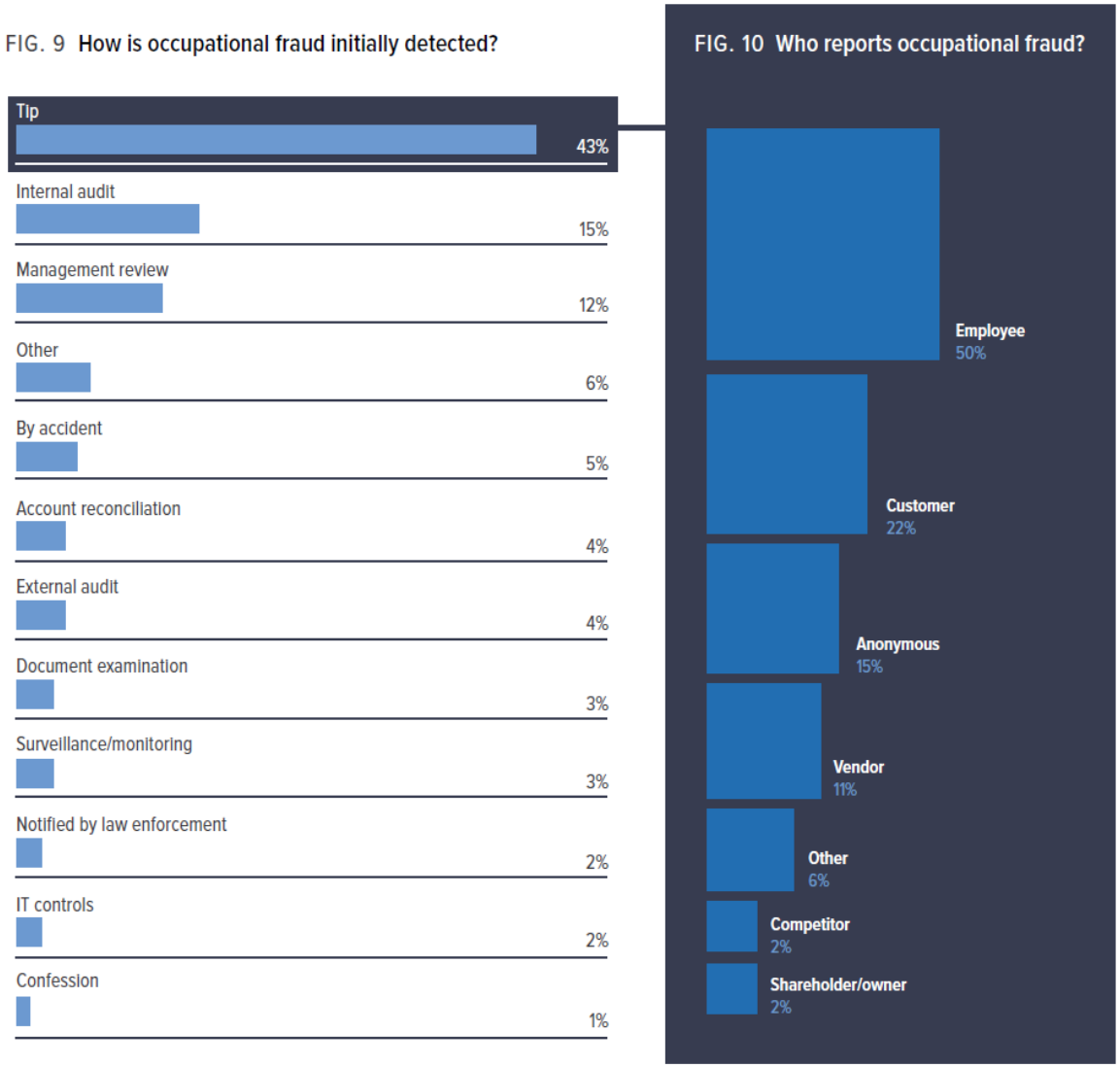
Benford's Law, Anomaly Detection, and Fraud Auditing

Robson Glasscock, PhD, CPA

Big Picture

The ACFE's 2020 Report to the Nations contains the following graphic:



Big Picture (cont.)

Tips uncover 43% of frauds and are the leading reason frauds are detected.

Internal audits uncover 15% of frauds and are in second place.

External audits uncover only 4% of frauds and are in seventh place.

Augmenting the traditional auditing approaches with techniques from data science and statistics may help prevent or detect fraud.

Today's Goals

We will explore Benford's Law and an anomaly detection algorithm in detail.

You will leave understanding when Benford's Law should and should not be expected to hold.

We will also use a publicly available dataset with sales transaction from an online retailer to identify potential errors and anomalies.

You will walk away seeing a systematic approach to summarizing, aggregating, and exploring the entire population rather than relying on sampling.

Benford's Law History

A logarithm is the inverse of raising a number to a power. Logarithms have a base, a number, and then the solution to the equation.

The base is the number that is being raised to a power, the number is the input to the function, and the solution is power the base is raised to.

The log in base 10 of 100 is 2 because we have to raise 10 to the power of 2 to get 100.

$$\log_{10}(100) = 2$$

$$10^2 = 100$$

$$\ln(100) = 4.61$$

$$e^{4.61} = 100$$

Benford's Law History (cont.)

Scientists and mathematicians had books of logarithm tables so that they could quickly look up answers to logarithmic equations.

Simon Newcomb saw that the books were dirtier and more worn for numbers that started with smaller first digits than larger first digits.

This indicated that people were typically trying to find solutions to numbers with leading 1's or 2's as opposed to 8's or 9's.

Newcomb suggested that the probability of a naturally occurring number between 1 and 9 being observed was

$$\log_{10}(N+1) - \log_{10}(N) = \log_{10}(1 + 1/N)$$

Benford's Law Distribution

Below are the expected frequencies for leading digits of variables that follow Benford's Law

Leading Digit	Expectation
1	30.1%
2	17.6%
3	12.5%
4	9.7%
5	7.9%
6	6.7%
7	5.8%
8	5.1%
9	4.6%

Benford's Law Distribution (cont.)

In a paper published in 1938, Frank Benford found that the above expected frequencies accurately described:

- Numbers in books and journals
- Street addresses
- Populations of cities
- Surface areas of rivers
- Death rates

Later work has found that Benford's Law also applies to:

- Electricity bills
- Stock prices
- Consumer prices

Benford's Law Distribution (cont.)

Benford's Law is more likely to hold when:

The variable spans multiple orders of magnitude (i.e., the minimum and maximum values are far apart)

The variable follows a power law distribution (e.g., the 80-20 "Pareto Principle" applies)

The variable comes from a distribution that is "skewed" or has a "fat tail"(e.g., the mean is greater than the median)

The variable is a product or other mathematical combination of other variables (e.g., invoice total = price x quantity)

Benford's Law Distribution (cont.)

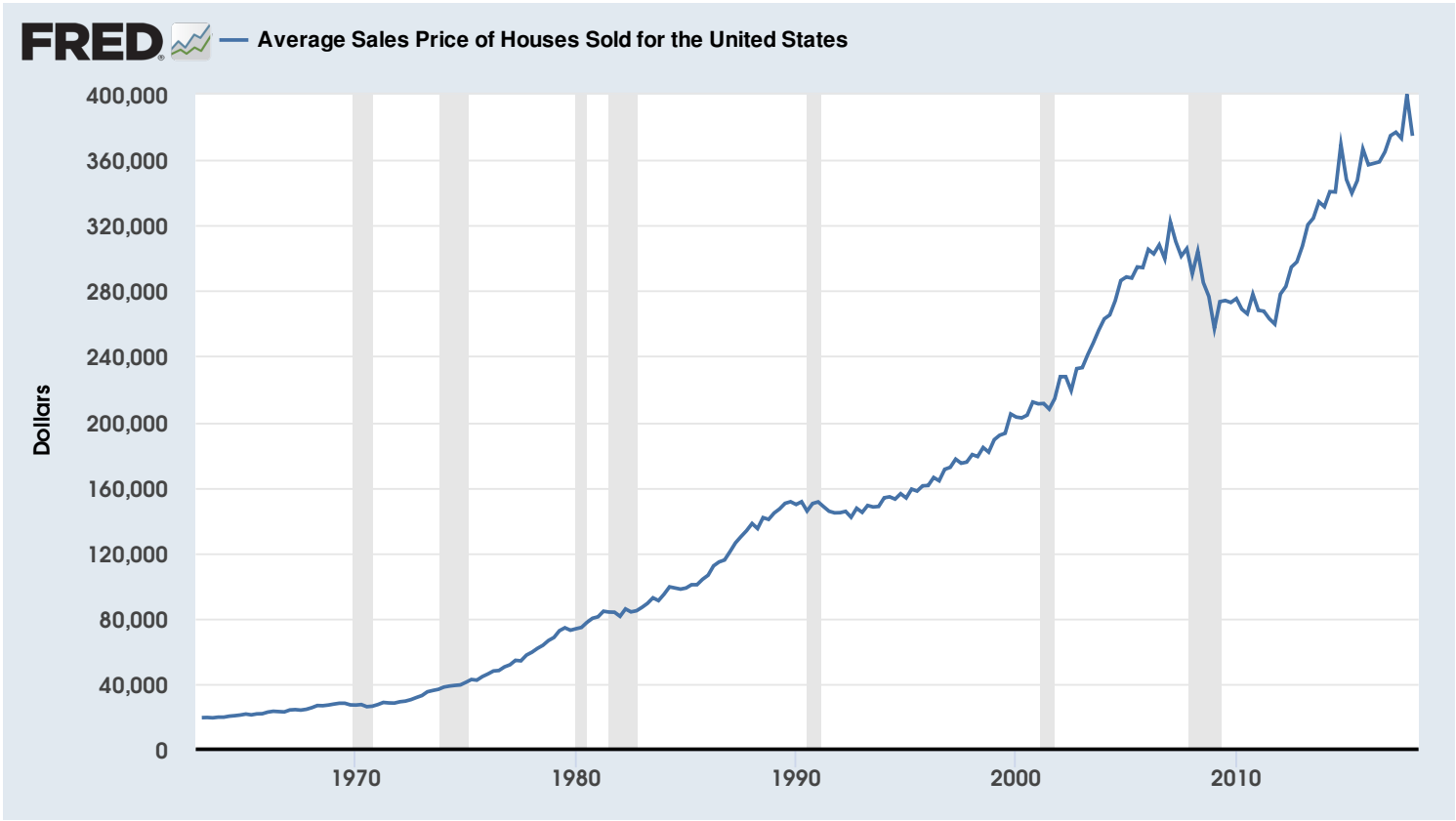
Benford's Law **is <u>not</u>** expected to hold when:

The variable is normally distributed.

The minimum and the maximum of the variable are relatively close together.

The variable is a sequentially generated number with a limited range (e.g., check numbers, invoices)

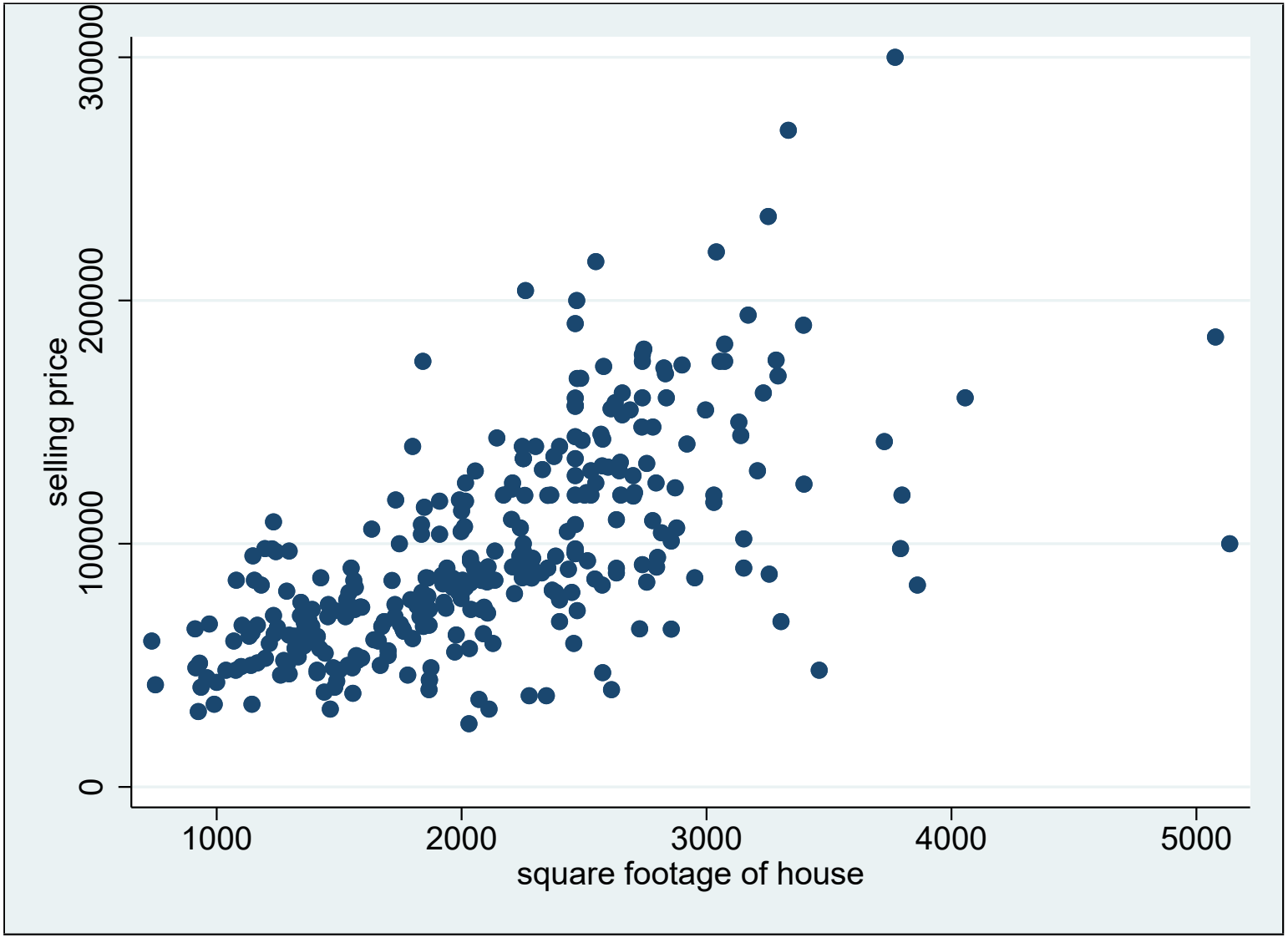
Some Useful Statistics- House Prices



Some Useful Statistics- House Prices (cont.)

Statistic	Excel Function	Measure of	What It Tells You	Price
Mean	=AVERAGE()	Central Tendency	This is the expected value of the distribution. It is calculated as an equally-weighted sum of all the values of price in the dataset.	$96,100.66
Median	=MEDIAN()	Central Tendency	Similar to the Mean but less influenced by outliers. The value of price where half of the observations in the data are larger than and half are smaller than.	$85,900.00
Variance	=VAR.S()	Dispersion	The sum of the difference between each house's price and the Mean, squared. This is harder to interpret than the Standard Deviation because the units are dollars, squared.	$1,868,290,835.11
Standard Deviation	=STDEV.S()	Dispersion	The square root of the Variance. A measure of dispersion from the Mean in non-squared dollars which is usually easier for people to interpret than the Variance.	$43,223.73
Covariance	=COVARIANCE.S()	Association	Measure of the linear relationship between two variables (e.g., price and area). Tells you if, on average, when one variable is above its Mean if the other variable is also above its Mean.	$19,385,134.69
Correlation	=CORREL()	Association	Measure of the linear relationship between two variables (e.g. price and area) bounded between -1 and +1. Calculated by the Covariance of the two variables divided by the product of each variable's Standard Deviation.	0.65

Some Useful Statistics- House Prices (cont.)



Some Useful Statistics- House Prices (cont.)

Statistic	Excel Function	Measure of	What It Tells You	Price
Mean	=AVERAGE()	Central Tendency	This is the expected value of the distribution. It is calculated as an equally-weighted sum of all the values of price in the dataset.	$96,100.66
Median	=MEDIAN()	Central Tendency	Similar to the Mean but less influenced by outliers. The value of price where half of the observations in the data are larger than and half are smaller than.	$85,900.00
Variance	=VAR.S()	Dispersion	The sum of the difference between each house's price and the Mean, squared. This is harder to interpret than the Standard Deviation because the units are dollars, squared.	$1,868,290,835.11
Standard Deviation	=STDEV.S()	Dispersion	The square root of the Variance. A measure of dispersion from the Mean in non-squared dollars which is usually easier for people to interpret than the Variance.	$43,223.73
Covariance	=COVARIANCE.S()	Association	Measure of the linear relationship between two variables (e.g., price and area). Tells you if, on average, when one variable is above its Mean if the other variable is also above its Mean.	$19,385,134.69
Correlation	=CORREL()	Association	Measure of the linear relationship between two variables (e.g. price and area) bounded between -1 and +1. Calculated by the Covariance of the two variables divided by the product of each variable's Standard Deviation.	0.65

Distributions Explained

Events and outcomes may be thought of as "random variables" which are described by "distributions":

A company's revenue next year.

The number of touchdowns the Broncos will score.

The number of internal control deficiencies found during an audit.

Outcomes of coin flips and dice rolls.

The average error in the population of an account balance or significant class of transactions based on a sample.

Distributions Explained (cont.)

Two functions of the distributions of a random variable are the probability density or mass function (PDF or PMF) and the cumulative density function (CDF).

Mathematically, the CDF is found by integrating the PDF, and this means the PDF is the derivative of the CDF.

The reason these mathematical functions are important is that **if you know how a variable is distributed, you can calculate probabilities of outcomes**.

Distributions Explained (cont.)

Think of the PDF as a graph with the possible outcomes of the variable on the x axis and the relative frequency of the outcome on the y axis.

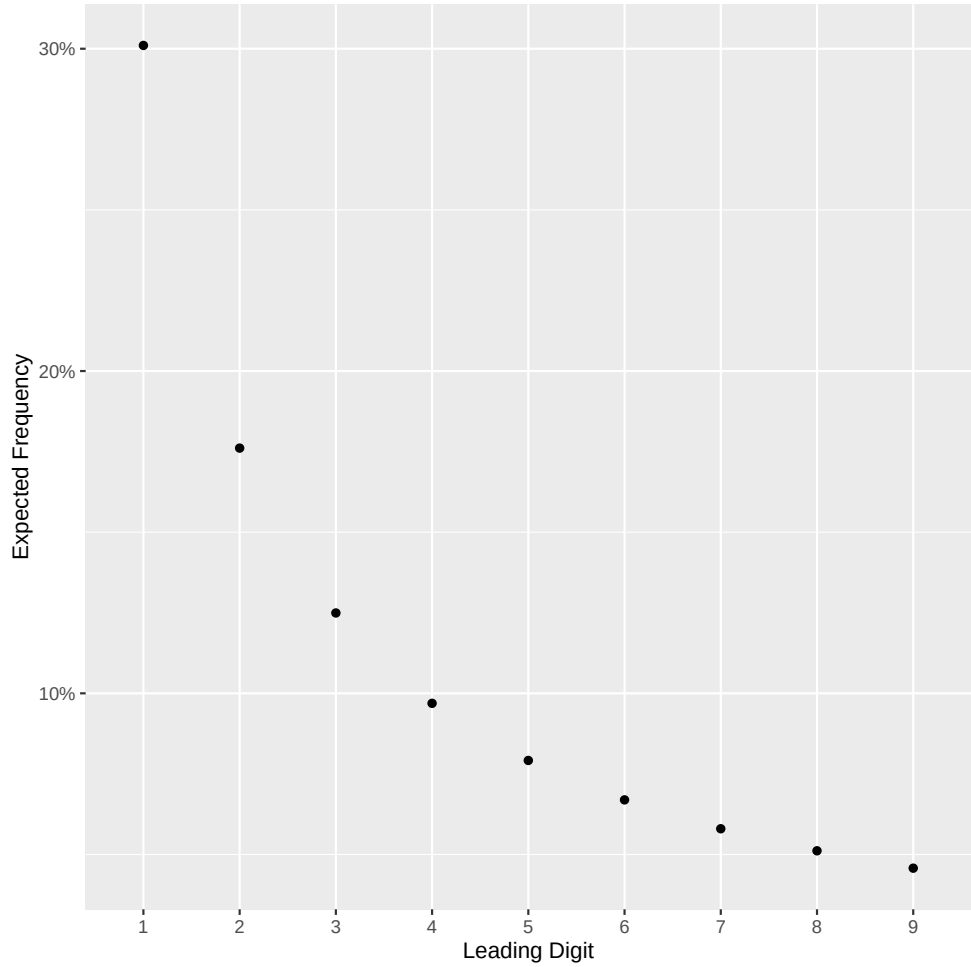
Think of the CDF as a graph with the possible outcomes of the variable on the x axis and the **cumulative** relative frequency of the outcome on the y axis.

The equation you saw earlier for Benford's Law

$$\log_{10}(N+1) - \log_{10}(N) = \log_{10}(1 + 1/N)$$

is a probability density function (PDF).

Distributions Explained (cont.)



The equation $\log_{10}(1 + 1/N)$ calculates the above frequencies for each number, and the sum across all numbers equals 100%.

Distributions Explained (cont.)

Using the PDF, the most likely outcome for a leading digit of a random variable that follows Benford's Law is a 1.

And the probability of observing a leading 1 is 30%.

Using the CDF, the probability of observing a leading 1 or 2 for the first digit is 30.1% + 17.6% = 47.7%

The Normal Distribution

Consider a random variable that is normally distributed.

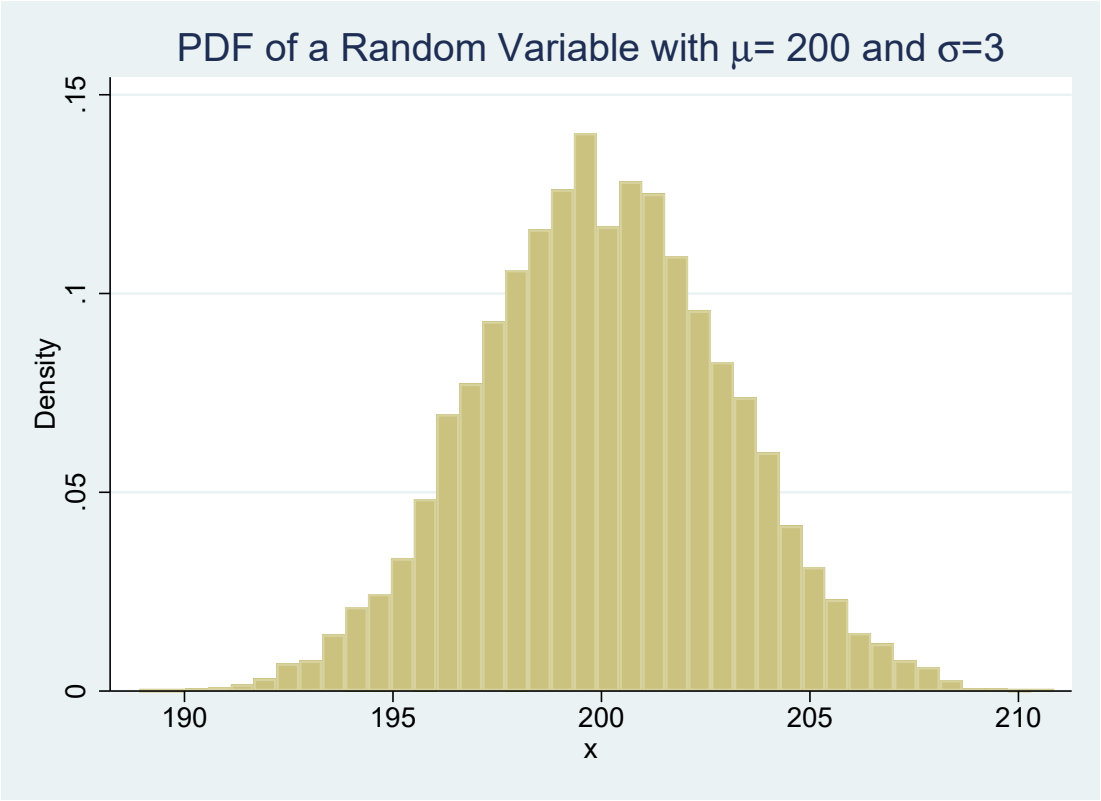
A powerful feature of the normal distribution is that knowing the mean μ and standard deviation σ of the variable allows one to quickly calculate probabilities.

68% of observations will lie within $\mu \pm \sigma$

95% of observations will lie within $\mu \pm 2\sigma$

99.7% of observations will lie within $\mu \pm 3\sigma$

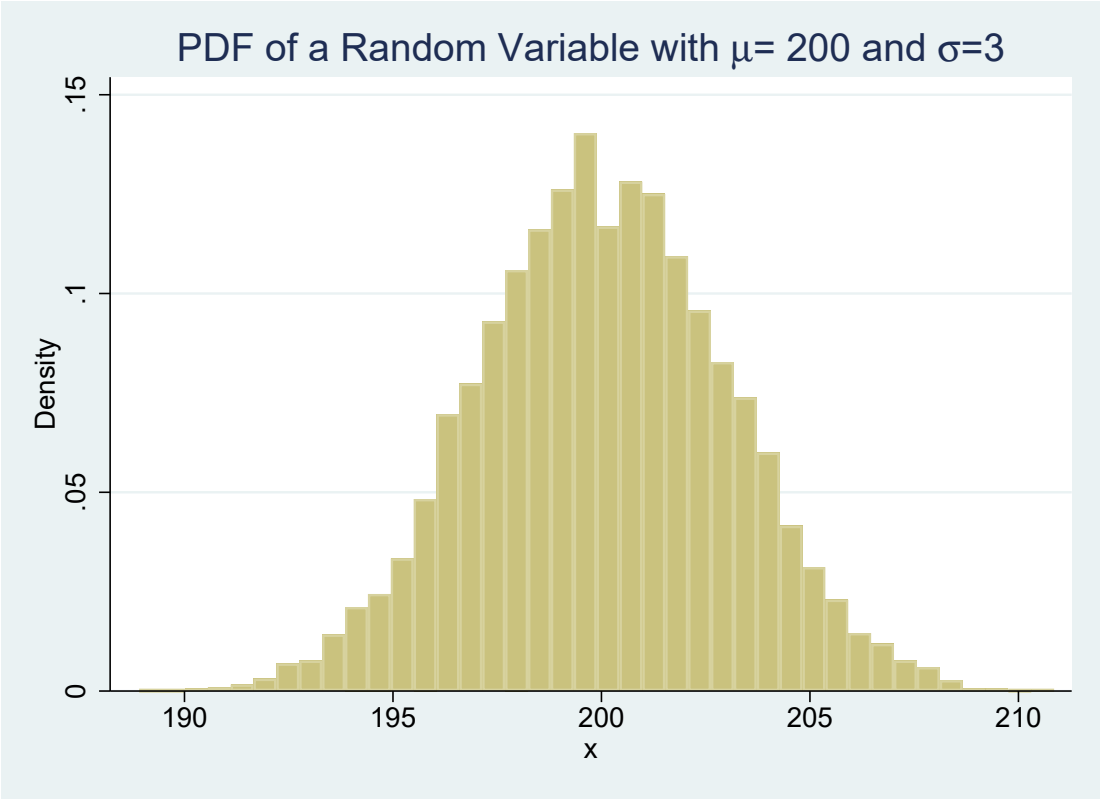
Normal Distribution Example



Think about where the mean plus or minus 1, 2, or 3 times the standard deviation would place you on the x axis.

Now consider the area under the curve near those points.

Normal Distribution Example

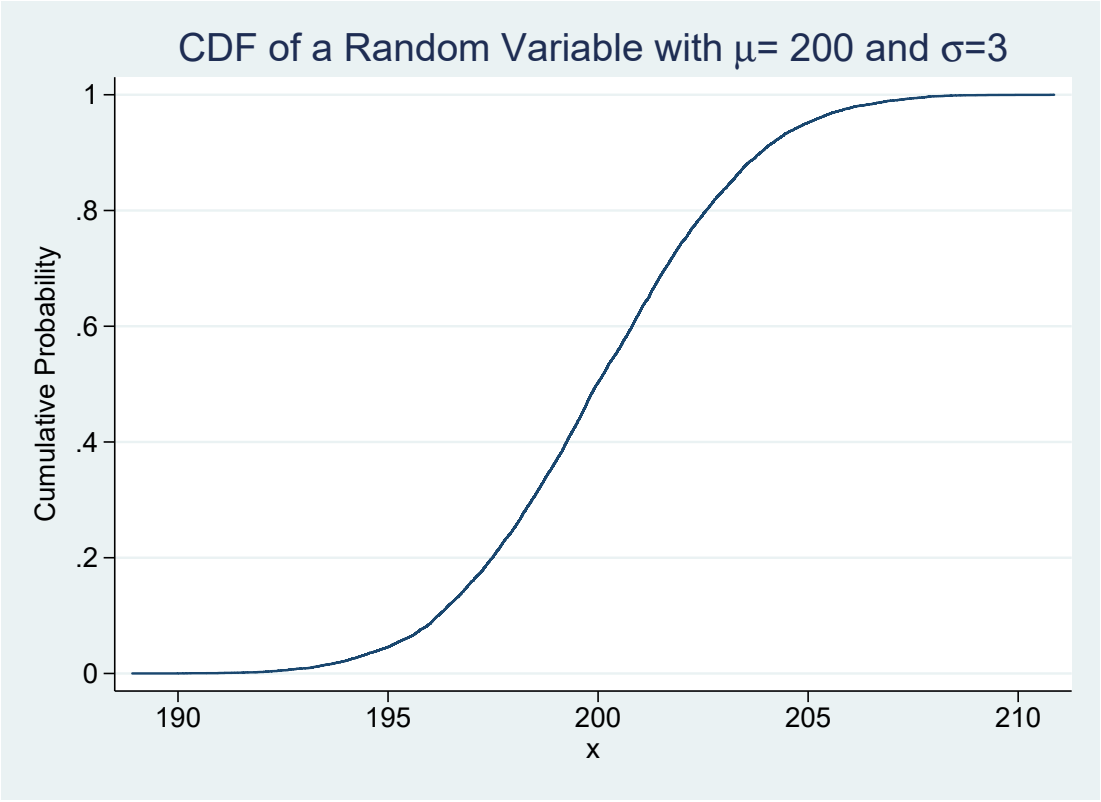


We expect that 68% of the observations are between 197 and 203.

We expect that 95% of the observations are between 194 and 206.

We expect that 99.7% of the observations are between 191 and 209.

Normal Distribution Example (cont.)



The normal distribution is symmetric and half of the observations lie above the mean and half life below the mean.

Can you see that 50% cumulative probability on the y axis at the mean of 200?

Z Scores

If a variable is normally distributed, it can be transformed into a "standard normal" variable by subtracting the mean of the variable from all observations and then dividing by the standard deviation.

This converts variables into *z scores* where

$$z = (X_i - \mu)/\sigma$$

and z now has a mean of zero and a variance of 1.

This is called **standardization** tells you how many standard deviations away from the mean the observation is.

Z Scores (cont.)

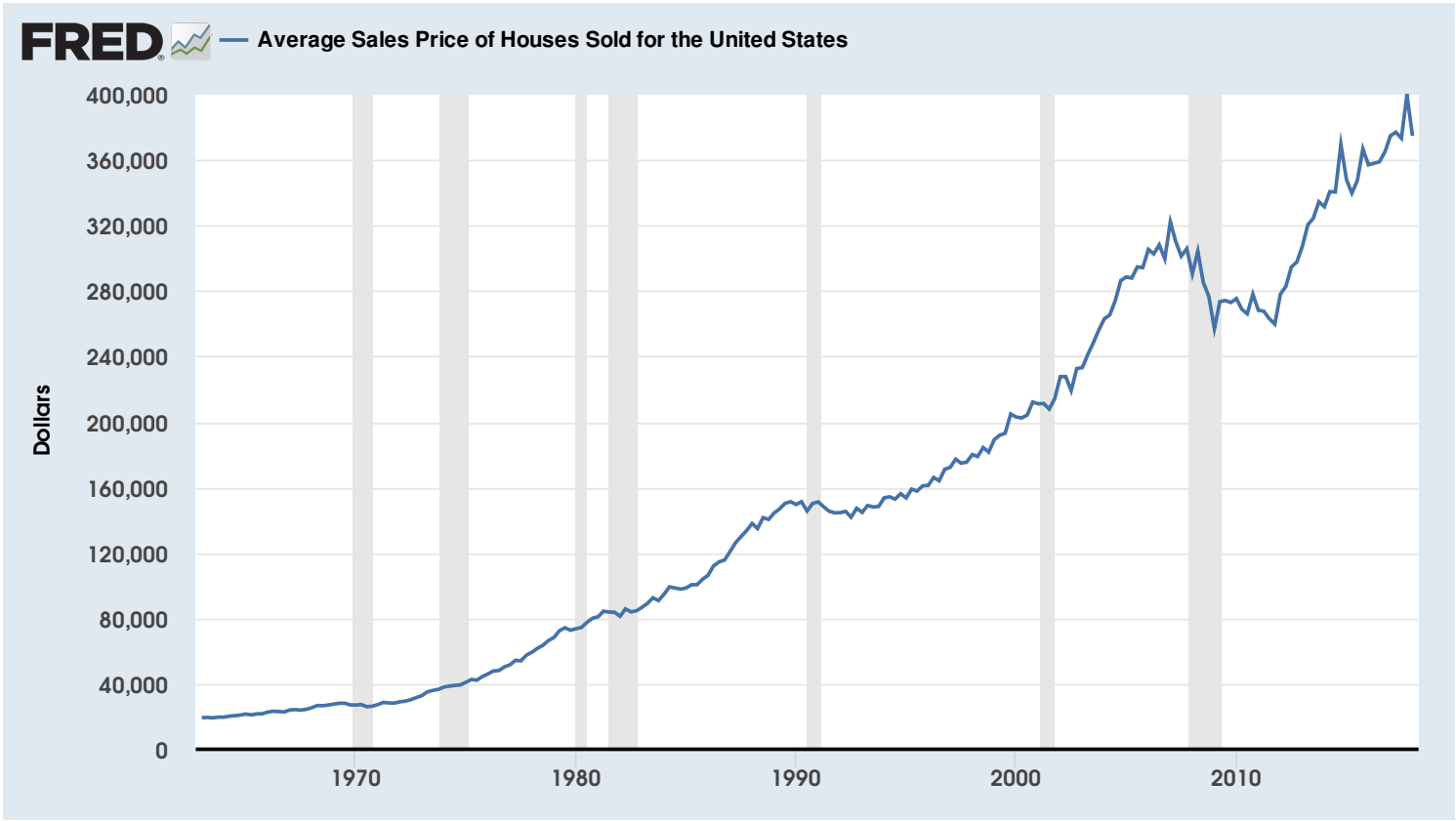
We saw previously that for a normally distributed variable:

95% of observations will lie within $\mu \pm 2\sigma$

99.7% of observations will lie within $\mu \pm 3\sigma$

This means that z scores with values larger than the absolute value of 2 should be rare in our dataset.

Back to House Prices



Back to House Prices (cont.)

Statistic	Excel Function	Measure of	What It Tells You	Price
Mean	=AVERAGE()	Central Tendency	This is the expected value of the distribution. It is calculated as an equally-weighted sum of all the values of price in the dataset.	$96,100.66
Median	=MEDIAN()	Central Tendency	Similar to the Mean but less influenced by outliers. The value of price where half of the observations in the data are larger than and half are smaller than.	$85,900.00
Variance	=VAR.S()	Dispersion	The sum of the difference between each house's price and the Mean, squared. This is harder to interpret than the Standard Deviation because the units are dollars, squared.	$1,868,290,835.11
Standard Deviation	=STDEV.S()	Dispersion	The square root of the Variance. A measure of dispersion from the Mean in non-squared dollars which is usually easier for people to interpret than the Variance.	$43,223.73
Covariance	=COVARIANCE.S()	Association	Measure of the linear relationship between two variables (e.g., price and area). Tells you if, on average, when one variable is above its Mean if the other variable is also above its Mean.	$19,385,134.69
Correlation	=CORREL()	Association	Measure of the linear relationship between two variables (e.g. price and area) bounded between -1 and +1. Calculated by the Covariance of the two variables divided by the product of each variable's Standard Deviation.	0.65

Z Scores in Excel

	A	B	C	D
1		Data	z_Score_Manual	z_Score
2		0.4725762	-0.031640561	-0.031640561
3		0.498445	-0.01888419	-0.01888419
4		0.4848192	-0.025603302	-0.025603302
5		0.8901567	0.174276251	0.174276251
1998		0.236188	-0.148208053	-0.148208053
1999		7	3.187155077	3.187155077
2000		90	44.11601863	44.11601863
2001				
2002	Mean	0.5367404		
2003	Std. Dev.	2.0279087		
2004				

	A	B	C	D
1		Data	z_Score_Manual	z_Score
2		=RAND()	=(B2-$B$2002)/$B$2003	=STANDARDIZE(B2,$B$2002, $B$2003)
3		=RAND()	=(B3-$B$2002)/$B$2003	=STANDARDIZE(B3,$B$2002, $B$2003)
4		=RAND()	=(B4-$B$2002)/$B$2003	=STANDARDIZE(B4,$B$2002, $B$2003)
5		=RAND()	=(B5-$B$2002)/$B$2003	=STANDARDIZE(B5,$B$2002, $B$2003)
1998		=RAND()	=(B1998-$B$2002)/$B$2003	=STANDARDIZE(B1998,$B$2002, $B$2003)
1999		-8	=(B1999-$B$2002)/$B$2003	=STANDARDIZE(B1999,$B$2002, $B$2003)
2000		90	=(B2000-$B$2002)/$B$2003	=STANDARDIZE(B2000,$B$2002, $B$2003)
2001				
2002	Mean	=AVERAGE(B2:B2000)		
2003	Std. Dev.	=STDEV.S(B2:B2000)		
2004				

Z Scores in Excel (cont.)

	A	B	C	D
1		Data	z Score Manual	z Score
2		0.2828278	-0.125758539	-0.125758539
3		0.8578509	0.157273629	0.157273629
4		0.6916208	0.075453522	0.075453522
5		0.661476	0.060615931	0.060615931
1998		0.8648971	0.160741866	0.160741866
1999		-8	-4.202649816	-4.202649816
2000		90	44.03393747	44.03393747
2001				
2002	Mean	0.5383255		
2003	Std. Dev.	2.0316528		
2004				

Z scores of observations larger than the mean are positive, and z scores of observations less than the mean are negative.

Z Scores in Excel (cont.)

	A	B	C	D
1		Data	z_Score_Manual	z_Score
2		0.5586269	0.008976279	0.008976279
3		0.0312783	-0.251151198	-0.251151198
4		0.5382624	-0.001068965	-0.001068965
5		0.4014141	-0.06857271	-0.06857271
1998		0.5449957	0.002252386	0.002252386
1999		7	3.186339748	3.186339748
2000		90	44.12810206	44.12810206
2001				
2002	Mean	0.5404295	0.00	0.00
2003	Std. Dev.	2.0272698	1	1

The *z scores* we created have a mean of 0 and a standard deviation of 1.

If the data is normally distributed *z scores* of 2 or larger in absolute value should occur around 5% of the time.

If the data is normally distributed *z scores* of 3 or larger in absolute value should occur around 0.3% of the time.

So... How Do We Use All of This?

Deviations from Benford's Law may alert the auditor that additional audit procedures are necessary.

Sample sizes could be increased in specific leading digit buckets that severely violate Bendford's Law.

The absolute value of *z scores* larger than 3 could be flagged as anomalies that warrant testing.

This is exactly what a popular unsupervised anomaly detection algorithm does.

Question

Consider the final grades in an undergraduate business school course.

Do you think the final grades will follow Benford's Law?

Question (cont.)

Final grades would follow the below distribution:

Leading Digit	Expectation
1	30.1%
2	17.6%
3	12.5%
4	9.7%
5	7.9%
6	6.7%
7	5.8%
8	5.1%
9	4.6%

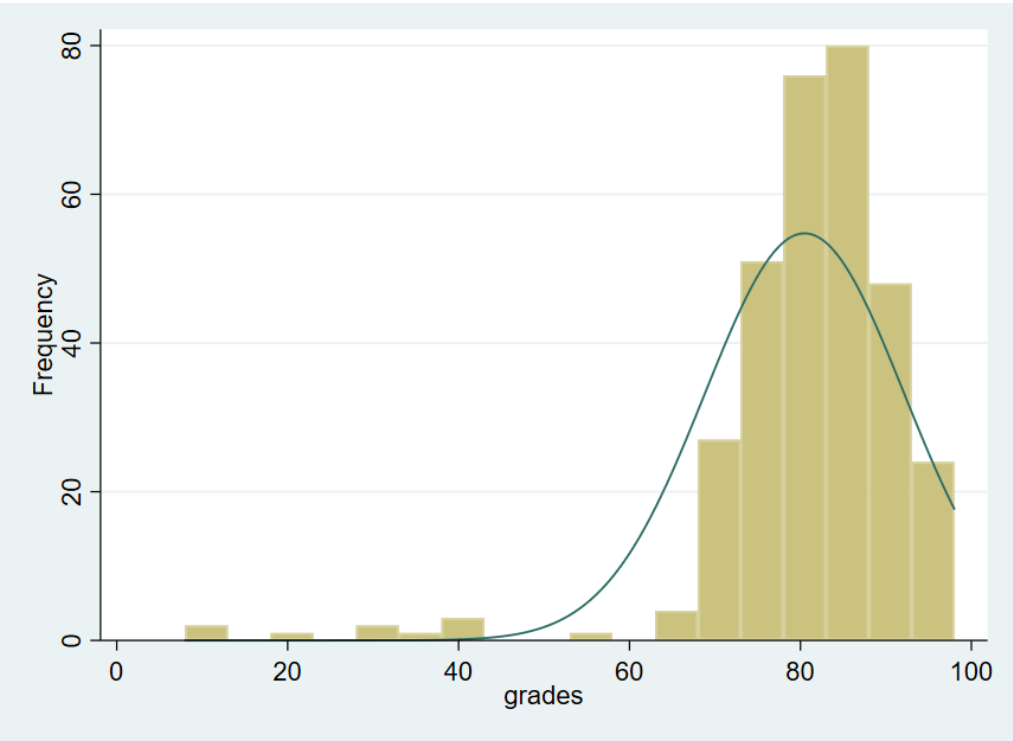
Is it reasonable to assume that 30% of final grades will start with a 1?

This would only happen if 30% of students' final grades were between 10 and 19 or greater than or equal to 100!

Additionally, only 16% of students would have B's, C's, or sub-100 A's.

Question (cont.)

```
count    320.00
mean      80.48
std       11.66
min        8.00
25%       76.75
50%       82.00
75%       87.00
max       95.00
Name: grades, dtype: float64
```



Question (cont.)

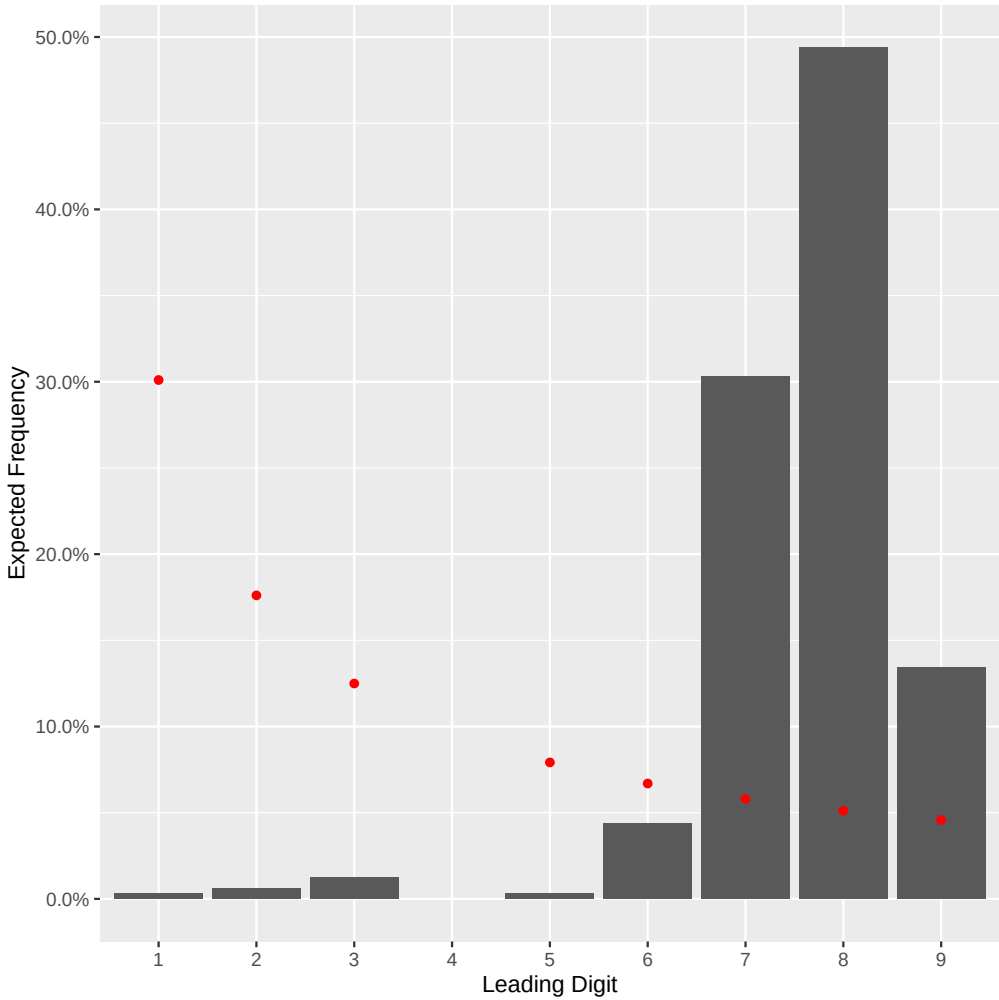
The minimum final grade is 8 and the maximum is 95.

The distribution is skewed to the left as evidenced by the mean of 80 and the median of 82, and a true normal distribution would have grades larger than 100.

But the normal distribution doesn't look completely out of question for a rough approximation of the final grades.

The above indicate that we **should not** expect grades to follow Benford's Law.

Question (cont.)



Question (cont.)

The expected vs. actual frequencies are below:

Leading Digit	Expectation	Observed
1	30.1%	0.3%
2	17.6%	0.6%
3	12.5%	1.3%
4	9.7%	0.0%
5	7.9%	0.3%
6	6.7%	4.3%
7	5.8%	30.3%
8	5.1%	49.3%
9	4.6%	13.4%

How Did We Get There?

Auditing software packages typically contain the capability to analyze a series of numbers against the Benford's Law expectations.

These packages are also able to conduct a variety of other analyses, but today we will use Python and conduct each analysis "by hand."

This is an auditing package agnostic approach to carefully thinking through the analyses and building them from the ground up.

The goal is show you what's possible with a tool that may be new to you while reinforcing the high level concepts with detailed applications.

Import The Grades Data With Python

```
# Read in the Excel data with grades.
df= pd.read_excel('grades.xlsx')

# Display the first ten observations in the dataset.
df.head(10)
```

	grades
0	94
1	94
2	93
3	92
4	91
5	90
6	90
7	89
8	89
9	89

Grades- Summary Statistics

```
# Examine summary statistics for grades.
df['grades'].describe().round(2)
```

```
count    320.00
mean      80.48
std       11.66
min        8.00
25%       76.75
50%       82.00
75%       87.00
max       95.00
Name: grades, dtype: float64
```

Strip Off The Leading Digit

```
# Create a variable with the lead digit of each grade.
df['lead'] = df['grades'].astype(str).str[0]

# Examine the modified dataset that now includes the first digit of each grade.
df.head(10)
```

	grades	lead
0	94	9
1	94	9
2	93	9
3	92	9
4	91	9
5	90	9
6	90	9
7	89	8
8	89	8
9	89	8

Calculate Leading Digit Frequencies

```
# First, get the counts of each leading digit.
df['lead'].value_counts()
```

```
8    158
7     97
9     43
6     14
3      4
2      2
1      1
5      1
Name: lead, dtype: int64
```

```
# Next, return the percentage of the above counts.
df['lead'].value_counts(normalize= True)
```

```
8    0.493750
7    0.303125
9    0.134375
6    0.043750
3    0.012500
2    0.006250
1    0.003125
5    0.003125
Name: lead, dtype: float64
```

Calculate Benford's Law Frequencies

```
def benfords(x):
    return np.log10(1 + 1/x)

df_lead_grades['percent expected'] = df_lead_grades['first digit'].apply(benfords)
```

The frequency expectations for each leading digit are calculated using:

$$\log_{10}(N+1) - \log_{10}(N) = \log_{10}(1 + 1/N)$$

Above we define a function named "benfords" which applies the above equation to each leading digit 1, 2, ... 8, 9.

Dataset Used For Graphics

df_lead_grades

	first digit	lead	percent expected
0	8	0.493750	0.051153
1	7	0.303125	0.057992
2	9	0.134375	0.045757
3	6	0.043750	0.066947
4	3	0.012500	0.124939
5	2	0.006250	0.176091
6	5	0.003125	0.079181
7	1	0.003125	0.301030

The above dataset contains everything needed to create a visualization:

- Each leading digit
- The percentage predicted by Benford's Law
- The percentage observed in the data

Application With E-Commerce Data

Kaggle, a subsidiary of Google, is a website that hosts machine learning competitions.

Joining Kaggle is free, users can download datasets, upload their analyses, and see analyses conducted by others.

We will examine a Kaggle dataset with sales data from a retailer.

The dataset is available below:

https://www.kaggle.com/carrie1/ecommerce-data/

First Ten Observations

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047.0	United Kingdom

First Ten Observations (cont.)

Caution is warranted when only looking at a sequential block near the beginning or end of a dataset, but:

- UnitPrices appear fairly low.
- The same customer initiated two transactions two minutes apart.
- We could split InvoiceDate into separate **date** and **time** variables if needed.
- Quantity appears fairly small.
- Aggregation at the InvoiceNo level may be useful.

First Ten Observations (cont.)

The Sales Dataset Contains:

- Invoice Numbers
- SKUs
- Product Descriptions
- Quantities Ordered
- Transaction Date
- Prices
- A Customer Identifier
- Customer's Country

The Sales Dataset Does **Not** Contain:

- Extensions of Price and Quantity (e.g., Price x Quantity)
- Invoice Totals

Dataset Overview

```
# Get a general idea about what the dataframe looks like
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

```
# Examine how many missing values there are in each column
for i in df.columns:
    print(i, df[i].isnull().sum())
```

```
InvoiceNo 0
StockCode 0
Description 1454
Quantity 0
InvoiceDate 0
UnitPrice 0
CustomerID 135080
Country 0
```

Integers vs. Floats

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047.0	United Kingdom

Dataset Overview (cont.)

```
# There are many missing values for CustomerID. Next, calculate this as a percentage
df['CustomerID'].isnull().sum()/ len(df) 0.249266943342886
```

The code above is:

- Referring to the CustomerID column in the dataset.
- Creating a Boolean series of 1's or 0's based on whether each CustomerID is a null value or not.
- Calculating the sum of the 1's from the prior step (i.e., adding up each time the logical condition is met).
- Dividing the sum of the 1's by the total number of rows in the dataset.

Questions

Is it reasonable that 25% of transactions do not have a customer identifier?

Does this indicate a problem with the data we have received? Has the data been garbled in some way?

Does this point to a problem with the company's internal controls?

Could this indicate a problem with the client's POS system?

Dataset Overview (cont.)

```
# Get a general idea about what the dataframe looks like
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

We are using Python, but the specific open source library we are using for analysis is called "pandas."

In the pandas library, if a column has the same datatype (e.g., numeric vs. string) the column will be imported as that datatype.

Dataset Overview (cont.)

```
# Get a general idea about what the dataframe looks like
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    541909 non-null  object 
 1   StockCode    541909 non-null  object 
 2   Description  540455 non-null  object 
 3   Quantity     541909 non-null  int64  
 4   InvoiceDate  541909 non-null  object 
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

You can see this with Quantity, which is an integer and with UnitPrice and CustomerID, which are floats.

Both of the above are numeric datatypes. This means mathematical operations will work on these variables.

Dataset Overview (cont.)

```
# Get a general idea about what the dataframe looks like
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

InvoiceNo, StockCode, Description, InvoiceDate, and Country are "objects."

This means these variables have been imported as strings (i.e., text) and mathematical operations **will not** work on these variables.

Quantity and UnitPrice

```
df[['Quantity', 'UnitPrice']].describe()
```

	Quantity	UnitPrice
count	541909.00	541909.00
mean	9.55	4.61
std	218.08	96.76
min	-80995.00	-11062.06
25%	1.00	1.25
50%	3.00	2.08
75%	10.00	4.13
max	80995.00	38970.00

Above contains descriptive statistics for Quantity and UnitPrice. We have:

The count, average, standard deviation, minimum, quartiles, and the maximum.

Note that the 50% is the median.

Quantity and UnitPrice: Quantity

Quantity contains negative values.

The average Quantity is 9.55, the 75th percentile is 10, but the maximum is 80,995.

If a company sells around 10 units on average, does it seem reasonable for a sale with 80,995 units?

The maximum value is the same as the largest negative value which could point to netting or backing out in the dataset.

Booking and then backing out a transaction of that magnitude may also lead to further internal control questions.

Quantity and UnitPrice: UnitPrice

UnitPrice also contains negative values.

The average UnitPrice is 4.61, but the maximum is 38,907.

Depending on the type of retailer, a price that large may be an obvious error or this could be expected.

If prices vary that much then this would need to be incorporated into the design of substantive analytical procedures.

Using the average price is unlikely to result in a substantive analytical procedure that is precise enough to be useful.

Negative UnitPrices

```
df['UnitPrice'][df['UnitPrice'] < 0].count() 2
df[df['UnitPrice'] < 0]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
299983	A563186	B	Adjust bad debt	1	8/12/2011 14:51	-11062.06	nan	United Kingdom
299984	A563187	B	Adjust bad debt	1	8/12/2011 14:52	-11062.06	nan	United Kingdom

Above we can see that there are two observations in the dataset with negative prices.

These are related to bad debt expense, and neither has an associated CustomerID.

Both of the InvoiceNo's have a leading letter, and the numbers do not repeat even though the entries are for identical amounts.

Zero UnitPrices

```
df['UnitPrice'][df['UnitPrice']==0].count() 2515
df[df['UnitPrice']==0][:10]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	12/1/2010 11:52	0.00	nan	United Kingdom
1970	536545	21134	NaN	1	12/1/2010 14:32	0.00	nan	United Kingdom
1971	536546	22145	NaN	1	12/1/2010 14:33	0.00	nan	United Kingdom
1972	536547	37509	NaN	1	12/1/2010 14:33	0.00	nan	United Kingdom
1987	536549	85226A	NaN	1	12/1/2010 14:34	0.00	nan	United Kingdom
1988	536550	85044	NaN	1	12/1/2010 14:34	0.00	nan	United Kingdom
2024	536552	20950	NaN	1	12/1/2010 14:34	0.00	nan	United Kingdom
2025	536553	37461	NaN	3	12/1/2010 14:35	0.00	nan	United Kingdom
2026	536554	84670	NaN	23	12/1/2010 14:35	0.00	nan	United Kingdom
2406	536589	21777	NaN	-10	12/1/2010 16:50	0.00	nan	United Kingdom

Checking for UnitPrices equal to zero yields the results above.

UnitPrices of 0 may make sense based on what you know about client, or these could be errors.

None of the first ten observations with UnitPrices of zero have CustomerID's and this may or not may not be a problem.

Zero UnitPrices (cont.)

```
df[df['UnitPrice'] == 0][-10:]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
535333	581210	23395	check	-26	12/7/2011 18:36	0.00	nan	United Kingdom
535334	581211	22142	check	14	12/7/2011 18:36	0.00	nan	United Kingdom
535335	581212	22578	lost	-1050	12/7/2011 18:38	0.00	nan	United Kingdom
535336	581213	22576	check	-30	12/7/2011 18:38	0.00	nan	United Kingdom
536908	581226	23090	missing	-338	12/8/2011 9:56	0.00	nan	United Kingdom
536981	581234	72817	NaN	27	12/8/2011 10:33	0.00	nan	United Kingdom
538504	581406	46000M	POLYESTER FILLER PAD 45x45cm	240	12/8/2011 13:58	0.00	nan	United Kingdom
538505	581406	46000S	POLYESTER FILLER PAD 40x40cm	300	12/8/2011 13:58	0.00	nan	United Kingdom
538554	581408	85175	NaN	20	12/8/2011 14:06	0.00	nan	United Kingdom
538919	581422	23169	smashed	-235	12/8/2011 15:24	0.00	nan	United Kingdom

We can also examine the last ten observations with UnitPrices of zero.

CustomerID is still missing in these observations.

The descriptions include "check," "lost," "missing," and "smashed."

Negative Quantities are also observed, but we already knew those existed in the data.

Zero UnitPrices (cont.)

```
df['CustomerID'][df['UnitPrice'] == 0].isna().sum()  2475
```

We can also check to see how often UnitPrices of zero are associated with missing CustomerID's.

Above you can see that this pattern holds in 2,475 out of the 2,515 observations.

These results may help guide discussions with management or within the audit engagement team.

Largest UnitPrices

```
df[df['UnitPrice'] > 10000]
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
15016	C537630	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:04	13541.33	nan	United Kingdom
15017	537632	AMAZONFEE	AMAZON FEE	1	12/7/2010 15:08	13541.33	nan	United Kingdom
16232	C537644	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:34	13474.79	nan	United Kingdom
16356	C537651	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:49	13541.33	nan	United Kingdom
43702	C540117	AMAZONFEE	AMAZON FEE	-1	1/5/2011 9:55	16888.02	nan	United Kingdom
43703	C540118	AMAZONFEE	AMAZON FEE	-1	1/5/2011 9:57	16453.71	nan	United Kingdom
222681	C556445	M	Manual	-1	6/10/2011 15:31	38970.00	15098.00	United Kingdom
299982	A563185	B	Adjust bad debt	1	8/12/2011 14:50	11062.06	nan	United Kingdom
524601	C580604	AMAZONFEE	AMAZON FEE	-1	12/5/2011 11:35	11586.50	nan	United Kingdom
524602	C580605	AMAZONFEE	AMAZON FEE	-1	12/5/2011 11:36	17836.46	nan	United Kingdom

UnitPrices greater than 10,000 are shown above.

The data we have is supposed to be sales data so it is strange to see charges from Amazon.

One of the Amazon Fees has a positive quantity and no leading alpha character for the InvoiceNo.

Largest UnitPrices (cont.)

```
df[df['UnitPrice'] > 10000]
```

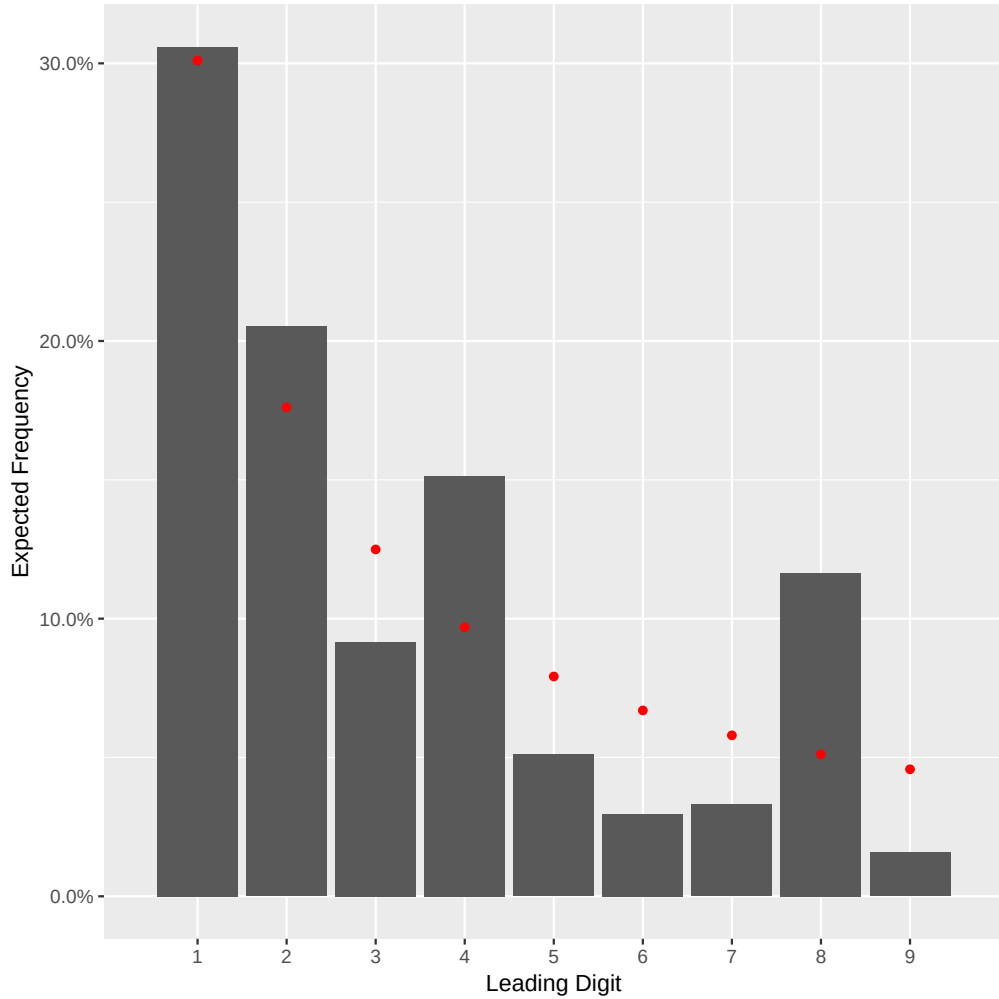
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
15016	C537630	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:04	13541.33	nan	United Kingdom
15017	537632	AMAZONFEE	AMAZON FEE	1	12/7/2010 15:08	13541.33	nan	United Kingdom
16232	C537644	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:34	13474.79	nan	United Kingdom
16356	C537651	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:49	13541.33	nan	United Kingdom
43702	C540117	AMAZONFEE	AMAZON FEE	-1	1/5/2011 9:55	16888.02	nan	United Kingdom
43703	C540118	AMAZONFEE	AMAZON FEE	-1	1/5/2011 9:57	16453.71	nan	United Kingdom
222681	C556445	M	Manual	-1	6/10/2011 15:31	38970.00	15098.00	United Kingdom
299982	A563185	B	Adjust bad debt	1	8/12/2011 14:50	11062.06	nan	United Kingdom
524601	C580604	AMAZONFEE	AMAZON FEE	-1	12/5/2011 11:35	11586.50	nan	United Kingdom
524602	C580605	AMAZONFEE	AMAZON FEE	-1	12/5/2011 11:36	17836.46	nan	United Kingdom

One of the bad debt adjustment entries we saw earlier for -11,062.06 has been backed out.

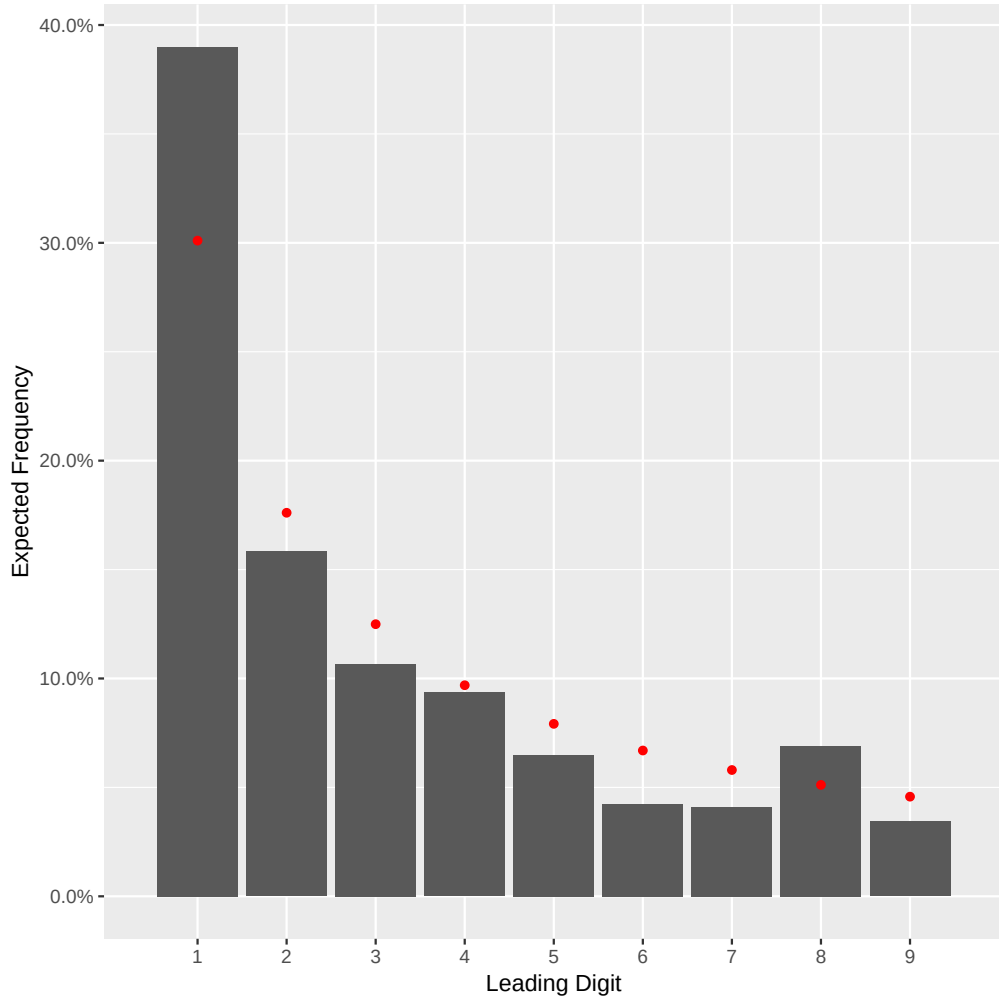
There is also a "Manual" entry.

These results could also result in questions for management or altering the audit plan.

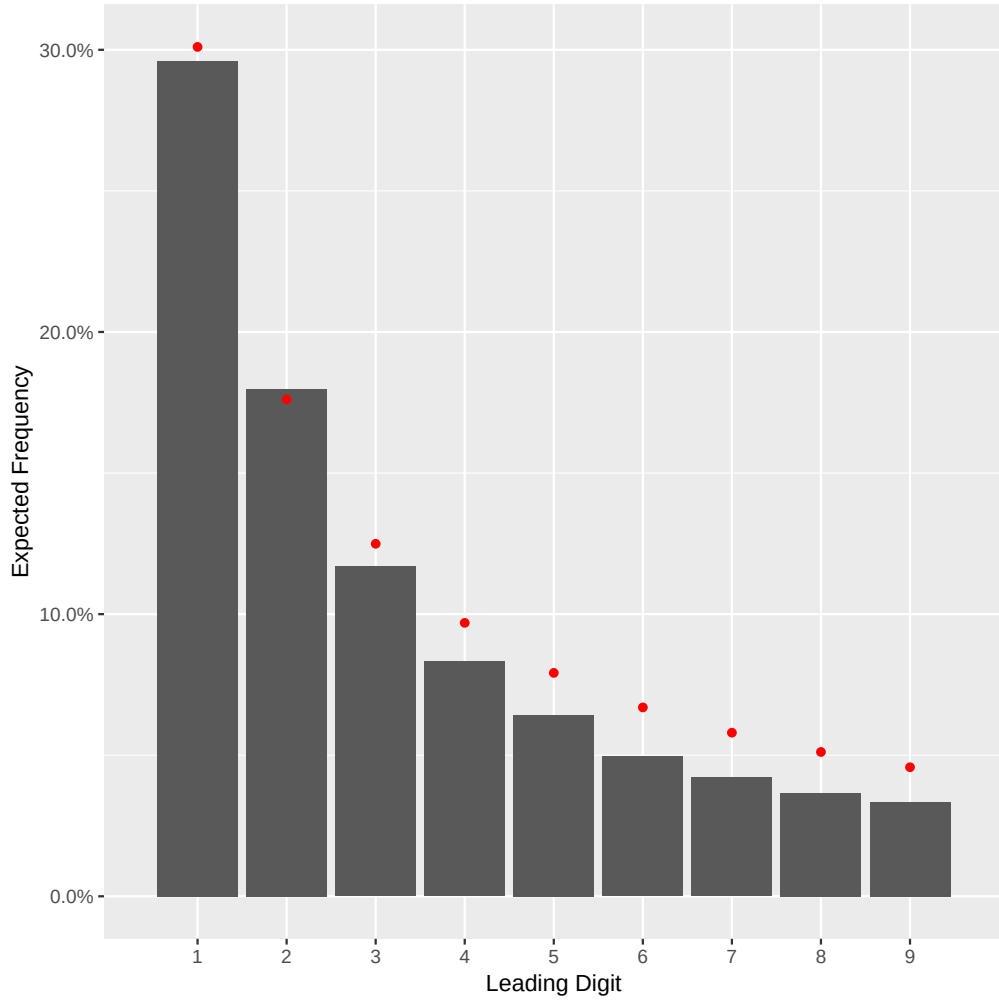
Unit Prices Benfords



Transaction Totals Benfords



Stock Prices and Benfords



Stock Prices, Transaction Totals, and UnitPrices

You can see that the variables with larger standard deviations followed Benford's Law more closely.

	Stock Price	Transaction Total	UnitPrice
Standard Deviation	1,750.06	379.78	97.01
25th Percentile	7.75	3.75	1.25
Median	16.40	9.84	2.08
75th Percentile	31.86	17.40	4.13

Conclusions

Normally distributed variables and variables that do not exhibit substantial variation are unlikely to follow Benford's Law.

Mathematical combinations of variables (e.g., price x quantity) are more likely to follow Benford's Law.

Using *z scores* is a useful way to detect outliers and anomalies.

Knowing how a variable is **distributed** allows one to calculate probabilities of various outcomes and gives an idea of how often we should expect certain things to occur or not occur.

Conclusions (cont.)

Using exploratory data analysis techniques from data science such as:

- Creating visualizations
- Identifying missing values
- Looking at summary statistics for the entire population
- Doing logical checks for transactions that don't make sense (e.g., negative prices, negative quantities, zero quantities, etc.)

may help guide discussions with management or within the audit team.

Thank you!

For Further Information

Robson Glasscock, PhD, CPA

Robson@xbanalyticsllc.com

(303) 204-6024

Attributions

https://www.journalofaccountancy.com/issues/2017/apr/excel-and-benfords-law-to-detect-fraud.html

https://www.ntrand.com/pareto-distribution/

https://oregonaudits.org/2016/01/05/how-to-apply-benfords-law-in-excel-to-detect-fraud

http://mathworld.wolfram.com/BenfordsLaw.html

https://towardsdatascience.com/what-is-benfords-law-and-why-is-it-important-for-data-science

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/T0HSJ1

https://www.kaggle.com/carrie1/ecommerce-data/

Fraud Examination, 6e. Albrecht et al.

https://acfepublic.s3-us-west-2.amazonaws.com/2020-Report-to-the-Nations.pdf