **Which metrics would you consider for numeric preferences?**
  - Which of those we discussed accounts for ratings bias?
- **What scoring criteria do you think would be important for a recommender used by a rental car agency to suggest vehicles? How about a recommender used by an online jeweler?**

# Chapter Topics

## Implementing Recommenders with Apache Mahout

- Overview
- Similarity metrics for binary preferences
- Similarity metrics for numeric preferences
- Scoring
- Review questions
- **Hands-On Exercise: Comparison of Similarity Metrics**
- Essential points
- Conclusion

## Hands-on Exercise: Similarity Metrics

---

- In this Hands-On Exercise, you will experiment with different similarity metrics and observing the results they produce
- Please refer to the Hands-On Exercise Manual for instructions on exercise #6

# Chapter Topics

## Implementing Recommenders with Apache Mahout

- Overview
- Similarity metrics for binary preferences
- Similarity metrics for numeric preferences
- Scoring
- Review questions
- Hands-On Exercise: Comparison of Similarity Metrics
- **Essential points**
- Conclusion

## Essential Points

---

- Mahout supports several similarity metrics
- Each similarity metric has limitations. The values of your preference data generally dictate which metric is the best choice.
- We indicate the relative importance of various criteria of our recommender input through scoring
- Scoring is an important function which requires domain knowledge and can yield significant business value

# Chapter Topics

## Implementing Recommenders with Apache Mahout

- Overview
- Similarity metrics for binary preferences
- Similarity metrics for numeric preferences
- Scoring
- Review questions
- Hands-On Exercise: Comparison of Similarity Metrics
- Essential points
- Conclusion

# Implementing Recommenders with Mahout

---

## In this chapter you have learned

- How several common similarity metrics work
- How data type, magnitude, and rating bias can influence your choice of a similarity metric
- What scoring is in the context of recommender systems
- How modifying your scoring factors can make your recommender more aligned with business interests

# Experimentation and Evaluation

---

Chapter 13



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- **Experimentation and Evaluation**
- Production Deployment and Beyond
- Conclusion

# Experimentation and Evaluation

---

## In this chapter you will learn

- How testing can lead to iterative improvements that continually benefit the organizations that conduct them
- Why user interface design is an important component of building and deploying a recommender system
- How to determine if your recommender is effective
- What considerations are involved in experiment design

# Chapter Topics

## Experimentation and Evaluation

- **Measuring recommender effectiveness**
- Designing effective experiments
- Conducting an effective experiment
- User interfaces for recommenders
- Review questions
- Hands-On Exercise: Improving Recommender Accuracy
- Essential points
- Conclusion

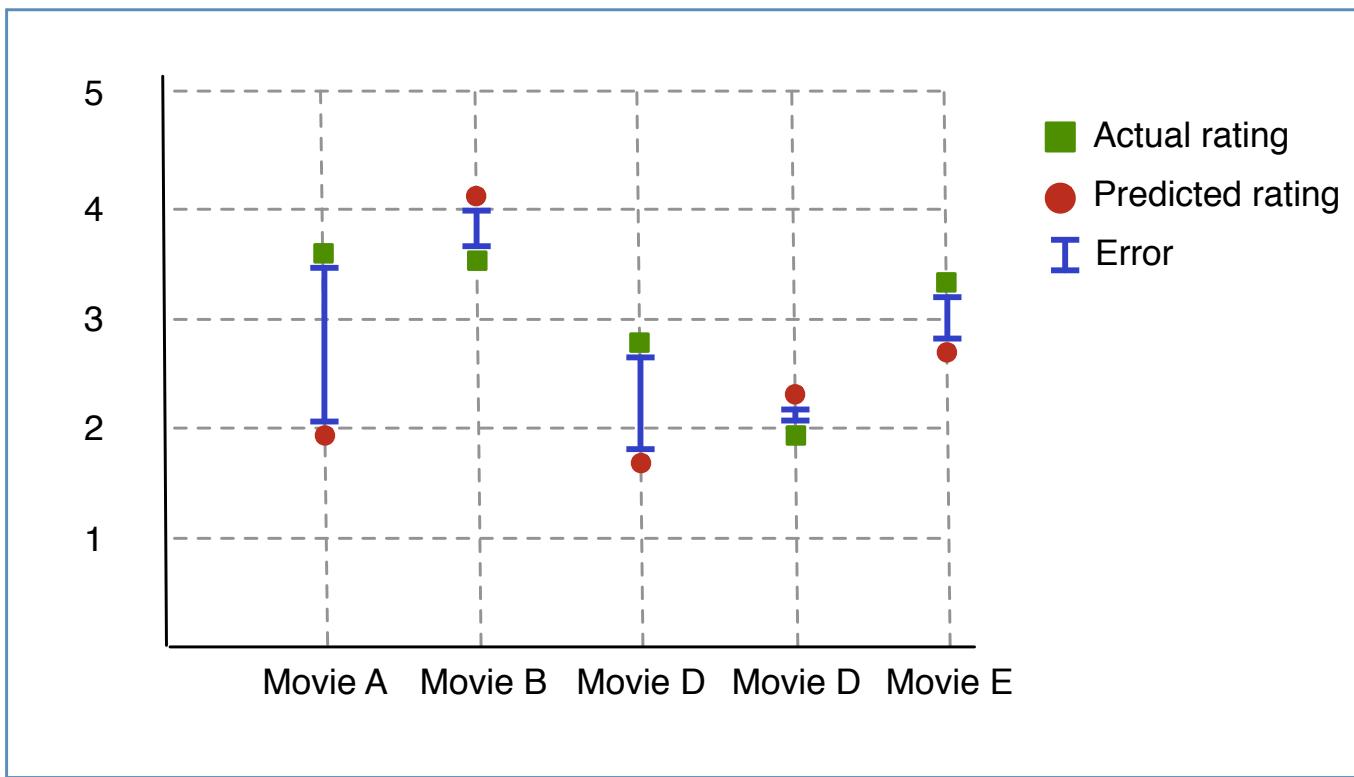
# Defining Recommender Effectiveness

---

- **You now know how to build a recommender system**
  - But how do you know if the recommendations are effective?
- **To answer this, you must first define “effective”**
  - Subjective terms can be difficult to measure
  - We must first determine the *accuracy* of our recommendations
  - We will then assess their *quality*

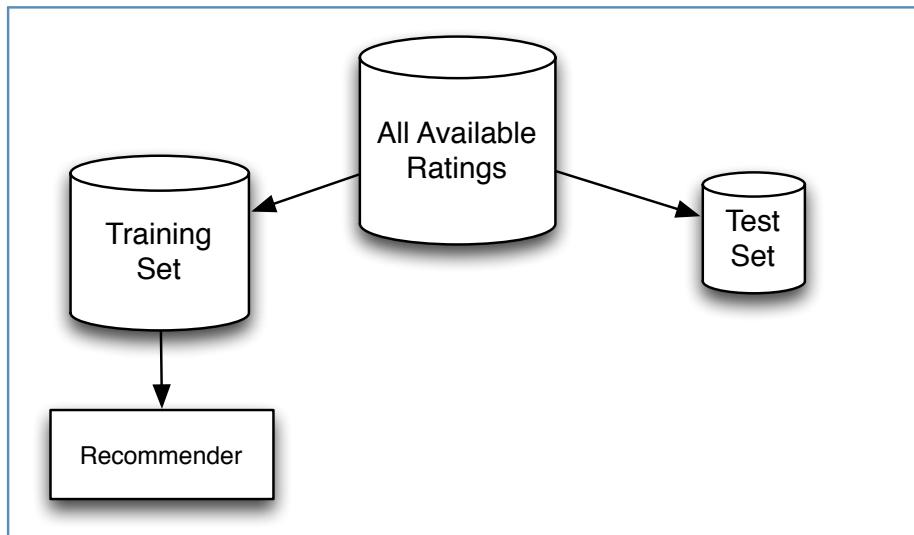
## Calculating the Error

- One way to measure accuracy is to compare our predicted ratings to the actual ratings given by the customer
  - The difference between these two values is the error



# Training Sets and Test Sets

- To calculate the error, we need actual and predicted ratings
  - How do we get a user's ratings for items he or she hasn't seen?
- We need to simulate this using a training set
  - Divide your recommender input (randomly) into two sets
    - The **training set** is used to generate recommendations
    - The **test set** is withheld from the recommender



## Prediction and Comparison

---

- We now ask our recommender to predict the rating for a item that we know exists only in the test set
  - We then simply compare this rating to the actual rating found in the test set to determine the error amount
- This process is repeated for each item in the test set

# RMSE

---

- RMSE is a common evaluation metric based on this difference
  - It stands for Root Mean Squared Error
- The formula for RMSE is simple
  - For each item, find the error between predicted and actual ratings
  - Square each error
  - Calculate the mean (average) of all squared errors
  - The RMSE is the square root of this mean value
- Lower numbers indicate more accurate predictions

	Actual Ratings	Predicted Ratings	RMSE
Case A	[2, 4, 3, 1]	[1, 4, 5, 2]	1.225
Case B	[2, 4, 3, 1]	[5, 2, 3, 4]	2.345

## Limitations of RMSE

---

- RMSE is widely used for evaluating recommender accuracy
  - It was used as the metric for the Netflix prize
- However, it has some important limitations
  - It's not appropriate for binary (boolean) preference data
  - It doesn't consider the order of our recommendations
  - It's sensitive to outliers: a single recommendation that is off by two negatively affects RMSE more than two that are off by one

	Actual Ratings	Predicted Ratings	Sum of Errors	RMSE
Case A	[5, 4]	[3, 4]	2	1.414
Case B	[3, 4]	[2, 3]	2	1.000

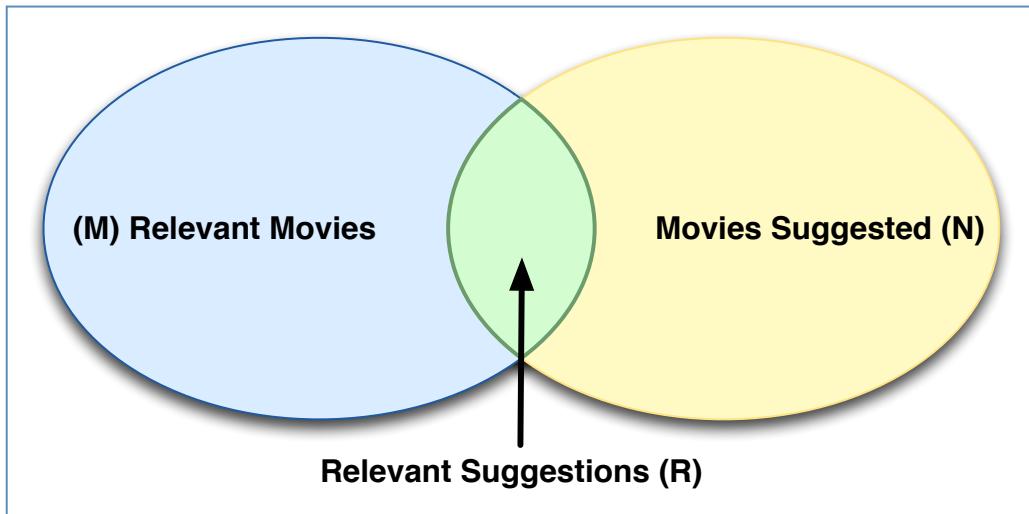
## Cross Validation

---

- **The K-fold cross validation approach can add more rigor**
  - Can mitigate the risk of sampling error
- **Instead of dividing the input data into training and test sets once**
  - Divide the input data into K equal partitions
  - One partition will be the test set while the rest are used for training
    - Train the recommender using the training set
    - Evaluate the recommender using the test set
  - Repeat these steps for each partition
    - Then calculate the mean RMSE value from all tests
- **This ultimately uses *all* data for both training and testing**

# Precision and Recall

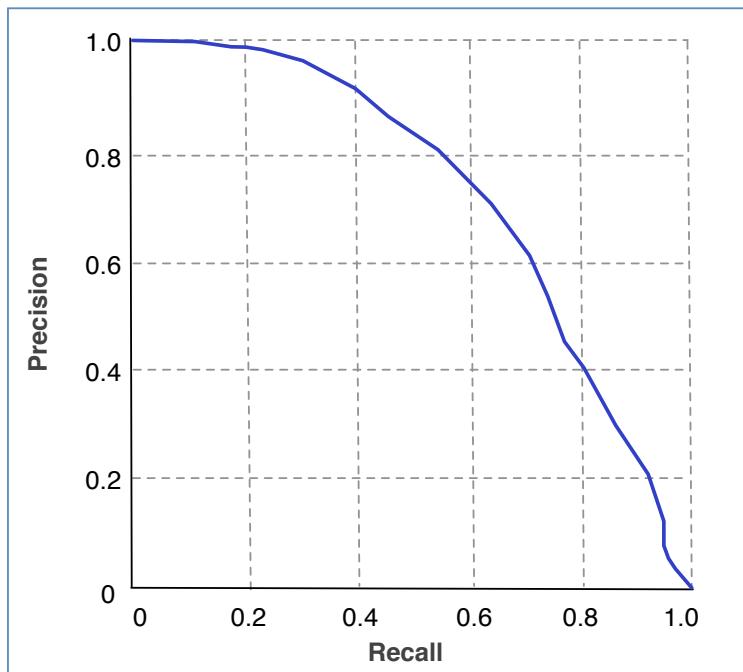
- Now that we've evaluated accuracy, let's assess quality
  - This is effectively a measure of their relevance to the user



- Precision is the ratio of relevant recommendations among all those we offered ( $R / N$ )
- Recall is the ratio of relevant recommendations we made among all relevant recommendations that were possible ( $R / M$ )

# Relationship Between Precision and Recall

- Precision and recall are most often inversely dependent
  - You could optimize for recall by presenting more suggestions
  - However, fewer of those suggestions would be relevant



- Therefore, “precision at N” is usually a more valuable metric

## Limitations of Precision and Recall

---

- **Although precision and recall are important metrics, they also have shortcomings**
- **Relevance and ratings are often at odds with one another**
  - A user considers an item relevant if he or she is familiar with it
  - An unknown item is not relevant, even if it's a good suggestion
- **Using precision and recall isn't scalable**
  - Removing per-user biases makes computation expensive
  - Cannot realistically do this on more than a small subset of users

# Chapter Topics

## Experimentation and Evaluation

- Measuring recommender effectiveness
- **Designing effective experiments**
- Conducting an effective experiment
- User interfaces for recommenders
- Review questions
- Hands-On Exercise: Improving Recommender Accuracy
- Essential points
- Conclusion

# Design of Experiments

---

- An experiment can help you evaluate different solutions and choose which will be most worthwhile for production use
- Overview of major steps in experiment design
  1. Construct a hypothesis
  2. Determine how to manipulate the independent variable
  3. Consider methods to control other variables
  4. Decide what to measure and how to collect results

# Constructing a Hypothesis

---

- **The hypothesis is a statement of what you expect to observe**
- **These are suggested by associations observed in the data**
  - “Customers who watch action movies use our service more”
- **May start out vague, but become precise with further analysis**
  - A hypothesis must also be something you can test
  - “We expect that by suggesting 10% more action movies to all customers, those who accept the recommendations will spend an average of 15% more time using our service during the same period than customers who do not.”

# Null Hypothesis

---

- **A hypothesis is a statement of correlation between two factors**
  - Your experiment attempts to prove or disprove this relationship
- **Possible explanations for behavior observed in an experiment**
  - Your experiment was poorly designed or executed
  - Your hypothesis is correct
  - Random chance
- **The null hypothesis relates to random chance**
  - It asserts that no relationship exists between these factors
  - The null hypothesis is negative – and can never be proven

# Confidence Intervals

---

- **Confidence intervals describe the likelihood of random chance**
  - In other words, the likelihood that the null hypothesis is correct
- **A confidence interval conveys statistical significance**
- **A 95% confidence interval means that 100 identical experiments**
  - Would produce results in the same range 95 times
  - The remaining five fall outside this range due to random chance

# Variable Manipulation and Control

---

- **The experiment must use the independent variable as input**
  - In this case, the number of action movies a customer watches
- **You must also control for other variables that may affect results**
  - Preference for (or against) specific actors
  - Customer's age, income level, location, or gender
  - Day of week and time of day
  - Customer's tendency to accept our recommendations

# Random Assignment and Stratification

---

- **Random assignment to groups helps to control variables**
  - Important to have a *control group* for comparison
  - Each group except the control group receives a treatment
- **For Web sites, the typical random assignment is simple**
  - Assign each user a unique identifier (e.g. in a browser cookie)
  - Convert this ID to an integer using a hash function
  - Calculate this integer (hash value) modulo number of groups
- **Stratification divides population into homogenous groups**
  - This should be done before random assignment
  - Stratification helps ensure proper distribution during assignment

# Population Size and Test Duration

---

- **You need a large enough population of test subjects**
  - So that you decrease the likelihood of sampling error
  - Ensures that observations aren't the result of personal quirks
- **Your test also needs to run over a long enough period**
  - A test that runs for only two hours is sensitive to time
  - A one-day test is sensitive to day of week or month

# Collecting Data

---

- **Recommenders *create data* as well as consume it**
  - What data will your experiment require?
  - What data will your experiment produce?
- **How will you collect it?**
  - Are you capturing everything of value?
  - Are you collecting it accurately?
  - Will your collection mechanism scale?

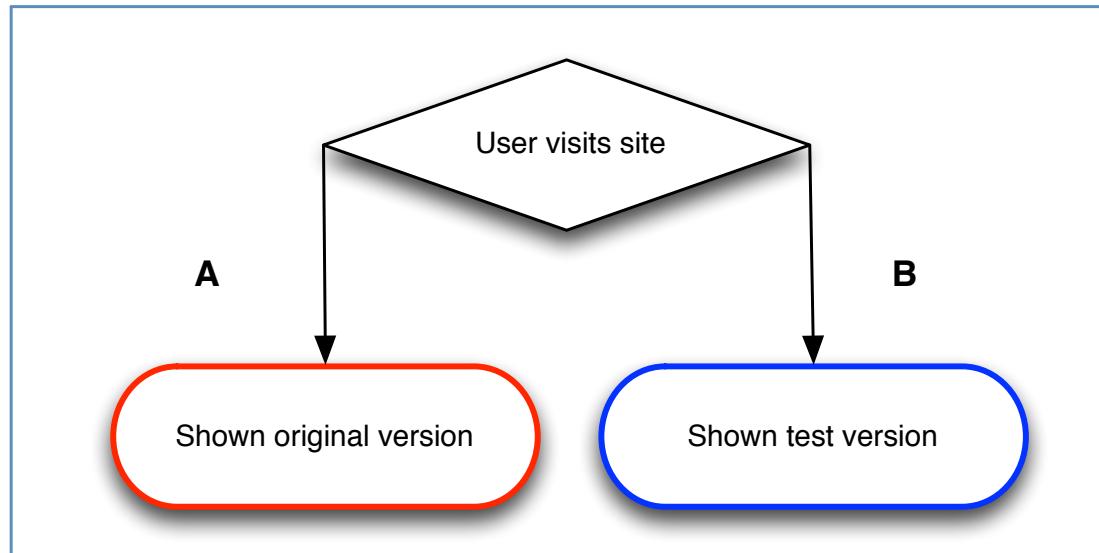
# Chapter Topics

## Experimentation and Evaluation

- Measuring recommender effectiveness
- Designing effective experiments
- **Conducting an effective experiment**
- User interfaces for recommenders
- Review questions
- Hands-On Exercise: Improving Recommender Accuracy
- Essential points
- Conclusion

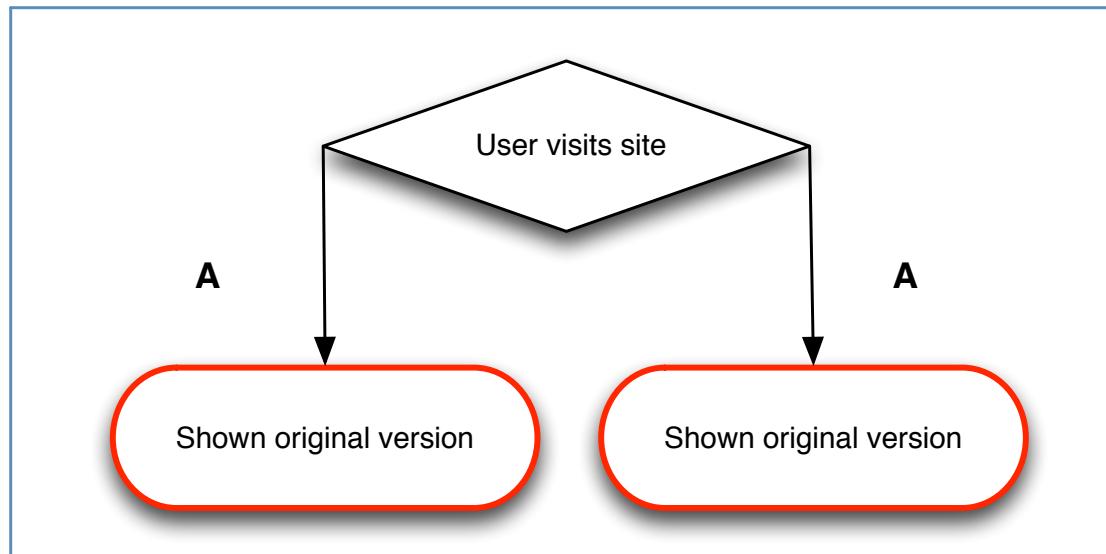
# The A/B Test

- There are many different types of experiments
- We'll focus on one type: the A/B test
  - Divide population into two groups
  - Group A sees the original (unmodified) version
  - Group B sees the test (modified) version



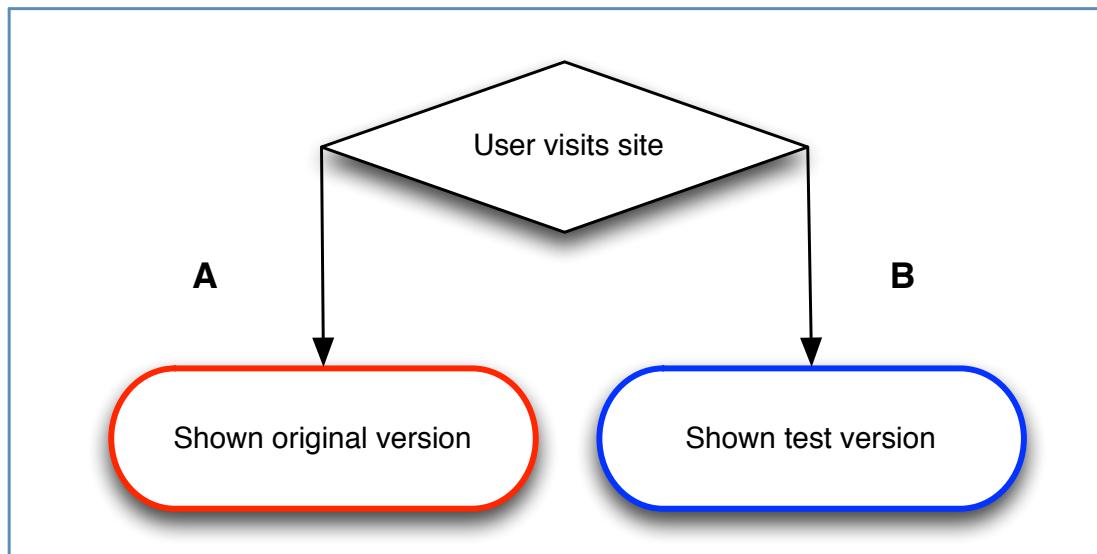
## Run an A/A Test First

- An A/A test still subdivides users into two groups
  - But both groups are shown the original version
- What benefit does an A/A test provide?
  - It verifies that your group assignment is correct
  - If it is, you should observe the same behavior from both groups



# Using A/B Tests for Recommender Systems

- For recommender systems, A and B might be a difference in
  - Similarity metrics
  - Collaborative filtering algorithms
  - Scoring criteria
  - Presentation (user interface)



# Using Test Results for Decision Support

---

- **In many organizations, decisions are made based on HiPPO**
  - Highest Paid Person's Opinion (i.e. what the boss says)
- **Successful organizations listen to the customer instead**
  - Not necessarily what the customer *says*, but what they *do*
  - They run a test and let the data speak for itself
- **Our results ultimately influence a decision on ROI**
  - Does this feature provide enough value to remain in production?
  - Does it warrant further investment?

# Chapter Topics

## Experimentation and Evaluation

- Measuring recommender effectiveness
- Designing effective experiments
- Conducting an effective experiment
- **User interfaces for recommenders**
- Review questions
- Hands-On Exercise: Improving Recommender Accuracy
- Essential points
- Conclusion

# Recommender System User Interfaces

---

- **Recommenders are “back end” systems**
  - You must provide some way of displaying its output to users
- **The user interface for presenting recommendations is important**
  - It must be intuitive for the customer to use
- **Many design factors can significantly influence acceptance rate**
  - Size of product icons
  - Position of recommended items relative to one another
  - Color of various elements on the page
  - Text (font face, size, style)
- **Determining the best design is often done through experiments**

# Presentation Example

# cloudera

## MOVIES

Our top picks for you



High School Musical (2006)  
The school's sports hero and a nerdy new girl surprise everyone by singing and dancing in the school play.

Genre: Musical  
Starring: Vanessa Hudgens and Zac Efron  
Directed by: Kenny Ortega

Our prediction: ★★★★☆

Comedies we think you'll enjoy



# Explain Why You're Recommending an Item

- One of these things is not like the other

Our top picks for you



**Land of the Dead (2006)**

Flesh eating zombies have taken over a city, and its trapped residents await their grisly deaths.

Genre:

Horror

Starring:

John Leguizamo, Simon Baker

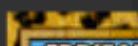
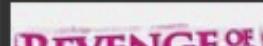
Directed by:

George Romero

Our prediction:



Comedies we think you'll enjoy



Recommended because: You watched Halloween



## Allow Users to Correct Inaccurate Data

- Removing misleading signals improves recommendation quality

## Our top picks for you



## Comedies we think you'll enjoy



**Land of the Dead (2006)**

Flesh eating zombies have taken over a city, and its trapped residents await their grisly deaths.

**Genre:**

**Starring:**

**Directed by:**

## Horror

John Leguizamo, Simon Baker  
George Romero

## Our prediction:



**Recommended because:** You watched Halloween

[ [Don't suggest movies like this in the future](#) ]



# Improving UI Through A/B Tests

---

- **The user interface for soliciting ratings is also important**
  - You'll get more (and better) ratings if it's easy to use
- **The A/B process can help you compare UI improvements**
  - Let's examine one of Cloudera Movies recent improvements...
- **We observed that 90% of customers over the age of 65 assigned a rating of 1 to every movie they watched**

# Improving UI Through A/B Tests (cont'd)

---

- We wanted to investigate what caused this
  - Were we making poor recommendations?
  - Were they just grumpy?
- We decided to conduct a brief online survey of these users
  - This quickly revealed that they had trouble with our ratings UI
- We originally used this simple drop down menu on our Web site



# Improving UI Through A/B Tests (cont'd)

- Our first A/B test showed the new design reduced it to 72%
  - Many users didn't understand what a rating of '1' meant

Previous design

Your rating:



1  
2  
3  
4  
5

New design

Your rating:



1 - Hated it  
2 - Not Good  
3 - OK  
4 - Liked it  
5 - Loved it

# Improving UI Through A/B Tests (cont'd)

- We thought some users hadn't understood what we're asking
  - Our next test showed only negligible improvement
    - We subsequently reverted to the previous label

Previous design	New design
<p>Your rating:</p>  <p>A dropdown menu with the following options: 1 - Hated it 2 - Not Good 3 - OK 4 - Liked it 5 - Loved it</p>	<p>How was the movie?</p>  <p>A dropdown menu with the following options: 1 - Hated it 2 - Not Good 3 - OK 4 - Liked it 5 - Loved it</p>

# Improving UI Through A/B Tests (cont'd)

- Our next test showed a major improvement (reduced to 39%)
  - Many users didn't choose the rating, they just clicked submit
  - The new design requires an explicit selection

Previous design

Your rating:

1 - Hated it

2 - Not Good

3 - OK

4 - Liked it

5 - Loved it

New design

Your rating:

-- Pick one --

1 - Hated it

2 - Not Good

3 - OK

4 - Liked it

5 - Loved it

# Improving UI Through A/B Tests (cont'd)

- Our current design showed a further 21% reduction
  - Many of our older users have poor vision and/or arthritis
  - This design is easier to see and requires less dexterity

Previous design

Your rating:

-- Pick one --

- 1 - Hated it
- 2 - Not Good
- 3 - OK
- 4 - Liked it
- 5 - Loved it

New design

Your rating:

- 
- 
- 
- I liked it
-

# Chapter Topics

## Experimentation and Evaluation

- Measuring recommender effectiveness
- Designing effective experiments
- Conducting an effective experiment
- User interfaces for recommenders
- **Review questions**
- Hands-On Exercise: Improving Recommender Accuracy
- Essential points
- Conclusion

## Review Questions

---

- **What hypothesis can you make based on the data you've seen in your job?**
- **What is one limitation of using the RMSE metric for accuracy?**

# Chapter Topics

## Experimentation and Evaluation

- Measuring recommender effectiveness
- Designing effective experiments
- Conducting an effective experiment
- User interfaces for recommenders
- Review questions
- **Hands-On Exercise: Improving Recommender Accuracy**
- Essential points
- Conclusion

## Hands-on Exercise: Improving Recommender Accuracy

---

- In this Hands-On Exercise, you will gain practice improving the accuracy of the recommender system using the techniques you've just learned
- Please refer to the Hands-On Exercise Manual for instructions on exercise #7

# Chapter Topics

## Experimentation and Evaluation

- Measuring recommender effectiveness
- Designing effective experiments
- Conducting an effective experiment
- User interfaces for recommenders
- Review questions
- Hands-On Exercise: Improving Recommender Accuracy
- **Essential points**
- Conclusion

## Essential Points

---

- An experiment can help you evaluate different solutions and choose which will be most worthwhile for production use
- The A/B test helps us compare the effectiveness of various treatments on a population of users
- Successful organizations value data more than opinions for making decisions
- The user interface is an essential part of a recommender system
- RMSE is a common way of measuring recommender accuracy
- Precision and recall can help us measure recommender quality

# Chapter Topics

## Experimentation and Evaluation

- Measuring recommender effectiveness
- Designing effective experiments
- Conducting an effective experiment
- User interfaces for recommenders
- Review questions
- Hands-On Exercise: Improving Recommender Accuracy
- Essential points
- **Conclusion**

# Experimentation and Evaluation

---

## In this chapter you have learned

- How testing can lead to iterative improvements that continually benefit the organizations that conduct them
- Why user interface design is an important component of building and deploying a recommender system
- How to determine if your recommender is effective
- What considerations are involved in experiment design

# Bibliography

---

The following offer more information on topics discussed in this chapter

- **Web Analytics 2.0**

- <http://tiny.cloudera.com/dscc13a>

- **Google's Overlapping Experiment Infrastructure**

- <http://tiny.cloudera.com/dscc13b>

- **Large Scale Analysis of Interleaved Search Evaluation**

- <http://tiny.cloudera.com/dscc13c>

- **Design of Experiments for Scientists and Engineers**

- <http://tiny.cloudera.com/dscc13d>

# Production Deployment and Beyond

## Chapter 14



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- **Production Deployment and Beyond**
- Conclusion

# Production Deployment and Beyond

---

## In this chapter you will learn

- Which factors may impede recommender accuracy
- How to assess whether further improvements are worthwhile
- Which problems Mahout users often encounter
- Several techniques for improving your recommender system
- What a data scientist's role is in improving system performance

# Chapter Topics

## Production Deployment and Beyond

- **Deploying to production**
- Tips and techniques for working at scale
- Summarizing and visualizing results
- Considerations for improvement
- Next steps for recommenders
- Review questions
- Essential points
- Conclusion

# Taking Recommenders to Production

---

- By “taking recommenders to production” we mean the process of moving from offline analysis of the data to making product recommendations to users on a live production system
- We should view this as a continuation of our experiments

# Successful Deployment

---

- **Initial experimentation answers one key question**
  - Does a change in input lead to a corresponding change in output
- **Successful deployment cycles should answer several more**
  - Does this negatively impact the production system?
  - How can we reduce the cost of testing so we can do more of it?
  - Have we tested the right thing?
- **And most importantly of all**
  - Does this create value for the organization?

# Chapter Topics

## Production Deployment and Beyond

- Deploying to production
- **Tips and techniques for working at scale**
- Summarizing and visualizing results
- Considerations for improvement
- Next steps for recommenders
- Review questions
- Essential points
- Conclusion

# Common Problems with Mahout

---

- **Those who are new to Mahout often stumble with a few issues**
- **Configuration**
  - Ensure that the `JAVA_HOME` environment variable is set
  - The `hadoop` executable must be in your `PATH` variable too
  - Performance settings must be appropriate for your Mahout job
    - More on this in a moment
- **Input data**
  - Ratings data for command-line jobs must be CSV
  - Users and items must be identified by integer values
  - Malformed ratings can skew predictions

## Common Problems with Mahout (cont'd)

---

- **Stray files and directories are also a problem**
  - Apache Mahout doesn't "fail fast" by validating preconditions
- **Here are a few things to check before running your job**
  - Does the output directory exist?
    - Mahout (and Hadoop) won't overwrite existing output
  - Does the temp directory contain files from a previous job?
    - The temp directory must either be empty or not exist
  - Does your input contain a stray file?
    - The only files in the input directory should be ratings data
    - Often results in an `ArrayIndexOutOfBoundsException`

# Ramping Up Your Deployment

- Almost any test has the potential for negative impact
  - The extent of which is initially unknown
- Consider deploying your change to a small population of users
  - This limits the potential damage
- Then ramp up in phases once any bugs are worked out

## Phase 1

A = 99%

B = 1%

## Phase 2

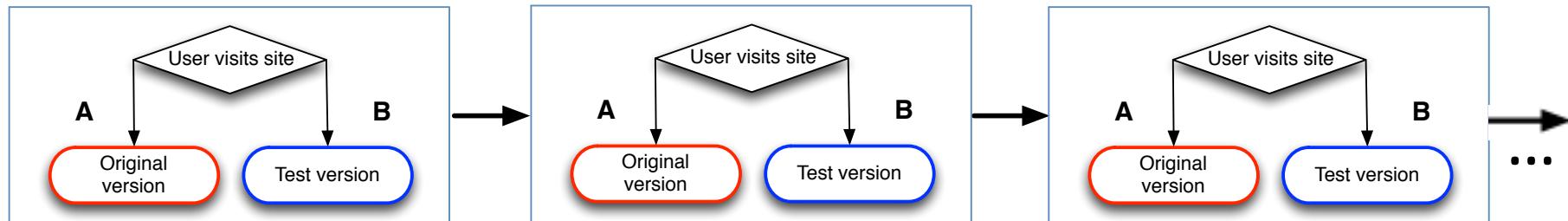
A = 85%

B = 15%

## Phase 3

A = 50%

B = 50%



# Improving Performance

---

- **Mahout (optionally) uses Hadoop**
  - Advice about Hadoop performance tuning generally applies
- **Mahout and Hadoop are implemented in Java**
  - Likewise, advice about Java performance tuning helps too
- **This is typically the domain of system administrators**

## Improving Performance (cont'd)

---

- **Data scientists focus on performance through optimization**
  - Not low-level configuration tuning like system administrators
- **How might a data scientist improve performance?**
  - Using a better similarity metric
  - Designing and implementing a better algorithm
  - Determining what input could be excluded without penalty

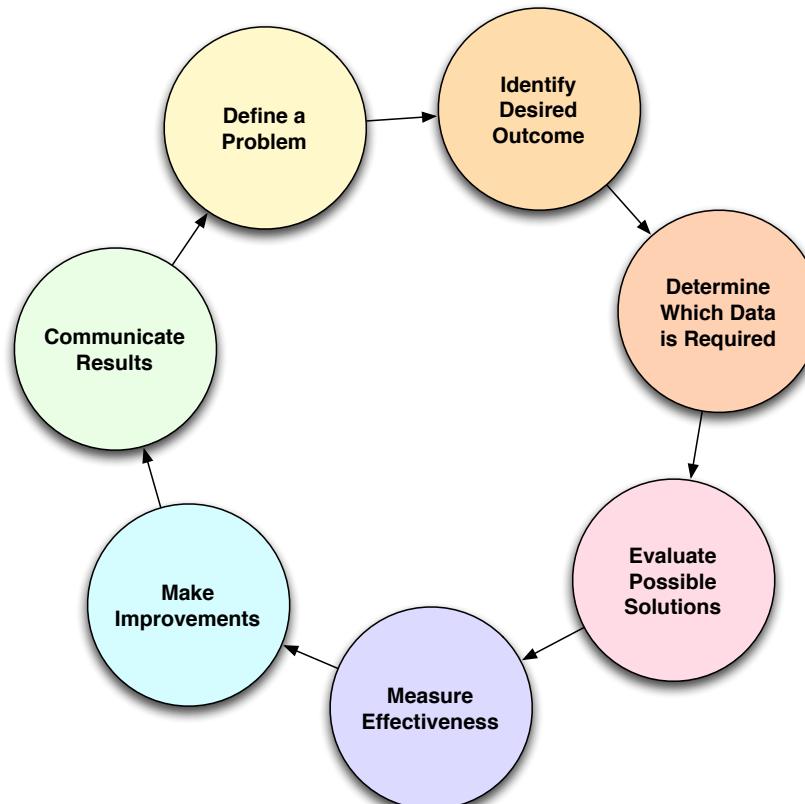
# Chapter Topics

## Production Deployment and Beyond

- Deploying to production
- Tips and techniques for working at scale
- **Summarizing and visualizing results**
- Considerations for improvement
- Next steps for recommenders
- Review questions
- Essential points
- Conclusion

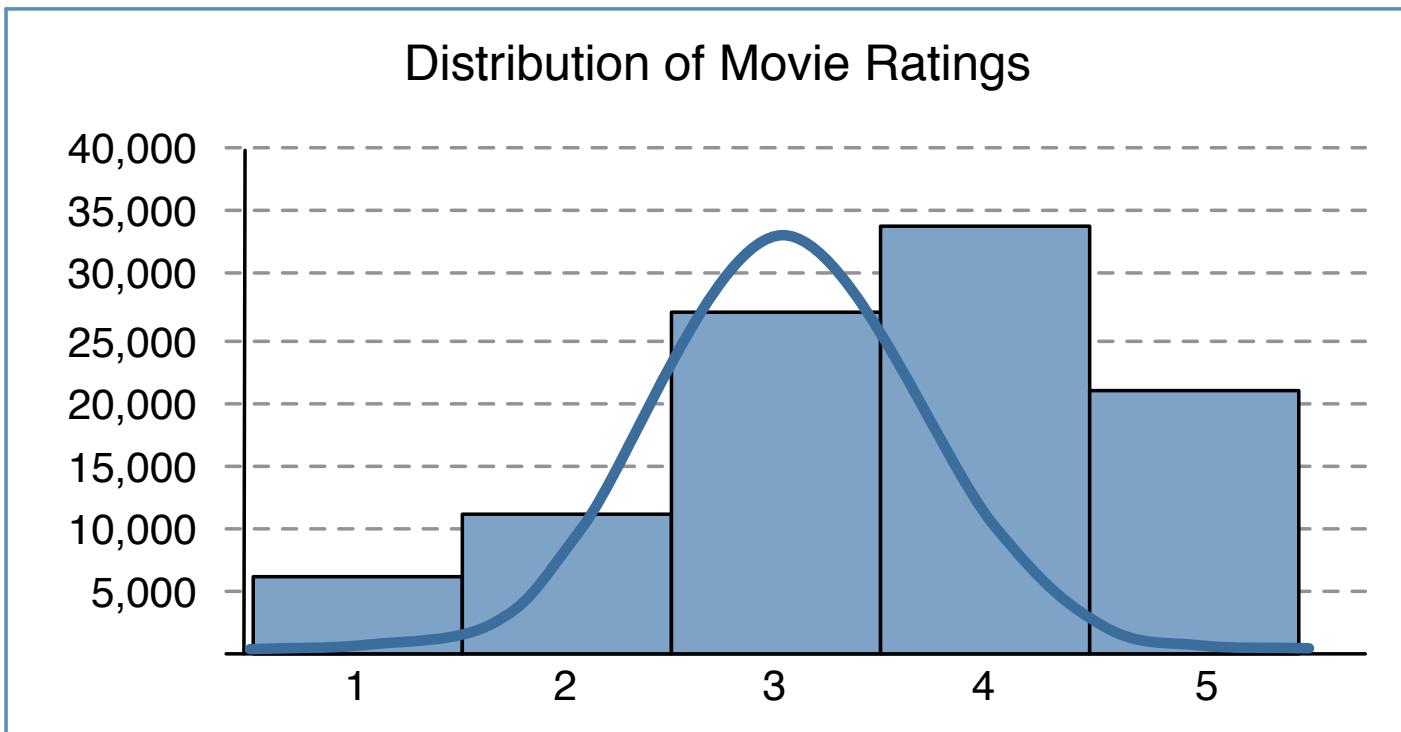
# Acquiring Results from Production

- Following an experiment, you must analyze the data it produced
- You first need to retrieve them from the production system
  - And the data science lifecycle repeats itself



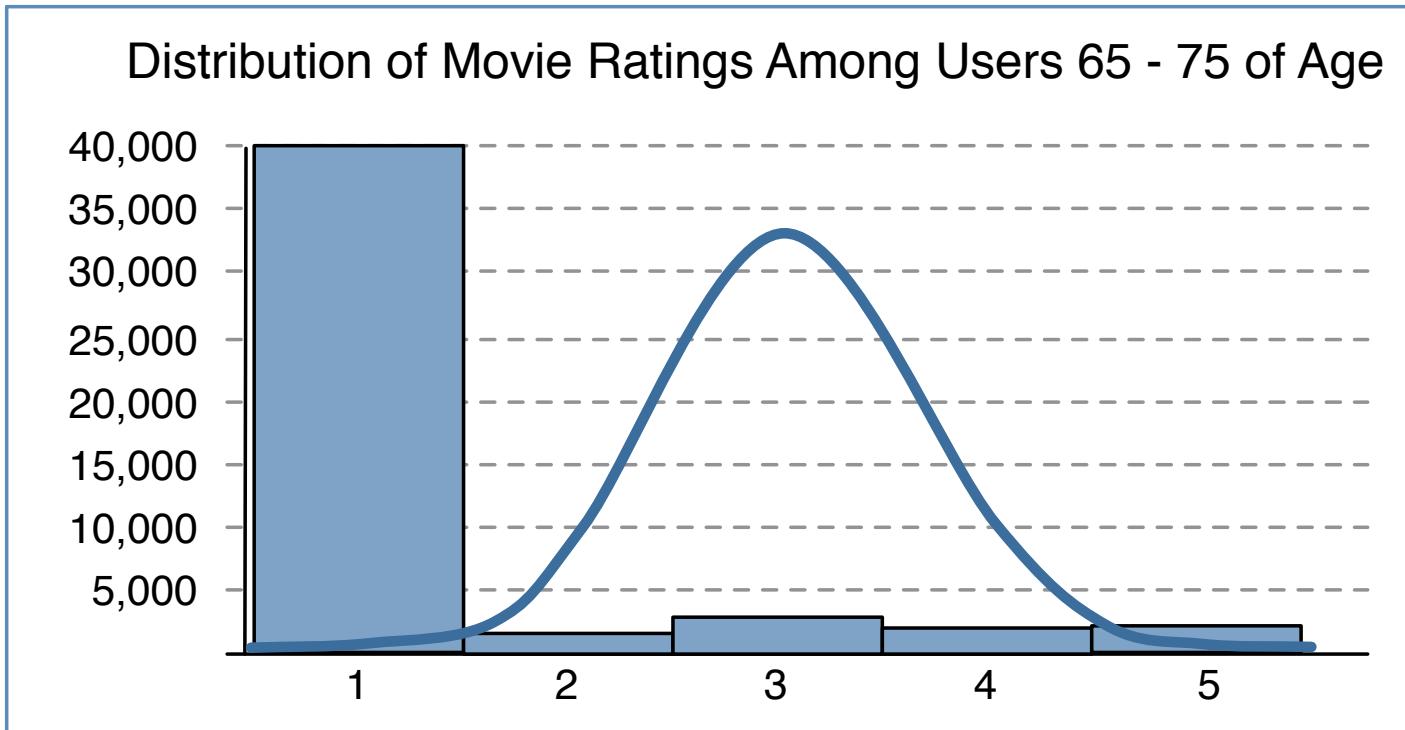
## Summarizing Data

- As with the original data that led to the experiment, one of our first steps should be to analyze the distribution of data



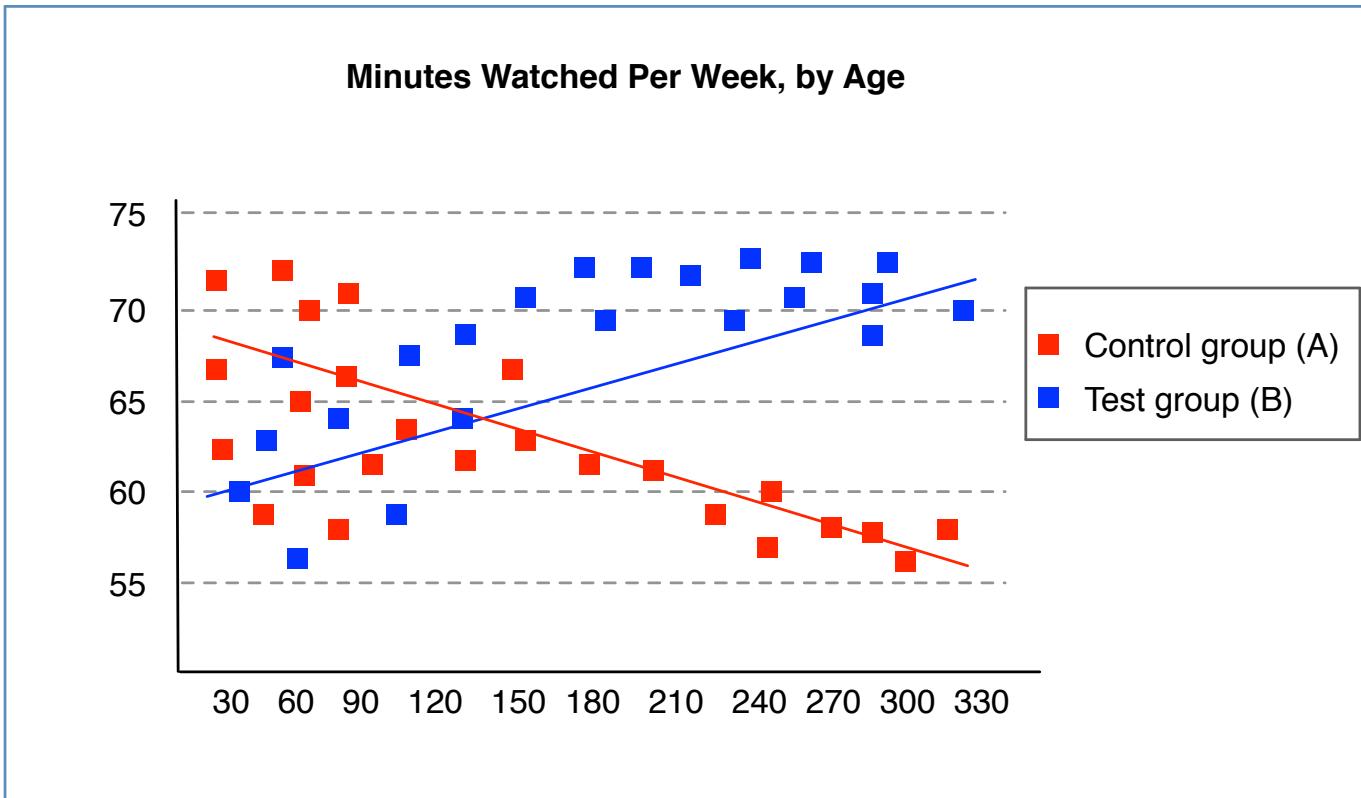
## Summarizing Data (cont'd)

- Analyzing the distribution for subsets of that data is also useful
  - This can lead to additional experiments and improvements



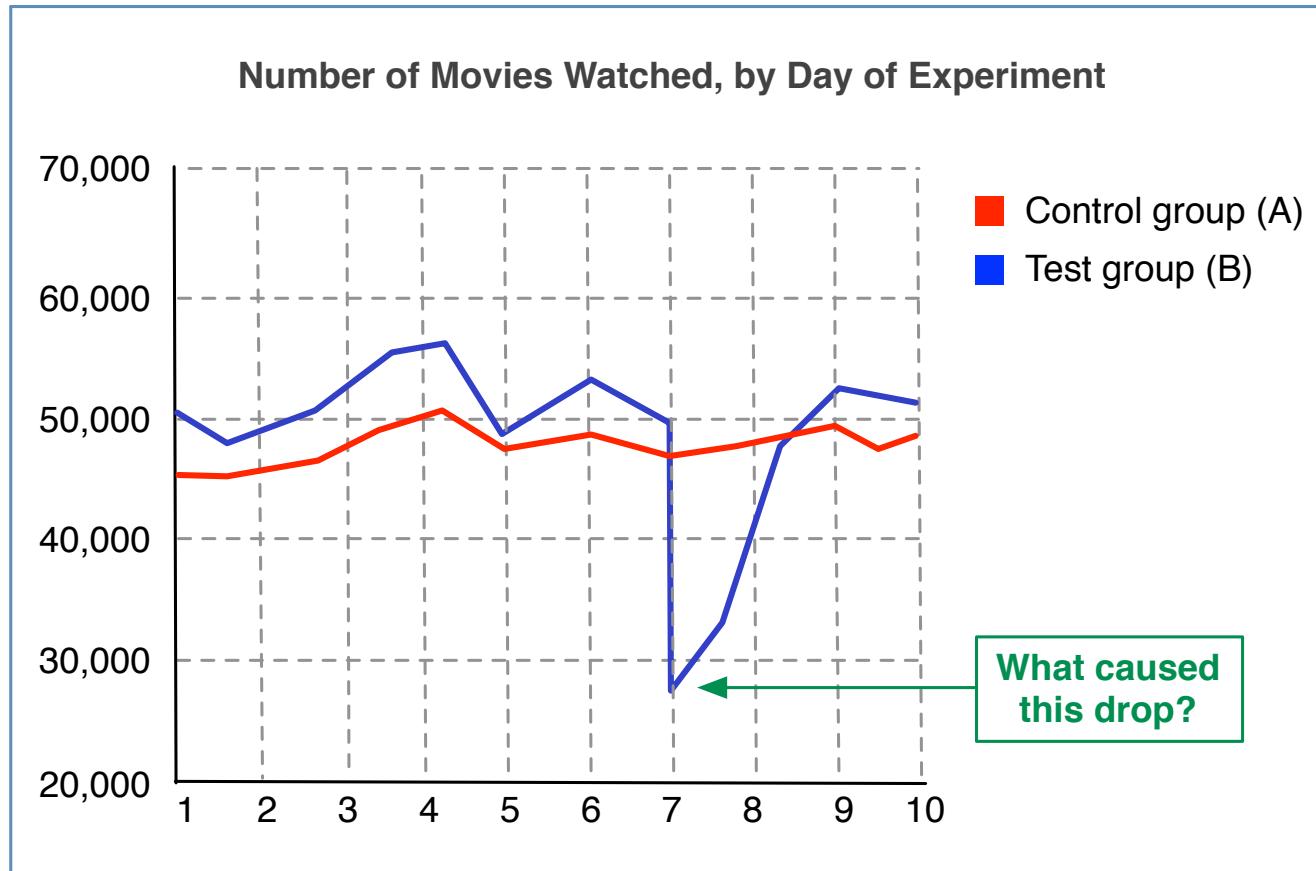
# Visualization

- Scatterplots and linear regression can help you spot trends
  - In this case, it may indicate that older people now find our recommendations more useful



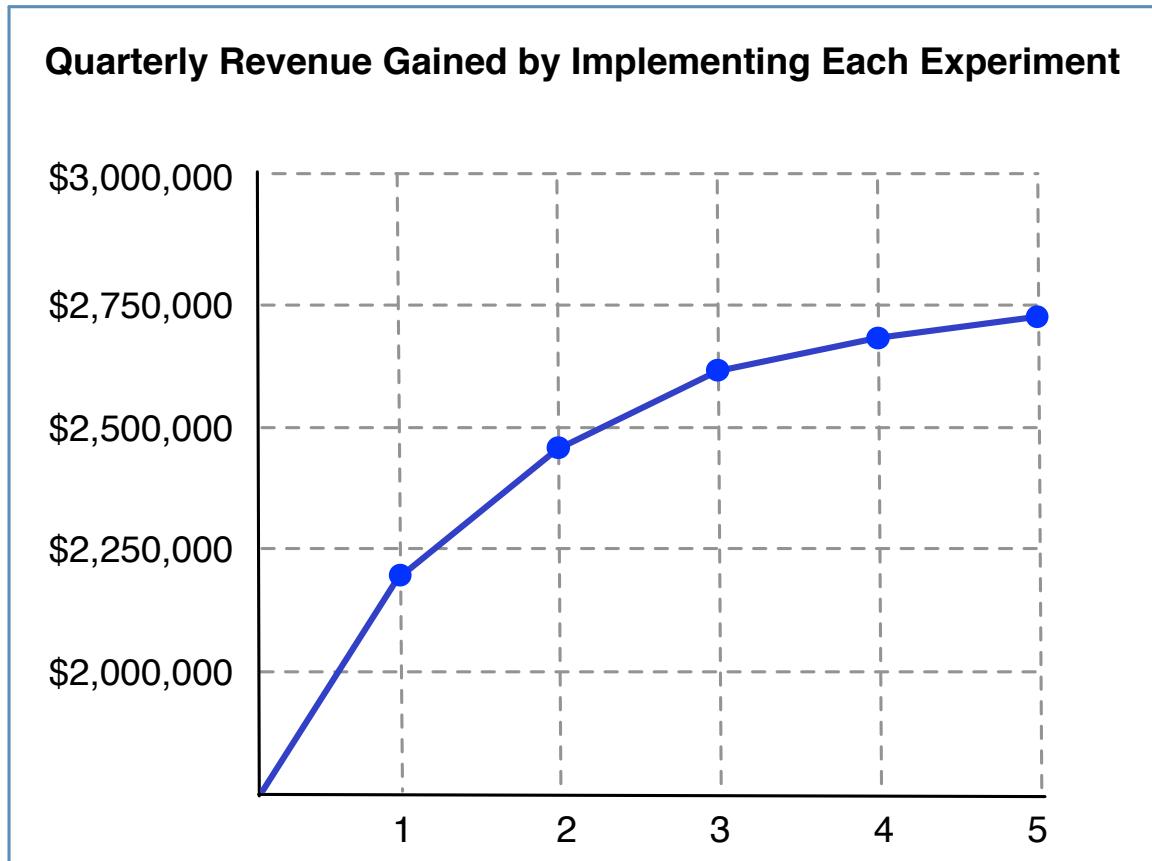
## Visualization (cont'd)

- Time-series plots can also reveal interesting things to explore
  - These are sometimes problems with your implementation



## Visualization (cont'd)

- **Visualization is used to clearly communicate results**
  - Can be an effective way to demonstrate the value you've created



# Chapter Topics

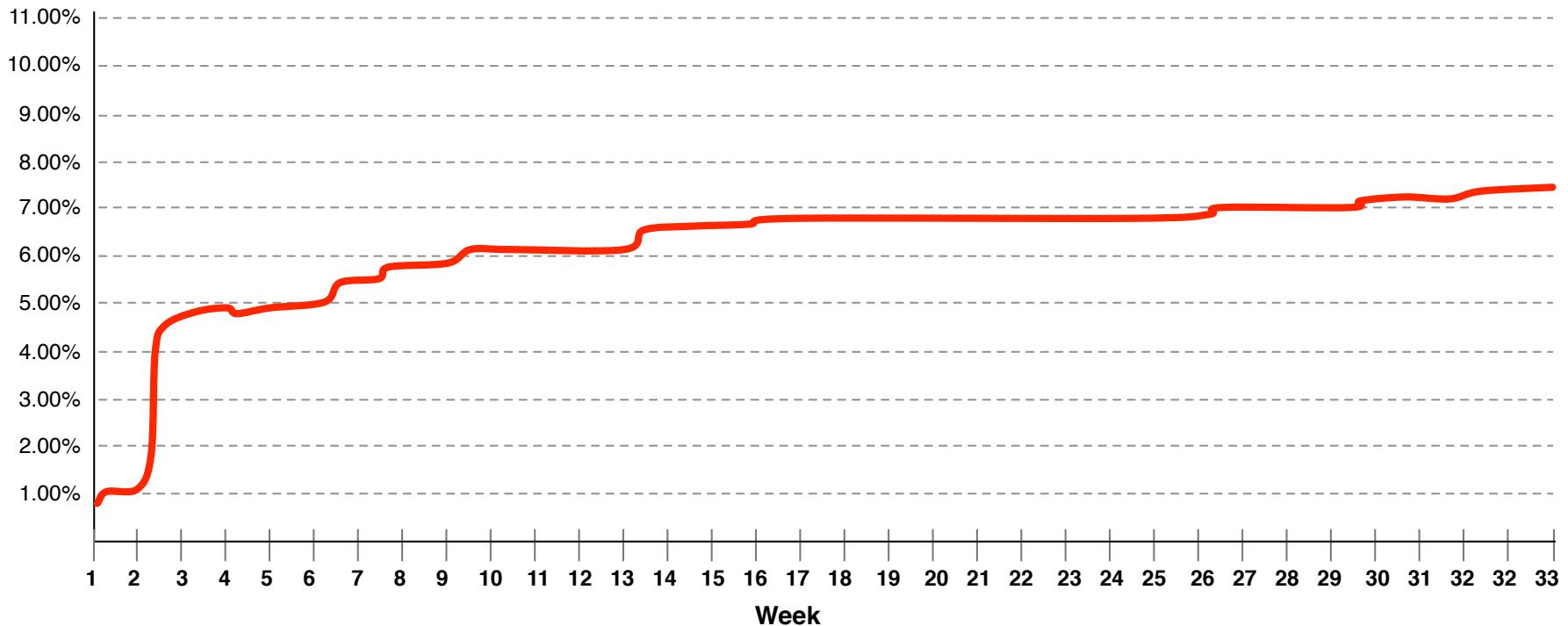
## Production Deployment and Beyond

- Deploying to production
- Tips and techniques for working at scale
- Summarizing and visualizing results
- **Considerations for improvement**
- Next steps for recommenders
- Review questions
- Essential points
- Conclusion

# More Accuracy Can Be Difficult to Achieve

- Improving accuracy becomes more difficult over time
  - Only the hard problems remain after initial “low hanging fruit”

Aggregate Improvement over Cinematch During First Eight Months of Netflix Prize, by Week



# Increasing Accuracy Might Not Pay

---

- **Increased accuracy may be possible, but not be worth the effort**
  - On the other hand, it very well could be
- **Whether the effort is a good investment depends largely on scale**
  - A tiny improvement for a small e-commerce site may net \$5,000
  - The same improvement for a huge retailer could net \$5,000,000
  - Whether it pays off depends on how much it costs to implement

# Perfect Accuracy Is Unobtainable

---

- **Recommenders consume input provided by humans**
  - We don't know how they feel, we only know what they tell us
  - And what they tell us is inconsistent
- **Ratings are subjective and ephemeral**
  - Mood, setting, and many other factors can affect ratings
  - A user may rate the same item differently under other conditions
  - These inconsistencies should be considered noise in the data
- **An RMSE of 0.0 may be possible**
  - But this doesn't mean the recommender is perfectly accurate
  - It really indicates *overfitting* (not properly accounting for noise)

## Reconsider Your Goals

---

- **If you haven't achieved the desired result, take a moment to consider whether you're solving the right problem**
  - Your goals need to be aligned with business interests
- **Recommender accuracy is not our actual goal**
  - Increasing customer satisfaction is more important
  - Reducing the customer cancelation rate is more important still
  - Creating a long-term increase in profits is most important of all

## Reconsider Your Metrics

---

- **Also reconsider how you measure progress towards this goal**
- **Metrics often focus on short-term observations for comparison**
  - But long-term metrics are the true measure of success
- **Take the example of an e-commerce site**
  - Time spent on site is not a good short-term metric
  - Average order value is better
- **A single number can be misleading**
  - Consider using multiple metrics

# The Performance Metric is Key

---

- **Quality and accuracy are important considerations**
  - But we must not lose sight of the original goals
- **Another key measure of effectiveness is performance**
  - How many of our recommendations did the user accept?
- **Performance is essential to a recommender**
  - It's a prerequisite for a meaningful measure of effectiveness
  - More importantly, an unused system has little value to the user

# Chapter Topics

## Production Deployment and Beyond

- Deploying to production
- Tips and techniques for working at scale
- Summarizing and visualizing results
- Considerations for improvement
- **Next steps for recommenders**
- Review questions
- Essential points
- Conclusion

# How Can You Improve Your Recommender?

---

- If you've decided to continue your investment in the recommender, there are many improvements to consider
- We suggest pursuing these through project lifecycle steps
  - Define a problem
  - Identify desired outcome
  - Evaluate possible solutions
  - Determine which data is required to implement the solution
  - Make the improvement
  - Measure its effectiveness through experimentation
- We'll now look at some possible solutions to consider

# Changing Relative Scores

---

- We can experiment with the relative weight of scoring factors
  - Some factors may be more important than first thought
  - Other factors may be less important
  - Still others may be unimportant or even detrimental

Previous %	New %	Description of Criteria
42%	38%	How customer rated similar movies
28%	32%	Cost of royalty payment required to show this movie
17%	16%	Whether the user searched for this movie by name
8%	0%	<del>How long this movie has been in the customer's queue</del>
5%	14%	Popularity of this movie among others in customer's region

# Adding Additional Scoring Factors

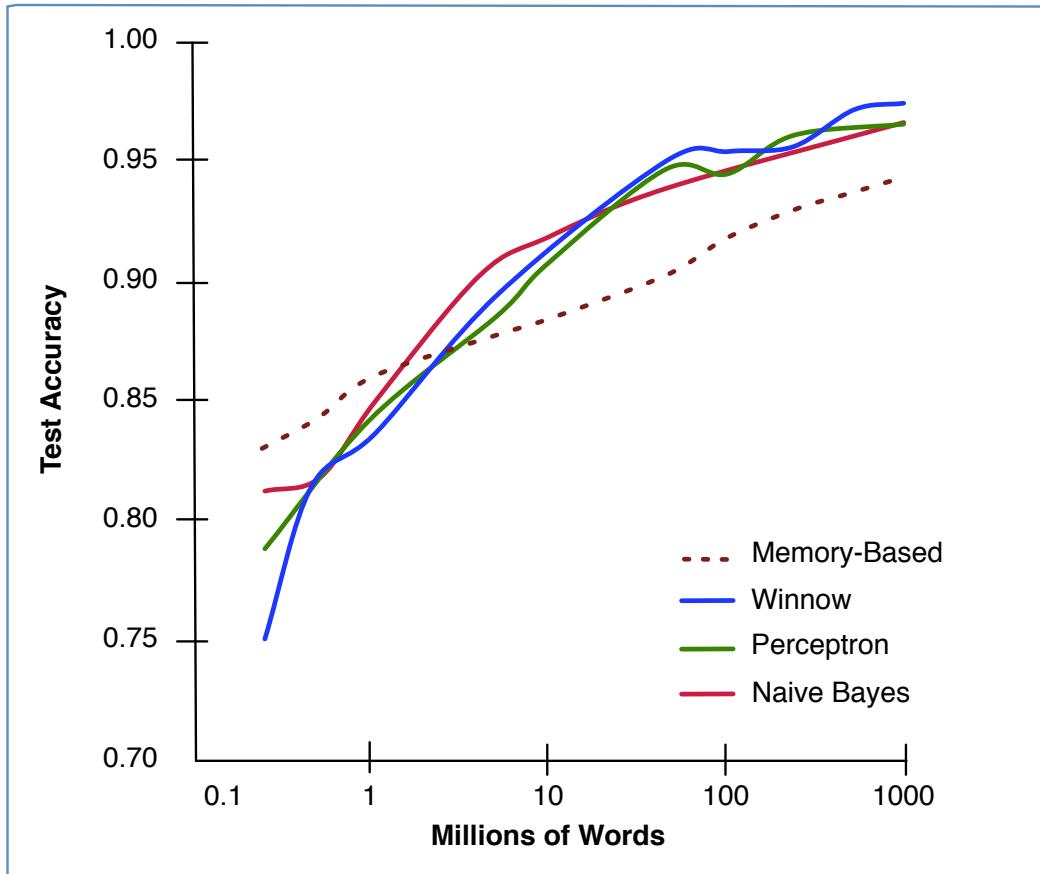
---

- **It's often worthwhile to add entirely new factors**
  - Remember that more data typically yields better results
- **Implicit feedback can be especially valuable**
  - This can offer a more accurate view of what's important to them

Score %	Description of Criteria
34%	How customer rated similar movies
29%	Cost of royalty payment required to show this movie
<b>16%</b>	<b>Did user previously ignore this recommendation</b>
14%	Whether the user searched for this movie by name
5%	How long this movie has been in the customer's queue
2%	Popularity of this movie among others in customer's region

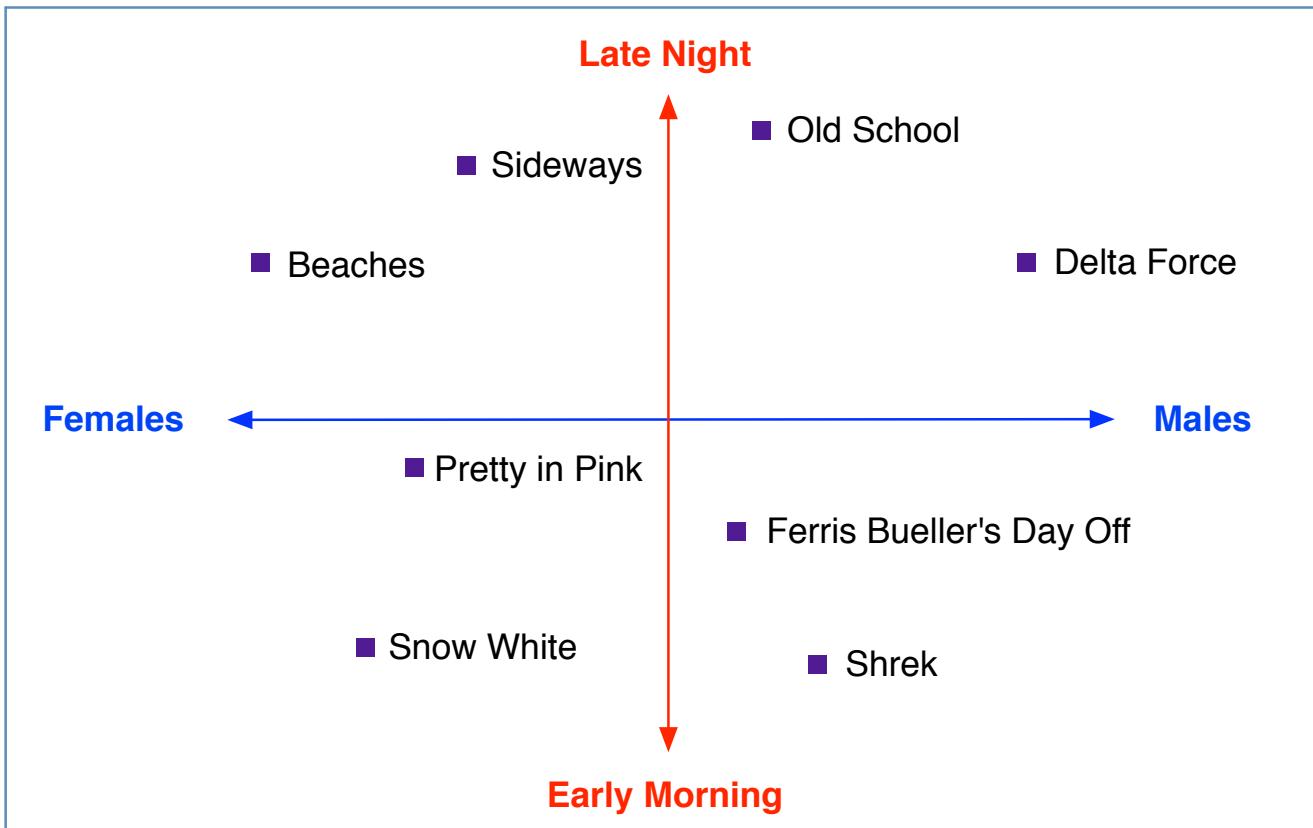
# Experiment with Other Algorithms

- We may find a that another algorithm works better for this case
  - But remember that more data is usually more important



# Matrix Factorization Techniques

- Matrix factorization compares the intersection of two properties
  - Can be used to make more appropriate recommendations



# Hybrid Systems

---

- We may integrate aspects of a content-based recommender
- Content-based recommenders consider properties such as
  - Actors
  - Director
  - Screenwriter
  - Theme
  - Language
- May take into account customer's ratings of the following
  - Movies which feature the same actor(s) or director
  - Movies with similar plots
  - Movies filmed in the same location

## Hybrid Systems (cont'd)

---

- **Current time of year and a movie's theme can influence choices**
  - *Sleepless in Seattle* might be good near Valentine's Day
  - *Halloween* is more appropriate for October
- **We might consider the customer's age and the movie's era**
  - Recommend *Grease* to someone who is 70
  - Recommend *The Breakfast Club* to someone who is 40

# User Interface Improvements

---

- **The UI is an important part of the recommender system**
  - Simple enhancements can often yield significant gains
- **Possible user interface improvements**
  - How we collect movie ratings from customers
  - How customers browse for movies by category
  - How customers search for specific movies

# Chapter Topics

## Production Deployment and Beyond

- Deploying to production
- Tips and techniques for working at scale
- Summarizing and visualizing results
- Considerations for improvement
- Next steps for recommenders
- **Review questions**
- Essential points
- Conclusion

## Review Questions

---

- Why might a small e-commerce site not benefit from improving their recommender while a much larger site would?
- Can you name three ways that Cloudera Movies might be able to improve its recommendations?

# Chapter Topics

## Production Deployment and Beyond

- Deploying to production
- Tips and techniques for working at scale
- Summarizing and visualizing results
- Considerations for improvement
- Next steps for recommenders
- Review questions
- **Essential points**
- Conclusion

## Essential Points

---

- Taking a recommender to production is often just the next step in a continued cycle of experimentation
- It's a good idea to validate input and output before submitting a job to Apache Mahout
- General advice for Hadoop and Java can be useful in tuning Mahout's performance
- There are many techniques that can improve a recommender
  - But it's not always worthwhile or even possible to do so

# Chapter Topics

## Production Deployment and Beyond

- Deploying to production
- Tips and techniques for working at scale
- Summarizing and visualizing results
- Considerations for improvement
- Next steps for recommenders
- Review questions
- Essential points
- **Conclusion**

# Production Deployment and Beyond

---

## In this chapter you have learned

- Which factors may impede recommender accuracy
- How to assess whether further improvements are worthwhile
- Which problems Mahout users often encounter
- Several techniques for improving your recommender system
- What a data scientist's role is in improving system performance

# Bibliography

---

The following offer more information on topics discussed in this chapter

- **Incorporating Contextual Information into Recommender Systems via a Multidimensional Approach**
  - <http://tiny.cloudera.com/dscc14a>
- **Towards the Next Generation of Recommender Systems**
  - <http://tiny.cloudera.com/dscc14b>
- **Matrix Factorization Techniques for Recommender Systems**
  - <http://tiny.cloudera.com/dscc14c>

# Conclusion

---

Chapter 15



# Course Chapters

- Introduction
- Data Science Overview
- Use Cases
- Project Lifecycle
- Data Acquisition
- Evaluating Input Data
- Data Transformation
- Data Analysis and Statistical Methods
- Fundamentals of Machine Learning
- Recommender Overview
- Introduction to Apache Mahout
- Implementing Recommenders with Apache Mahout
- Experimentation and Evaluation
- Production Deployment and Beyond
- Conclusion

# Chapter Topics

## Conclusion

- Essential points
- Next steps

## Essential Points

---

- **Data science is the combination of several skills, including mathematics, software engineering, communications, and domain experience**
- **Data products are built from data and produce new data as they're used, allowing them to be further improved**
- **Key themes of data science use cases include**
  - Access to large amounts of data
  - Acquisition of several types of data from different sources
  - The ability to analyze this data at scale to find interesting patterns
  - Problem solving focused on a specific and actionable result
- **The lifecycle of a data science project is iterative**
  - Stating a problem and constantly measuring results are key

## Essential Points (cont'd)

---

- **Data can be found both inside and outside your organization**
  - It's not always provided in the most convenient formats
  - Issues with data quality are inevitable at scale
  - Filtering, normalization, and transformation are often required
- **Summarizing and visualizing data is an important first step**
  - These techniques can reveal errors or other interesting patterns
  - Skewed distributions can be misleading

## Essential Points (cont'd)

---

- **More data is usually preferable to a better algorithm, at scale**
- **Machine learning is often described in terms of the Three C's**
  - Collaborative Filtering (recommendations)
  - Clustering (grouping items into subsets)
  - Classification (identifying items by type)
- **Two main approaches to recommenders**
  - Content-based recommenders consider an item attributes
  - Collaborative filtering considers the actions of other users
  - A hybrid of these two approaches is also possible
- **Collaborative filtering can be user-based or item-based**
  - The item-based approach tends to scale better

## Essential Points (cont'd)

---

- Preferences can be captured explicitly or implicitly
- Similarity metrics are chosen based on preference data type
  - Binary data
    - Tanimoto coefficient
    - Log likelihood
  - Numeric data
    - Euclidean distance
    - Uncentered cosine
    - Pearson correlation
- There's no “best” similarity metric
  - Each one is better in some cases than others
  - Magnitude and ratings bias are also important considerations

## Essential Points (cont'd)

---

- **Apache Mahout is an open source machine learning library**
  - It's really a collection of various implementations and utilities
  - Many (though not all) of Mahout's algorithms can use Hadoop for better performance and scalability
- **Scoring is a key aspect of building recommender systems**
  - Indicates relative importance of various criteria
  - Often requires domain knowledge
  - Can yield significant value for the organization
- **The user interface is an essential part of recommender systems**

## Essential Points (cont'd)

---

- **An experiment can help you evaluate different solutions**
  - And select the ones that will provide the best benefit
- **Successful organizations constantly experiment**
  - They often employ A/B tests as a way of comparing treatments
  - Data matters more than opinions when making decisions
- **Measuring recommender effectiveness frequently involves**
  - RMSE (accuracy)
  - Precision and recall (quality)
  - Performance (acceptance rate)

## Essential Points (cont'd)

---

- **Production deployment is often just the start of a new experiment**
- **There are many techniques for making further improvements**
  - Not all of these will prove worthwhile to implement
  - Always consider your costs and potential gains

# Chapter Topics

## Conclusion

- Essential points
- **Next steps**

# Which Course to Take Next?

---

**Cloudera offers a range of training courses**

- **For developers**
  - Developer Training for Apache Hadoop
  - Cloudera Training for Apache HBase
- **For analysts and DBAs**
  - Cloudera Training for Apache Hive and Pig
- **For architects, managers, CIOs and CTOs**
  - Essentials for Apache Hadoop
- **For system administrators**
  - Administrator Training for Apache Hadoop

# Certification Overview

---

- **Cloudera offers a range of industry-recognized certifications**
  - Cloudera Certified Developer for Apache Hadoop
  - Cloudera Certified Administrator for Apache Hadoop
  - Cloudera Certified Specialist in Apache HBase
- **Cloudera Certified Professional (CCP) Data Scientist coming in 2013**
  - Our Data Scientist certification will involve a two-step process

# Certification Overview (cont'd)

---

- **Step One: CCP Data Scientist Written Examination**
  - A two-hour written qualification exam covering concepts and skills from a broad range of data science topics
- **Step Two: CCP Data Scientist Lab Examination**
  - This hands-on exam measures both your technical ability and your capacity to develop creative approaches to building data products.
  - Must pass written exam before you're eligible to schedule the lab exam
- **You'll receive a free invitation to participate in the beta and final versions of the written exam when they're available**
- **For more information about Cloudera certification, refer to <http://university.cloudera.com/certification.html>**

# Thanks!

---

- **Thank you for attending the course!**
- **If you have any questions or comments, please contact us via**  
<http://www.cloudera.com/>

# Hadoop Overview

## Appendix A



# What is Apache Hadoop?

---

- **It's a scalable data storage and processing system**
  - Open source Apache project
  - Harnesses the power of industry-standard hardware
  - Distributed and fault-tolerant
  - Mostly written in Java
- **“Core” Hadoop consists of two main parts**
  - Storage: Hadoop Distributed File System (HDFS)
  - Processing: MapReduce
- **Many other tools use Hadoop or are built on top of Hadoop**
  - This includes Hive, Sqoop, Flume, and Mahout
  - These are collectively known as the “Hadoop ecosystem”

# HDFS: Hadoop Distributed File System

---

- **Based on Google's GFS (Google File System)**
  - Highly optimized for processing data with MapReduce
- **Provides storage for massive amounts of data**
  - Using inexpensive commodity hardware
  - Cost per GB is typically about 1/10<sup>th</sup> that of enterprise storage
- **At load time, data is distributed across all nodes**
  - This improves reliability *and* performance

# Comparing HDFS to Other Filesystems

---

- **HDFS is not built into the operating system**
  - It must be accessed via the `hadoop fs` command
    - Or via the REST or Java APIs
- **In some ways, HDFS is similar to a UNIX filesystem**
  - Hierarchical
  - UNIX-style paths (e.g. `/foo/bar/myfile.txt`)
  - File ownership and permissions
- **There are also some major deviations from UNIX**
  - No concept of a current directory
  - Cannot modify files once written

# Using Hadoop Commands to Access HDFS

- **The hadoop fs utility has several subcommands**
  - These are very similar to standard UNIX commands
- **The examples below show a few common uses**
  1. List the contents of the /clouderamovies directory
  2. Display the contents of a file stored in HDFS
  3. Copy a local file to the /clouderamovies directory in HDFS
  4. Copy a file from HDFS to the local filesystem

```
$ hadoop fs -ls /clouderamovies  
$ hadoop fs -cat /clouderamovies/remotefile.txt  
$ hadoop fs -put localfile.txt /clouderamovies  
$ hadoop fs -get /clouderamovies/remotefile.txt
```

# MapReduce Introduction

---

- **MapReduce is not a language, it's a programming model**
  - A style of processing data you could implement in any language
- **MapReduce has its roots in functional programming**
  - Many languages have functions named `map` and `reduce`
  - These functions have largely the same purpose in Hadoop
- **Popularized for large-scale data processing by Google**

# MapReduce Benefits

---

- **Scalable because you process one record at a time**
  - The input data and processing tasks can be distributed across (potentially thousands of) machines for faster processing
- **Complex details are abstracted away from the developer**
  - MapReduce jobs don't require writing code to handle
    - Reading input from files
    - Writing output to files
    - Networking among nodes in a cluster
    - Synchronization between processes

# Understanding Map and Reduce

---

- **MapReduce consists of two functions: Map and Reduce**
  - The output from Map becomes the input to Reduce
  - Hadoop automatically sorts and groups data between these functions
- **The Map function always runs first**
  - Typically used to filter, transform, or parse data
- **The Reduce function is optional**
  - Normally used to summarize data from the Map function
  - Since this isn't always needed, you can run “Map-only” jobs
- **Each piece is simple, but can be powerful when combined**

# MapReduce Example in Python

---

- **MapReduce code for Hadoop is typically written in Java**
  - But possible to use nearly any language with *Hadoop Streaming*
- **The following example will use MapReduce in Python**
  - It processes log files in order to summarize events by type
- **The example will illustrate both the data flow and the code**

## Job Input

---

- Our job will count the event type (highlighted below) across all log files supplied as input to the job

```
2012-09-06 22:16:49.391 CDT INFO "This can wait"  
2012-09-06 22:16:49.392 CDT INFO "Blah blah"  
2012-09-06 22:16:49.394 CDT WARN "Hmmm...."  
2012-09-06 22:16:49.395 CDT INFO "More blather"  
2012-09-06 22:16:49.397 CDT WARN "Hey there"  
2012-09-06 22:16:49.398 CDT INFO "Spewing data"  
2012-09-06 22:16:49.399 CDT ERROR "Oh boy!"
```

- Each mapper gets a chunk of entire job's input data to process
  - This “chunk” is called an **InputSplit**

# Python Code for Map Function

- Our map function will parse the event type
  - And then output that event (key) and a literal 1 (value)

```
1 #!/usr/bin/env python
2
3 import sys
4
5 levels = ['TRACE', 'DEBUG', 'INFO',
6           'WARN', 'ERROR', 'FATAL']
7
8 for line in sys.stdin:
9     fields = line.split()
10    for field in fields:
11        field = field.strip().upper()
12        if field in levels:
13            print "%s\t1" % field
```

Define list of known log events

Split every line (record) we receive on standard input into fields, normalized by case

If this field matches a log level, print it, a tab separator, and the literal value 1

# Output of Map Function

---

- The map function produces key/value pairs as output

INFO	1
INFO	1
WARN	1
INFO	1
WARN	1
INFO	1
ERROR	1

# Input to Reduce Function

---

- **The Reducer receives a key and all values for that key**
  - Keys are always passed to reducers in sorted order
  - Although not obvious here, values are unordered

ERROR	1
INFO	1
WARN	1
WARN	1

# Python Code for Reduce Function

- The Reducer first extracts the key and value it was passed

```
1 #!/usr/bin/env python  
2  
3 import sys  
4  
5 previous_key = ''  
6 sum = 0  
7  
8 for line in sys.stdin:  
9     fields = line.split()  
10    key, value = line.split()  
11  
12    value = int(value)  
13    # continued on next slide
```

Initialize loop variables

Extract the key and value passed via standard input

# Python Code for Reduce Function

- Then simply adds up the value for each key

```
14 # continued from previous slide
15 if key == previous_key:
16     sum = sum + value
17 else:
18     if previous_key != '':
19         print '%s\t%i' % (previous_key, sum)
20     previous_key = key
21     sum = 1
22
23 print '%s\t%i' % (previous_key, sum)
```

If key unchanged,  
increment the count

If key changed, print  
sum for previous key

Re-init loop variables

Print sum for final key

# Output of Reduce Function

---

- The output of this Reduce function is a sum for each level

ERROR	1
INFO	4
WARN	2

# Recap of Data Flow

## Map input

```
2012-09-06 22:16:49.391 CDT INFO "This can wait"  
2012-09-06 22:16:49.392 CDT INFO "Blah blah"  
2012-09-06 22:16:49.394 CDT WARN "Hmmm..."  
2012-09-06 22:16:49.395 CDT INFO "More blather"  
2012-09-06 22:16:49.397 CDT WARN "Hey there"  
2012-09-06 22:16:49.398 CDT INFO "Spewing data"  
2012-09-06 22:16:49.399 CDT ERROR "Oh boy!"
```

## Map output

INFO	1
INFO	1
WARN	1
INFO	1
WARN	1
INFO	1
ERROR	1

## Reduce input

ERROR	1
INFO	1
WARN	1
WARN	1

## Reduce output

ERROR	1
INFO	4
WARN	2

# How to Run a Hadoop Streaming Job

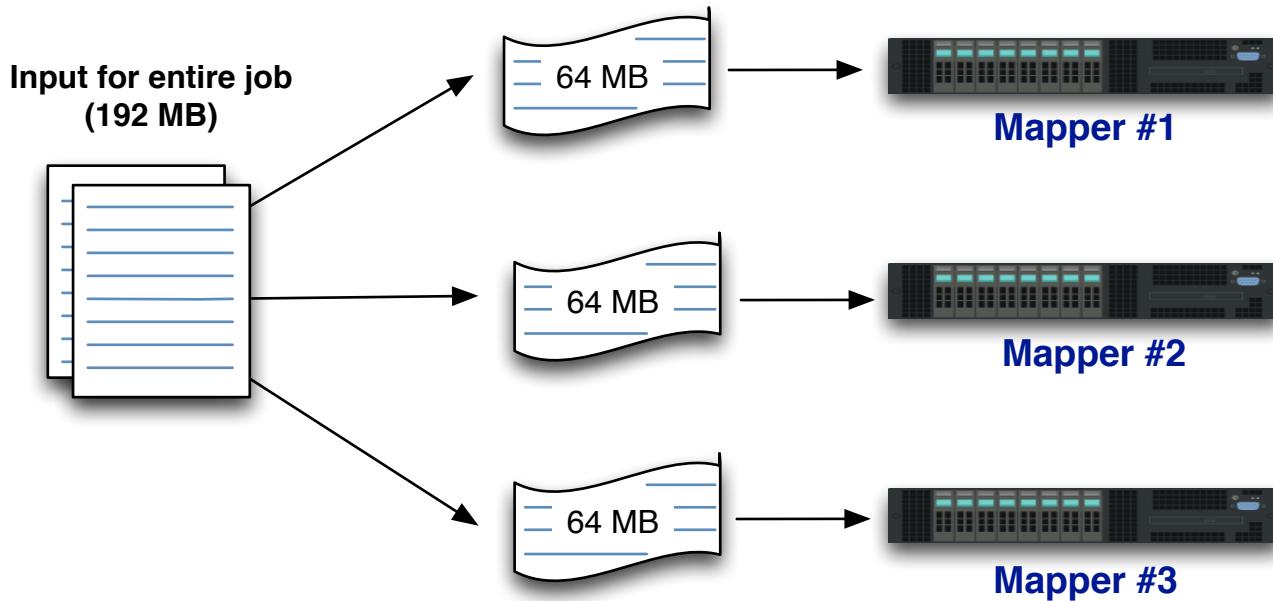
- Jobs are submitted via the `hadoop jar` command

```
$ hadoop jar $STREAMJAR \
  -mapper mapper.py -file mapper.py \
  -reducer reducer.py -file reducer.py \
  -input /user/training/input \
  -output /user/training/output
```

- The `$STREAMJAR` variable is defined in your virtual machine
  - It points to the Hadoop Streaming JAR file
  - Outside of your VM, you'll have to specify this file's path
    - Exact name and location vary based on Hadoop version
- The `-input` argument specifies HDFS path for job input
- The `-output` argument specifies HDFS path for job output
  - It must not already exist or the job will fail

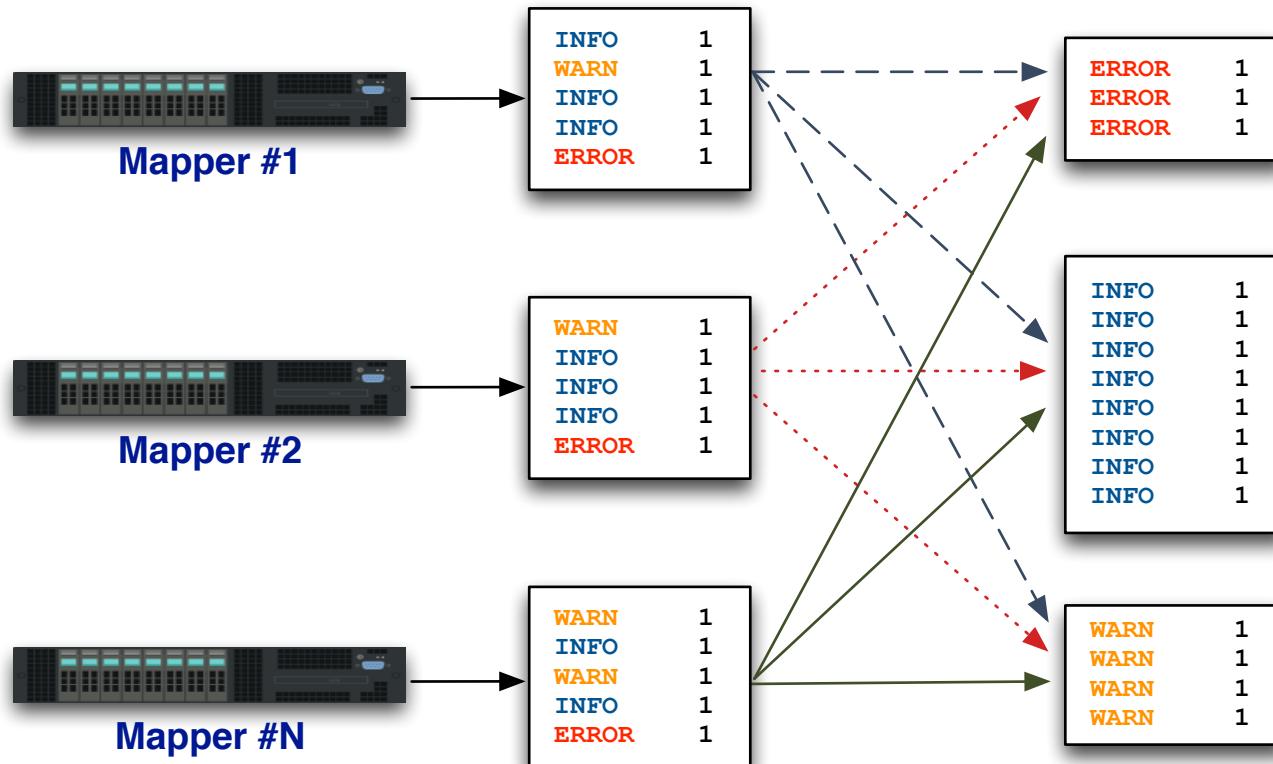
# Input Data Feeds the Map Tasks

- **Input for the entire job is subdivided into InputSplits**
  - Each of these serves as input to a single Map task



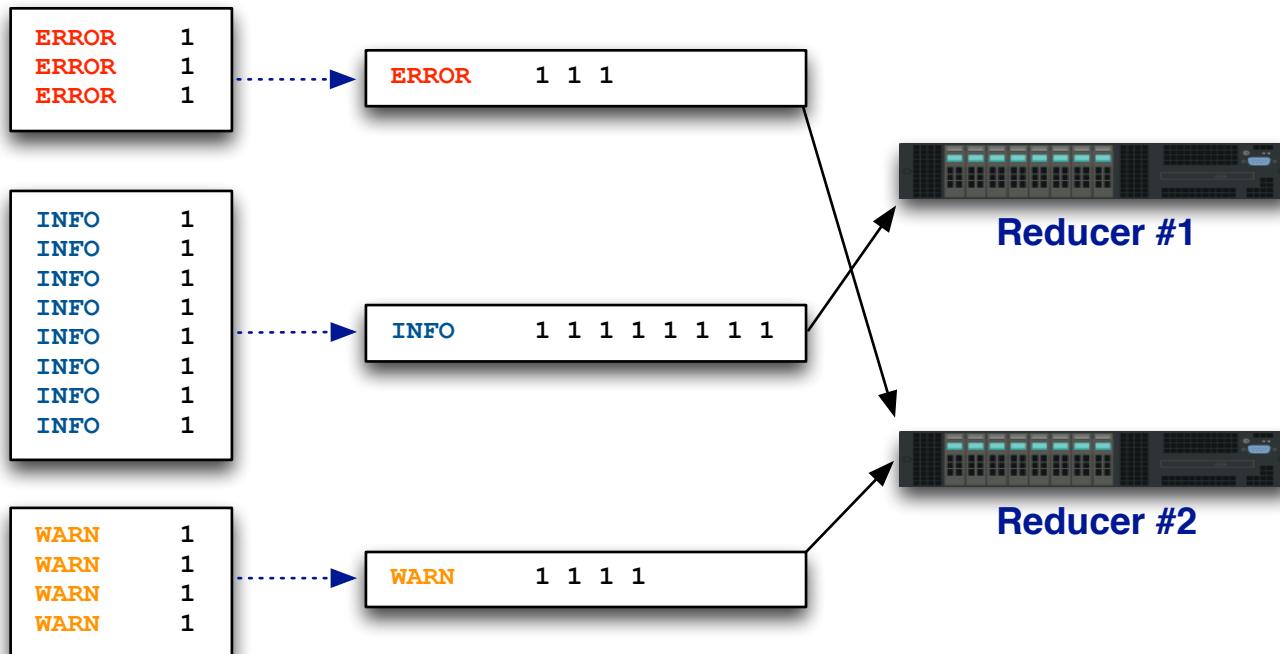
# Mappers Feed the Shuffle and Sort

- Output of all Mappers is partitioned, merged, and sorted
  - No code required – Hadoop does this automatically



# Shuffle and Sort Feeds the Reducers

- All values for a given key are then collapsed into a list
  - The key and all its values are fed to reducers as input
  - Note that it is common for jobs to have only one reducer



# Each Reducer Produces an Output File

- These are stored in HDFS below your output directory
  - Use `hadoop fs -getmerge` to combine them into a local copy



Reducer #1



Reducer #2

# Mathematical Formulas

## Appendix B



## Tanimoto Coefficient

---

- The Tanimoto coefficient measures the items shared between two sets and can be expressed using the formula below

$$T = \frac{N_c}{(N_a + N_b - N_c)}$$

- Where
  - $T$  is the Tanimoto coefficient
  - $N_a$  is the number of items in set A
  - $N_b$  is the number of items in set B
  - $N_c$  is the number of items in the intersection of sets A and B

## Log Likelihood

---

- Log likelihood is a similarity metric for binary data
- Computing the log likelihood is a multi-step process that involves first computing the entropy for the values in rows and columns for a contingency matrix like this

	How many watched X?	How many did not watch X?
How many watched Y?	# who watched both X and Y	# who watched Y but not X
How many did not watch Y?	# who watched X but not Y	# who watched neither X nor Y

- The code that Mahout uses to calculate log likelihood can be found here
  - [http://tiny.cloudera.com/dsc\\_apc\\_01](http://tiny.cloudera.com/dsc_apc_01)

## Euclidean Distance

---

- Euclidean distance measures the distance between two points and can be expressed using the formula below

$$D = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

- Where
  - D is the Euclidean distance
  - $q_i$  and  $p_i$  represent coordinates for corresponding points in a set

## Pearson Correlation

---

- The Pearson correlation is used to measure similarity between sets of values and can be expressed using the formula below

$$P = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right) \left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}}$$

- Where

- P is the Pearson Correlation
- X is the first set of values
- Y is the second set of values
- N is the size of set X (which must also match the size of set Y)

# Root Mean Squared Error

---

- Root Mean Squared Error (RMSE) is a common metric for measuring the accuracy of predicted ratings and can be expressed using the formula below

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (x_i - y_i)^2}{N}}$$

- Where
  - $x_i$  is a value from the set of predicted ratings
  - $y_i$  is a value from the set of actual ratings

# Language and Tool Reference

## Appendix C



# UNIX Command Line Tools: Finding Help

---

- The next few slides explain a few common UNIX commands.
- You can find complete information by using the **man** command
  - For example, use this command to read about the **sort** utility

```
$ man sort
```

- Many commands show a summary when called with **--help**

```
$ sort --help
```

# UNIX Command Line Tools: Viewing Files

- The **cat** command can display the entire content of a file

```
$ cat example.txt
This is the first line of
the example.txt file. There
are only four lines of text
and this is the last line.
```

- You can use the ‘>’ redirection operator to concatenate the contents of several files together

```
$ cat file01.txt file02.txt > combined.txt
```

# UNIX Command Line Tools: head and tail

- The **head** command can display just the first few lines of a file
  - It shows ten lines by default
  - Use the **-n** option to see a specified number of lines

```
$ head -n1 example.txt  
This is the first line of
```

- The **tail** command is similar, but shows the last few lines
  - Like the **head** command, it also supports a **-n** option

```
$ tail -n2 example.txt  
are only four lines of text  
and this is the last line.
```

## UNIX Command Line Tools: cut

- The **cut** command can display a particular field from a file
  - Based on the index specified in the **-f** option
  - This example also demonstrates how commands can be connected together via the pipe ‘|’ operator

```
$ head -n 3 tab-delimited-data.txt
202    Washington D.C.
212    New York City
213    Los Angeles

$ head -n 3 | cut -f 1 tab-delimited-data.txt
202
212
213
```

- An alternate delimiter can be specified via the **-d** option

# UNIX Command Line Tools: sort and uniq

- The **sort** command can be used to sort the contents of a file
  - This is done in lexicographic order by default, but you can sort numerically by specifying the **-n** option
  - The **-r** option will sort in reverse order

```
$ sort -rn areacodes.txt | head -n 2
989      Michigan
985      Louisiana
```

- The **uniq** command will remove all duplicates from sorted input
  - The **-c** option will count the occurrences of the input data

```
$ sort names.txt | uniq -c | sort -rn
61      Smith
31      Brown
```

# UNIX Command Line Tools: grep

- The **grep** command is used to filter input data
  - Case-sensitive by default; case-sensitivity can be disabled via **-i** option

```
$ grep -i smith addressbook.txt
Smith, Jim      (314) 555-7234
Smith, Linda   (415) 555-3678
```

- The **-v** option will output only lines that do **not** match the pattern
- The **egrep** variant is similar, but supports regular expressions

```
$ egrep -i 'sm[iy]the?' addressbook.txt
Smith, Jim      (314) 555-7234
Smith, Linda   (415) 555-3678
Smithe, Joe    (213) 555-1395
Smythe, Paula  (504) 555-6128
```

# Sqoop

---

- **Sqoop exchanges data between a relational database and HDFS**
  - Can import all tables, a single table, or a partial table into HDFS
  - Data can be imported in delimited text or Avro file format
  - Sqoop can also export data from HDFS to a database
- **The following command imports a table from a MySQL database into HDFS as a tab-delimited file**

```
$ sqoop import \
--connect jdbc:mysql://localhost/movielens \
--username training --password training \
--fields-terminated-by '\t' \
--warehouse-dir /clouderamovies \
--table movie
```

# Hive

---

- **Hive is an alternative to writing low-level MapReduce code**
  - Users can analyze data stored in Hadoop data via HiveQL
    - HiveQL is a declarative language very similar to SQL
- **Hive does *not* turn your Hadoop cluster into a database**
  - Instead, the Hive interpreter turns HiveQL into MapReduce jobs
  - Hive tables are simply directories of data stored in HDFS
    - The `create table` statement instructs Hive how to parse it
- **Hive is especially useful for joining data**

```
SELECT customer.id, customer.name, sum(order.cost)
      FROM customer INNER JOIN order
        ON (customer.id = order.customer_id)
     WHERE customer.zipcode = '63105'
 GROUP BY customer.id, customer.name;
```

# Invoking Hive

- There are three main ways to execute HiveQL commands
  1. Directly, via the Hive shell
  2. By specifying a HiveQL command in a string
  3. By specifying a text file containing HiveQL code

```
$ hive  
hive> select count(*) from mytable;  
912  
  
$ hive -e "select count(*) from mytable"  
912  
  
$ hive -f mycommands.hql  
912
```

NOTE: log and status messages normally seen when running Hive commands have been removed for brevity

# Python Basics

---

- **Python is a popular general-purpose programming language**
- **Similar to Java in some ways**
  - High-level language
  - Supports object-oriented programming
  - Cross-platform (Linux, Macintosh, Windows, and other systems)
- **Differs from Java in others ways**
  - No compilation step required
  - Dynamically typed
    - This means you're not required to explicitly state what type of data a variable holds, as you do in Java
  - Uses whitespace (rather than braces) to delimit blocks of code

# Accessing Python

- You can interact directly with the Python interpreter
  - The >>> represents the interpreter's prompt

```
$ python
>>> print "hello python"
hello python
```

- However, it's more common to run source code from a file

```
$ cat myprogram.py
print "hello world"

$ python myprogram.py
hello python
```

# Python: Strings

---

- A string is defined inside double quotes
  - In Python, strings can be treated as character arrays (see below)
  - Arrays in are indexed from zero, as in C or Java
- The `len` function returns the length of the string passed to it
- The `strip()` method removes leading and trailing whitespace

```
>>> x = "example"
>>> print x[0]
e
>>> print x[0:4]
exam
>>> x = " example "
>>> print len(x)
9
>>> print len(x.strip())
7
```

# Python: Loops and Conditional Expressions

---

- Python uses whitespace to denote blocks of code
  - Be careful how you indent things!
- Loops and conditional expressions both contain a colon

```
x = "this is a test"
for word in x.split():
    if word == "test":
        print "It's a test!"
    else:
        print "Not a test"
```