Statistical Foundations

IAA Faculty

3/2/23

Table of contents

1 Statistical Foundations

Authors

Aric Labarr, Phd - Institute for Advanced Analytics @ NC State
Susan Simmons, PhD - Institute for Advanced Analytics @ NC State

with special thanks to the contributions from Dr. Shaina Race

Structure of the book

The book is broken down into small sections that aim to demonstrate a single concept at a time. The companion text, Data Manipulation Techniques with R introduces the foundations of programming in R. This book is a work in progress. Submit any issues here. Please check back frequently for updates.

Acknowledgements

The authors would like to thank the members of the faculty and the TAs at the Institute for Advanced Analytics for providing feedback on this work.

2 Introduction to Statistics

Welcome to your introduction to statistics. You will be learning the basics of statistics, along with applications of statistics within the R language. This book will provide fundamentals of the concepts and the code to apply these concepts in R.
This Chapter aims to answer the following questions:

What type of data is being analyzed?

Nominal

Ordinal

Continuous/Discrete

How do we describe distributions of these variables?

Center

Spread

Shape

Graphical Display

How do we create confidence intervals for parameters?

How do we perform hypothesis testing?

One sample t-test

Two sample t-test

Testing Normality

Testing Equality of Variances

Testing Equality of Means

Mann-Whitney-Wilcoxon Test

The following packages will be used in this textbook. Below we install and add the packages to our libraries so that any version issues can be dealt with at the beginning of the course. Sometimes packages require that you update to the latest version of R; if you see an error that indicates that situation, download the latest version of R from CRAN and install it.

```
install.packages('AmesHousing')
install.packages('tidyverse')
install.packages('car')
install.packages('DescTools')
install.packages('corrplot')
install.packages('mosaic')
install.packages('modelr')
install.packages('plotly')
install.packages('ggplot2')
install.packages('Hmisc')
install.packages('onehot')
install.packages('jmuOutlier')
install.packages('leaps')
install.packages('glmnet')
install.packages('nortest')
install.packages('lmtest')
install.packages('InformationValue')
install.packages('gmodels')
install.packages('vcdExtra')
install.packages('TSA')
install.packages('carData')
install.packages('epiDisplay')
install.packages('gridExtra')
```

```
library(AmesHousing)
library(tidyverse)
library(car)
library(DescTools)
library(corrplot)
library(mosaic)
library(modelr)
library(plotly)
library(ggplot2)
library(Hmisc)
library(onehot)
library(jmuOutlier)
library(leaps)
library(glmnet)
library(nortest)
library(lmtest)
library(InformationValue)
```

```
library(gmodels)
library(vcdExtra)
library(TSA)
library(carData)
library(epiDisplay)
library(gridExtra)
```

2.1 Exploratory Data Analysis (EDA)

The crucial first step to any data science problem is exploratory data analysis (EDA). Before you attempt to run any models, or jump towards any formal statistical analysis, you must ***explore your data***. Many unexpected frustrations arise when exploratory analysis is overlooked; knowing your data is critical to your ability to make necessary assumptions about it. This preliminary analysis will help inform our decisions for data manipulation, give us a base-level understanding of our variables and the relationships between them, and help determine which statistical analyses might be appropriate for the questions we are trying to answer. Some of the questions we aim to answer through exploratory analysis are:

What kind of variables to you have?

Continuous

Nominal

Ordinal

How are the attributes stored?

Strings

Integers

Floats/Numeric

Dates

What do their distributions look like?

Center/Location

Spread

Shape

Are there any anomolies?

Outliers

Leverage points

Missing values

Low-frequency categories

Throughout the textbook, we will continue to use a real-estate data set that contains the `sale_price` and numerous physical attributes of nearly 3,000 homes in Ames, Iowa in the early 2000s. To access this data, we first add the `AmesHousing` package to our library and create the nicely formatted data with the `make_ordinal_ames()` function.

```
library(AmesHousing)

ames <- make_ordinal_ames()
str(ames)
```

2.1.1 Types of Variables

The columns of a data set are referred to by the following **equivalent terms**:

Variables

Features

Attributes

Predictors/Targets

Factors

Inputs/Outputs

This book may use any of these words interchangeably to refer to a quality or quantity of interest in our data.

Nominal Variables

A **nominal** or **categorical variable** is a *quality of interest* whose values have no logical ordering. Color ("blue", "red", "green"...), ethnicity ("African-American", "Asian", "Caucasian",...), and style of house ("ranch", "two-story", "duplex", ...) are all examples of nominal attributes. The categories or values that these variables can take on - those words listed in quotes and parenthesis - are called the **levels** of the variable.

In modeling, nominal attributes are commonly transformed into **dummy variables**. Dummy variables are binary columns that indicate the presence or absence of a quality. There is more than one way to create dummy variables, and the treatment will be different depending on

what type of model you are using. Linear regression models will use either **reference-level** or **effects coding**, whereas other machine learning models are more likely to use one-hot encoding or a variation thereof.

One-hot encoding

For machine learning applications, it is common to create a binary dummy column for each level of your categorical variable. This is the most intuitive representation of categorical information, answering indicative questions for each level of the variable: *"is it blue?"*, *"is it red?"* etc. The table below gives an example of some data, the original nominal variable (color) and the one-hot encoded color information.

Table 2.1: One-hot dummy variable coding for the categorical attribute *color*

Observation	Color	Blue	Red	Yellow	Other
1	Blue	1	0	0	0
2	Yellow	0	0	1	0
3	Blue	1	0	0	0
4	Red	0	1	0	0
5	Red	0	1	0	0
6	Blue	1	0	0	0
7	Yellow	0	0	1	0
8	Other	0	0	0	1

We will demonstrate the creation of this data using some simple random categorical data:

```
set.seed(41)
dat <- data.frame(y = c(rnorm(10,2), rnorm(10,1),rnorm(10,0)),
                  x1 = factor(rep(c("A", "B", "C"), each = 10)),
                  x2 = factor(rep(c("Z", "X", "Y","W","V","U"), each = 5)))
print(dat)
```

```
           y x1 x2
1  1.2056317  A  Z
2  2.1972575  A  Z
3  3.0017043  A  Z
4  3.2888254  A  Z
5  2.9057534  A  Z
6  2.4936675  A  X
7  2.5992858  A  X
8  0.4203930  A  X
9  3.0006207  A  X
```

```
10  4.1880077  A  X
11 -0.2093244  B  Y
12  0.4126881  B  Y
13  2.0561206  B  Y
14  0.6834151  B  Y
15  0.9454590  B  Y
16  1.3297513  B  W
17  1.6630951  B  W
18  1.8783282  B  W
19  1.2028743  B  W
20  3.2744025  B  W
21 -0.8992970  C  V
22  2.1394903  C  V
23 -1.1659510  C  V
24 -0.0471304  C  V
25  0.4158763  C  V
26  1.7200805  C  U
27 -0.7843607  C  U
28 -1.3039296  C  U
29 -0.4520359  C  U
30 -1.7739919  C  U
```

Unlike reference and effects coding, which are typically specified within the `lm()` function as we will see in Chapter Chapter **??**, one-hot encoding is most quickly achieved through use of the `onehot` package in R, which first creates an "encoder" to do the job quickly.

The speed of this function has been tested against both the base R `model.matrix()` function and the `dummyVars()` function in the `caret` package and is *substantially* faster than either.

```
library(onehot)

encoder  = onehot(dat)
dummies = predict(encoder,dat)
head(dummies)
```

```
           y x1=A x1=B x1=C x2=U x2=V x2=W x2=X x2=Y x2=Z
[1,] 1.205632    1    0    0    0    0    0    0    0    1
[2,] 2.197258    1    0    0    0    0    0    0    0    1
[3,] 3.001704    1    0    0    0    0    0    0    0    1
[4,] 3.288825    1    0    0    0    0    0    0    0    1
[5,] 2.905753    1    0    0    0    0    0    0    0    1
[6,] 2.493667    1    0    0    0    0    0    1    0    0
```

Reference-level coding

Reference-level coding is similar to one-hot encoding except one of the levels of the attribute, called the **reference level**, is omitted. Notice that the 4 dummy columns from Table Table **??** collectively form a linearly dependent set; that is, if you know the values of 3 of the 4 dummy variables you can determine the 4^{th} with complete certainty. This would be a problem for linear regression, where we assume our input attributes are not linearly dependent as we will discuss in Chapter Chapter **??**.

A reference level of the attribute is often specified by the user to be a particular level worthy of comparison (a baseline), as the estimates in the regression output will be interpreted in a way that compares each non-reference level to the reference level. If a reference level is not specified by the user, one will be picked by the software by default either using the order in which the levels were encountered in the data, or their alphabetical ordering. Users should check the documentation of the associated function to understand what to expect.

Table Table **??** transforms the one-hot encoding from Table Table **??** into reference-level coding with the color "blue" as the reference level. Notice the absence of the column indicating "blue" and how each blue observation exists as a row of zeros.

Table 2.2: Reference-level dummy variable coding for the categorical attribute *color* and the reference level of "blue"

Observation	Color	Red	Yellow	Other
1	Blue	0	0	0
2	Yellow	0	1	0
3	Blue	0	0	0
4	Red	1	0	0
5	Red	1	0	0
6	Blue	0	0	0
7	Yellow	0	1	0
8	Other	0	0	1

Effects coding

Effects coding is useful for obtaining a more general comparative interpretation when you have approximately equal sample sizes across each level of your categorical attribute. Effects coding is designed to allow the user to compare each level to *all* of the other levels. More specifically the mean of each level is compared to the **overall mean** of your data. However, the comparison is actually to the so-called *grand mean*, which is the mean of the means of each group. When sample sizes are equal, the grand mean and the overall sample mean are equivalent. When sample sizes are *not* equal, the parameter estimates for effects coding should not be used for interpretation or explanation.

Effects coding still requires a *reference level*, however the purpose of the reference level is not the same as it was in reference-level coding. Here, the reference level is left out in the sense that no comparison is made between it and the overall mean. Table Table **??** shows our same example with effects coding. Again we notice the absence of the column indicating "blue" but now the reference level receives values of `-1` rather than `0` for all 3 dummy columns. We will revisit the interpretation of linear regression coefficients under this coding scheme in Chapter Chapter **??**.

Table 2.3: Effects coding for the categorical attribute *color* and the reference level of "blue"

Observation	Color	Red	Yellow	Other
1	Blue	-1	-1	-1
2	Yellow	0	1	0
3	Blue	-1	-1	-1
4	Red	1	0	0
5	Red	1	0	0
6	Blue	-1	-1	-1
7	Yellow	0	1	0
8	Other	0	0	1

Interval Variables

An interval variable is a *quantity of interest* on which the mathematical operations of addition, subtraction, multiplication and division can be performed. Time, temperature and age are all examples of interval attributes. To illustrate the definition, note that "15 minutes" divided by "5 minutes" is 3, which indicates that 15 minutes is 3 times as long as 5 minutes. The sensible interpretation of this simple arithmetic sentence demonstrates the nature of interval attributes. One should note that such arithmetic would not make sense in the treatment of nominal variables.

Ordinal Variables

Ordinal variables are attributes that are qualitative in nature but have some natural ordering. *Level of education* is a common example, with a level of 'PhD' indicating *more* education than 'Bachelors' but lacking a numerical framework to quantify *how much more*. The treatment of ordinal variables will depend on the application. Survey responses on a Likert scale are also ordinal - a response of 4="somewhat agree" on a 1-to-5 scale of agreement cannot reliably be said to be twice as enthusiastic as a response of 2="somewhat disagree". These are not interval measurements, though they are often treated as such in a trade-off for computational efficiency.

Ordinal variables will either be given some numeric value and treated as interval variables or they will be treated as categorical variables and dummy variables will be created. The choice of solution is up to the analyst. When numeric values are assigned to ordinal variables, the possibilities are many. For example, consider *level of education*. The simplest ordinal treatment for such an attribute might be something like Table Table **??**.

Table 2.4: One potential approach to scaling the ordinal attribute level of education

Level of Education	Numeric Value
No H.S. Diploma	1
H.S. Diploma or GED	2
Associates or Certificate	3
Bachelors	4
Graduate Certificate	5
Masters	6
PhD	7

While numeric values have been assigned and this data *could* be used like an interval attribute, it's important to realize that the notion of a "one-unit increase" is qualitative in nature rather than quantitative. However, if we're interested in learning whether there is a *linear* type of relationship between education and another attribute (meaning as education level increases, the value of another attribute increases or decreases), this would be the path to get us there. However we're making an assumption in this model that the difference between a H.S. Diploma and an Associates degree (a difference of "1 unit") is the same as the difference between a Master's degree and a PhD (also a difference of "1 unit"). These types of assumptions can be flawed, and it is often desirable to develop an alternative system of measurement based either on domain expertise or the target variable of interest. This is the notion behind **optimal scaling** and **target-level encoding**.

Optimal Scaling

The primary idea behind optimal scaling is to transform an ordinal attribute into an interval one in a way that doesn't restrict the numeric values to simply the integers $1, 2, 3, ...$ It's reasonable for a data scientist to use domain expertise to develop an alternative scheme.

For example, if analyzing movie theater concessions with ordinal drink sizes {small, medium, large}, one is not restricted to the numeric valuation of 1=small, 2=medium, and 3=large just because it's an ordinal variable with 3 levels. Perhaps it would make more sense to use the drink size in fluid ounces to represent the ordinality. If the small drink is 12 ounces, the medium is 20 ounces, and the large is 48 ounces, then using those values as the numerical representation would be every bit as (if not more) reasonable than using the standard integers 1, 2, and 3.

If we re-consider the ordinal attribute level of education, we might decide to represent the approximate years of post-secondary schooling required to obtain a given level. This might lead us to something like the attribute values in Table Table **??**

Table 2.5: One potential approach to scaling the ordinal attribute level of education

Level of Education	Numeric Value
No H.S. Diploma	-1
H.S. Diploma or GED	0
Associate's or Certificate	2
Bachelor's	4
Graduate Certificate	5
Master's	6
PhD	8

If we were modeling the effect of education on something like salary, it seems reasonable to assume that the jumps between levels should not have equal distance like they did in Table **??**. It seems reasonable to assume that one would experience a larger salary lift from Associate's to Bachelor's degree than they would from No H.S. Diploma to GED. The most common way to determine the numeric values for categories is to use information from the response variable. This is commonly referred to as **target level encoding**.

Target Level Encoding

The values in Table Table **??** might have struck the reader as logical but arbitrary. To be more scientific about the determination of those numeric values, one might wish to use information from the response variable to obtain a more precise expected change in salary for each level increase in education. At first hearing this, one might question the validity of the technique; isn't the goal to *predict* salary? This line of thought is natural, which is why having a holdout sample is extremely important in this situation. To implement Target level encoding, we can simply create a look-up table that matches each level of education to the average or median salary obtained for that level. These values can be used just as readily as the arbitrary levels created in Table Table **??** to encode the ordinal attribute!

2.1.2 Distributions

After reviewing the types and formats of the data inputs, we move on to some basic **univariate** (one variable at a time) analysis. We start by describing the distribution of values that each variable takes on. For nominal variables, this amounts to frequency tables and bar charts of how often each level of the variable appears in the data set.

We'll begin by exploring one of our nominal features, `Heating_QC` which categorizes the quality and condition of a home's heating system. To create plots in R, we will use the popular `ggplot2`

library. At the same time, we will load the `tidyverse` library which we will use in the next chunk of code.

```
library(ggplot2)
library(tidyverse)
ggplot(data = ames) +
  geom_bar(mapping = aes(x = Heating_QC),fill="orange") + labs(x="Heating System Quality",
```

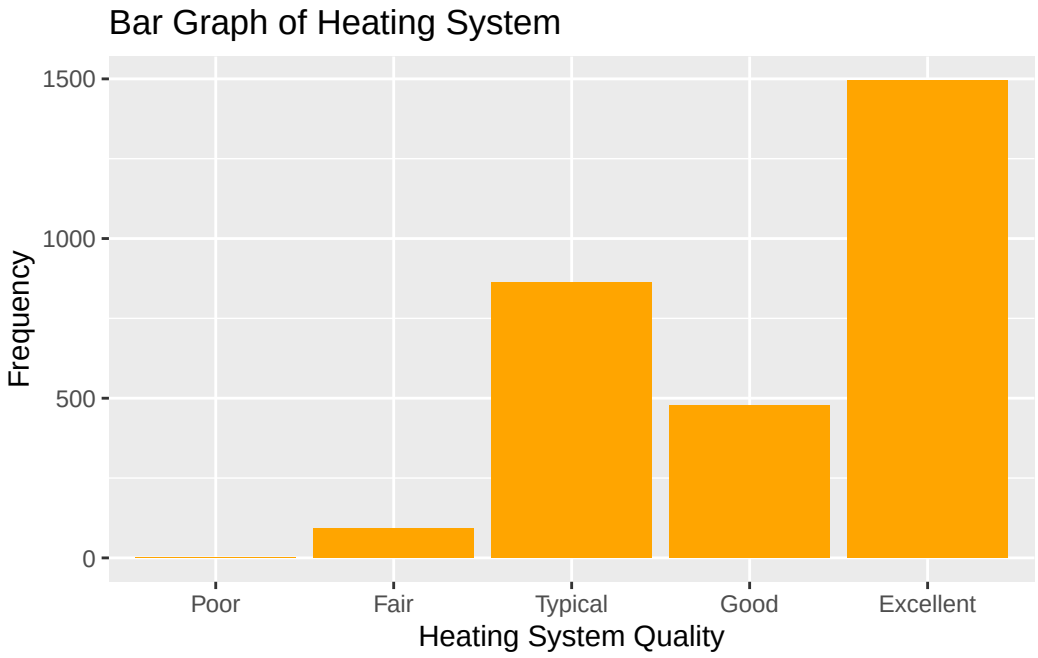

Figure 2.1: Distribution of Nominal Variable Heating_QC

To summon the same information in tabular form, we can use the `count()` function to create a table:

```
ames %>%
  count(Heating_QC)
```

```
# A tibble: 5 x 2
  Heating_QC     n
  <ord>      <int>
1 Poor           3
2 Fair          92
3 Typical      864
```

```
4 Good          476
5 Excellent    1495
```

You'll notice that very few houses (3) have heating systems in `Poor` condition, and the majority of houses have systems rated `Excellent`. **It will likely make sense to combine the categories of `Fair` and `Poor` in our eventual analysis, a decision we will later revisit.**

Next we create a **histogram** for an interval attribute like `Sale_Price`:

```
ggplot(data = ames) +
  geom_histogram(mapping = aes(x = Sale_Price/1000),fill="blue") +
  labs(x = "Sales Price (Thousands $)",y="Frequency",title="Histogram of Sales Price in Th
```

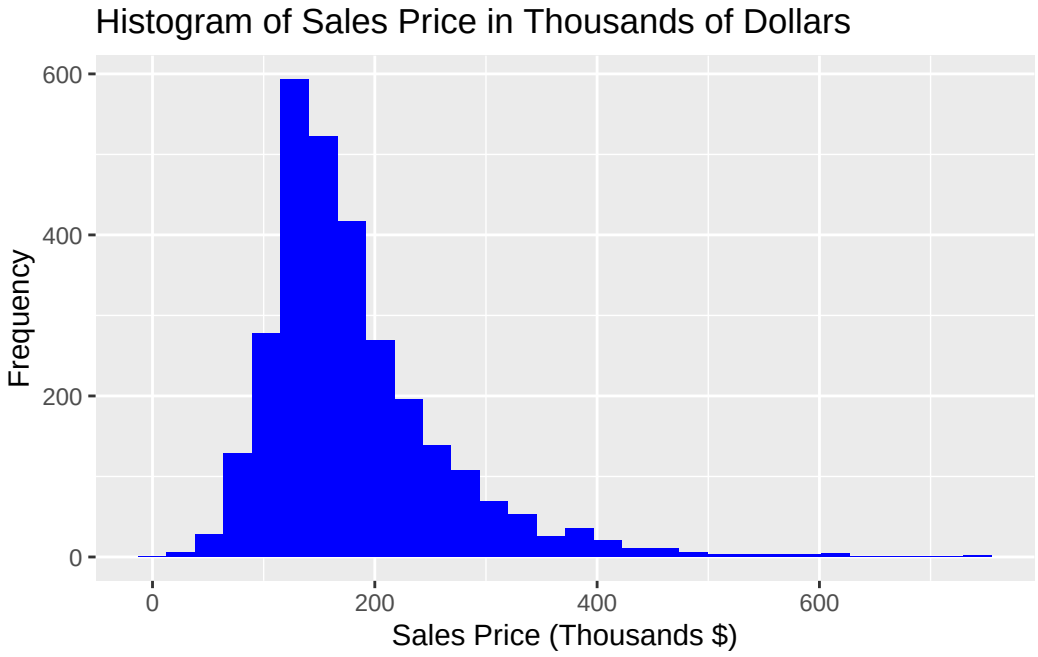

Figure 2.2: Distribution of Interval Variable Sale_Price

From this initial inspection, we can conclude that most of the houses sell for less than $200,000 and there are a number of expensive anomalies. To more concretely describe and quantify a statistical distribution, we use statistics that describe the *location, spread, and shape* of the data.

Location

The *location* is a measure of central tendency for a distribution. Most common measures of central tendency are **mean**, **median**, and **mode**.

We define each of these terms below for a variable **x** having n observations with values $\{x_i\}_{i=1}^n$, sorted in order of magnitude such that $x_1 \leq x_2 \leq \cdots \leq x_n$:

Mean: The **average** of the observations, $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} x_i$

Median: The "middle value" of the data. Formally, when n is odd, the median is the observation value, $x_m = x_{\frac{(n+1)}{2}}$ for which $x_i < x_m$ for 50% of the observations (excluding x_m). When n is even, x_m is the average of $x_{\frac{n}{2}}$ and $x_{(\frac{n}{2}+1)}$. The median is also known as the 2^{nd} **quartile** (see next section).

Mode: The most commonly occurring value in the data. Most commonly used to describe nominal attributes.

Example

The following table contains the heights of 10 students randomly sampled from NC State's campus. Compute the mean, median, mode and quartiles of this variable.

height	60	62	63	65	67	67	67	68	68	69

Solution:

The mean is `(60+62+63+65+67+67+67+68+68+69)/10` = 65.6.

The median (second quartile) is `(67+67)/2` = 67.

The mode is 67.

Spread

Once we have an understanding of the central tendency of a data set, we move on to describing the spread (the dispersion or variation) of the data. **Range**, **interquartile range**, **variance**, and **standard deviation** are all statistics that describe spread.

Range: The difference between the maximum and minimum data values.

Sample variance: The sum of squared differences between each data point and the mean, divided by (n-1). $\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

Standard deviation: The square root of the sample variance.

In order to define **interquartile range**, we must first define **percentiles** and **quartiles**.

Percentiles: The 99 intermediate values of the data which divide the observations into 100 equally-sized groups. The r^{th} percentile of the data, P_r is the number for which $r\%$ of the data is less than P_r.

Quartiles: The quartiles of the data are the 25^{th}, 50^{th} and 75^{th} percentiles. They are denoted as Q_1 (1^{st} quartile), Q_2 (2^{nd} quartile = median) and Q_3 (3^{rd} quartile), respectively.

Interquartile range (IQR): The difference between the 25^{th} and 75^{th} percentiles.

One should note that standard deviation is more frequently reported than variance because it shares the same units as the original data, and because of the guidance provided by the empirical rule. If we're exploring something like `Sale_Price`, which has the unit "dollars", then the variance would be measured in "square-dollars", which hampers the intuition. Standard deviation, on the other hand, would share the unit "dollars", aiding our fundamental understanding.

Example Let's again use the table of heights from the previous example, this time computing the range, IQR, sample variance and standard deviation.

height	60	62	63	65	67	67	67	68	68	69

Solution:

The range `69-60` = 9.

The variance is `((60-65.6)^2+(62-65.6)^2+(63-65.6)^2+(65-65.6)^2+(67-65.6)^2+(67-65.6)^2+(67-65.`
= 8.933

The standard deviation is `sqrt(8.933)` = 2.989

The first quartile is `(62+63)/2` = 62.5

The third quartile is `(68+68)/2` = 68

The IQR is `68 - 62.5` = 5.5.

Shape

The final description we will want to give to distributions regards their shape. Is the histogram *symmetric*? Is it *unimodal* (having a single large "heap" of data) or *multimodal* (having multiple heaps")? Does it have a longer tail on one side than the other (*skew*)? Is there a lot more or less data in the tails than you might expect?

We'll formalize these ideas with some illustrations. A distribution is right (left) skewed if it has a longer tail on its right (left) side, as shown in Figure Figure **??**.

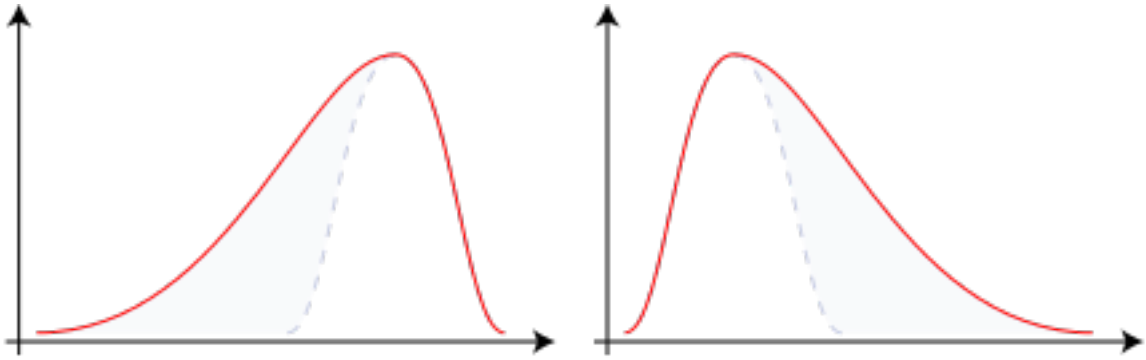

Figure 2.3: Examples of Left-Skewed (Negative Skew) and Right-skewed (Positive Skew) distributions respectively

A distribution is called *bimodal* if it has two "heaps", as shown in Figure Figure **??**.

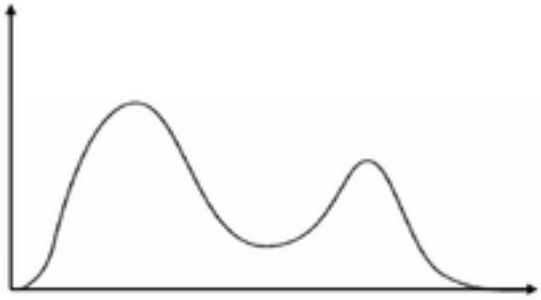

Figure 2.4: Example of a Bimodal Distribution

Summary Functions in R

There are many ways to obtain all of the statistics described in the preceding sections, below we highlight 3:

The `describe` function from the `Hmisc` package which can work on the entire dataset or a subset of columns.

```
library(Hmisc)

Hmisc::describe(ames$Sale_Price)
```

```
ames$Sale_Price
       n  missing distinct     Info     Mean      Gmd      .05      .10
```

```
    2930        0     1032        1   180796    81960    87500   105450
     .25      .50      .75      .90      .95
  129500   160000   213500   281242   335000

lowest :  12789  13100  34900  35000  35311, highest: 611657 615000 625000 745000 755000
```

The tidyverse `summarise` function, in this case obtaining statistics for each `Exter_Qual` separately.

```
library(tidyverse)
ames %>% group_by(Exter_Qual) %>% dplyr::summarise(average = mean(Sale_Price), st.dev = sd
```

```
# A tibble: 4 x 5
  Exter_Qual average  st.dev maximum minimum
  <ord>        <dbl>   <dbl>   <int>   <int>
1 Fair        89924.  38014.  200000   13100
2 Typical    143374.  41504.  415000   12789
3 Good       230756.  70411.  745000   52000
4 Excellent  377919. 106988.  755000  160000
```

The base R `summary` function, which can work on the entire dataset or an individual variable

```
summary(ames$Sale_Price)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12789  129500  160000  180796  213500  755000
```

2.1.3 The Normal Distribution

The normal distribution, also known as the Gaussian distribution, is one of the most fundamental concepts in statistics. It is one that arises naturally out of many applications and settings. The normal distribution has the following characteristics:

Symmetric

Fully defined by mean and standard deviation (equivalently, variance)

Bell-shaped/Unimodal

Mean = Median = Mode

Assymptotic to the x-axis (theoretical bounds are $-\infty$ to ∞)

Much of the normal distributions utility can be summarized in the **empirical rule**, which states that:

$\approx 68\%$ of data in normal distribution lies within 1 standard deviation of the mean.

$\approx 95\%$ of data in normal distribution lies within 2 standard deviations of the mean.

$\approx 99.7\%$ of data in normal distribution lies within 3 standard deviations of the mean.

We can thus conclude that observations found outside of 3 standard deviations from the mean are quite rare, expected less than 1% of the time.

2.1.4 Skewness

Skewness is a statistic that describes the symmetry (or lack thereof) of a distribution. A normal distribution is perfectly symmetric and has a skewness of 0. Distributions that are more right skewed will have positive values of skewness whereas distributions that are more left skewed will have negative values of skewness.

2.1.5 Kurtosis

Kurtosis is a statistic that describes the *tailedness* of a distribution. The normal distribution has a kurtosis of 3. Distributions that are more tailed (leptokurtic or heavy-tailed) will have kurtosis values greater than 3 whereas distributions that are more less tailed (platykurtic or thin-tailed) will have values of kurtosis less than 3. For this reason, kurtosis is often reported in the form of *excess kurtosis* which is the raw kurtosis value minus 3. This is meant as a comparison to the normal distribution so that positive values indicate thicker tails and negative values indicate thinner tails than the normal.

In Figure Figure **??** below, we compare classical examples of leptokurtic and platykurtic distributions to a normal distribution with the same mean and variance.

2.1.6 Graphical Displays of Distributions

There are three types of plots for examining the distribution of your data values:

Histograms

Normal Probability Plots (QQ-plots)

Box Plots

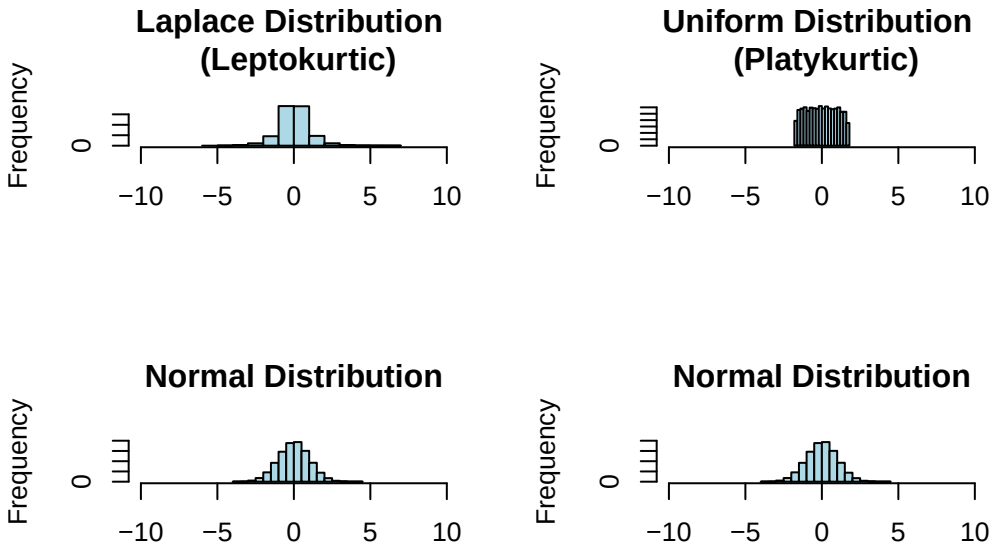

Figure 2.5: The Laplace distribution (top left) is leptokurtic because it has more data in its tails than the normal distribution with the same mean and variance. The uniform distribution (top right) is platykurtic because it has less data in its tails than the normal distribution with the same mean and variance (it effectively has no tails).

Histograms

A histogram shows the shape of a univariate distribution. Each bar in the histogram represents a group of values (a *bin*). The height of the bar represents the either the frequency of or the percent of values in the bin. The width and number of bins is determined automatically, but the user can adjust them to see more or less detail in the histogram. Figure Figure **??** demonstrated a histogram of sale price. Sometimes it's nice to overlay a continuous approximation to the underlying distribution using a *kernal density estimator* with the `geom_density` plot, demonstrated in Figure Figure **??**.

```
ggplot(ames,aes(x=Sale_Price/1000)) +
    geom_histogram(aes(y=..density..), alpha=0.5) + geom_density( alpha = 0.2) +
    labs(x = "Sales Price (Thousands $)")
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```

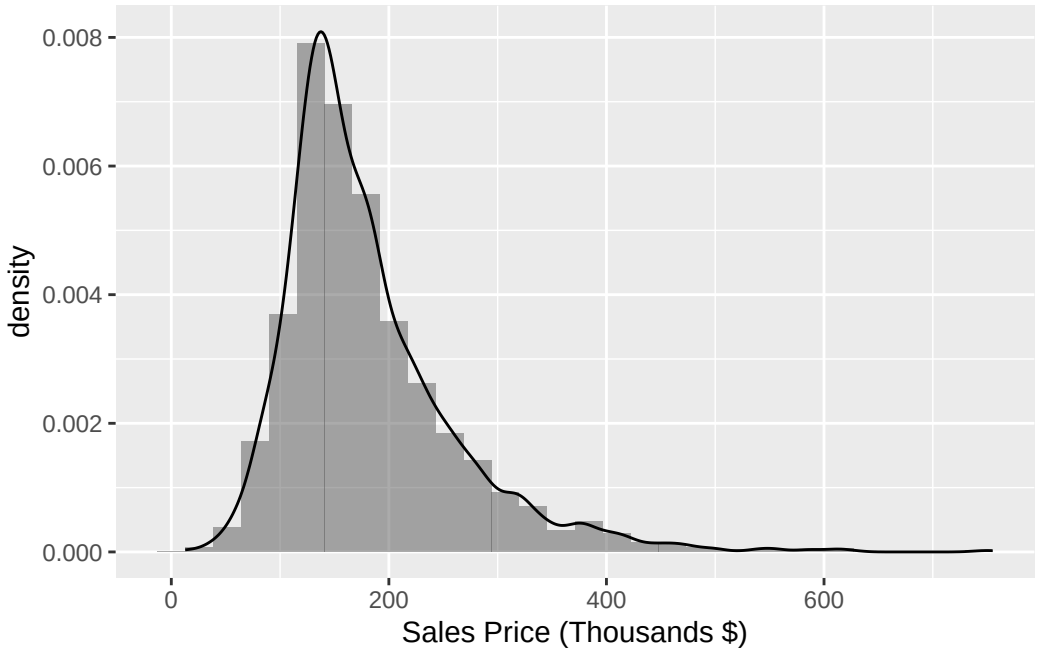

Figure 2.6: Histogram of Sale_Price with kernal density estimator

In our next example, Figure Figure **??**, we'll complicate the previous example by showing *two* distributions of sale price, one for each level of the binary variable `Central_Air`, overlaid on the same axes.

We can immediately see that there are many more houses that have central air than do not in this data. It appears as though the two distributions have different locations, with the purple distribution centered at a larger sale price. To normalize that quantity and compare the raw probability densities, we can change our axes to density as in Figure Figure **??**.

```
ggplot(ames,aes(x=Sale_Price/1000)) +
    geom_histogram(data=subset(ames,Central_Air == 'Y'),aes(fill=Central_Air), alpha = 0.2
    geom_histogram(data=subset(ames,Central_Air == 'N'),aes(fill=Central_Air), alpha  = 0.
    labs(x = "Sales Price (Thousands $)")  + scale_fill_manual(name="Central_Air",values=c
```

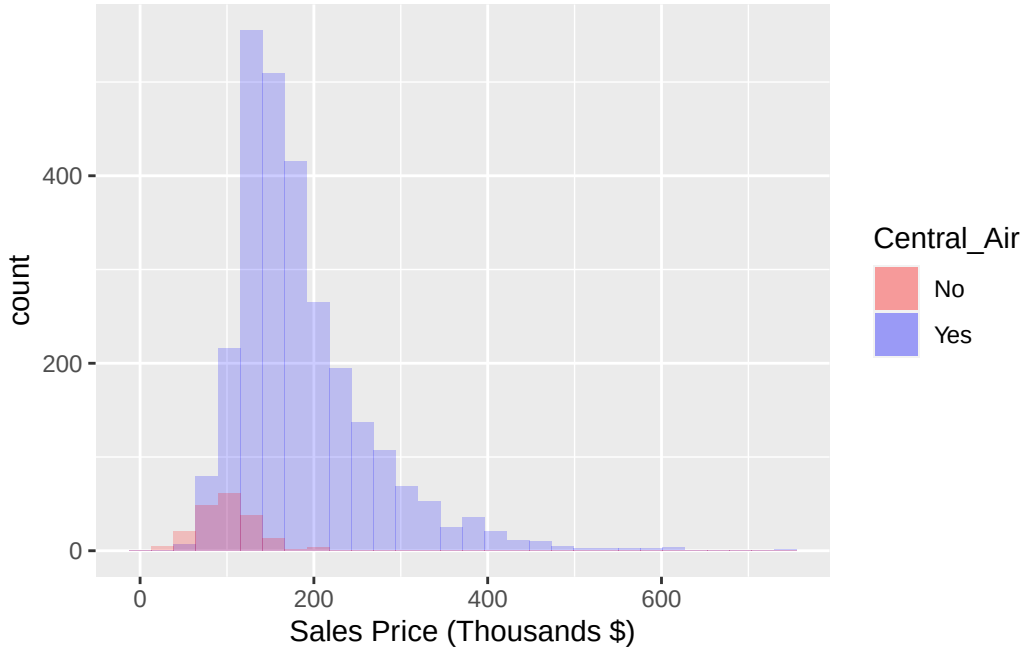

Figure 2.7: Histogram: Frequency of Sale_Price for Each value of Central_Air

```
ggplot(ames,aes(x=Sale_Price/1000)) +
    geom_histogram(data=subset(ames,Central_Air == 'Y'),aes(y=..density..,fill=Central_Air
    geom_histogram(data=subset(ames,Central_Air == 'N'),aes(y=..density..,fill=Central_Air
    labs(x = "Sales Price (Thousands $)")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

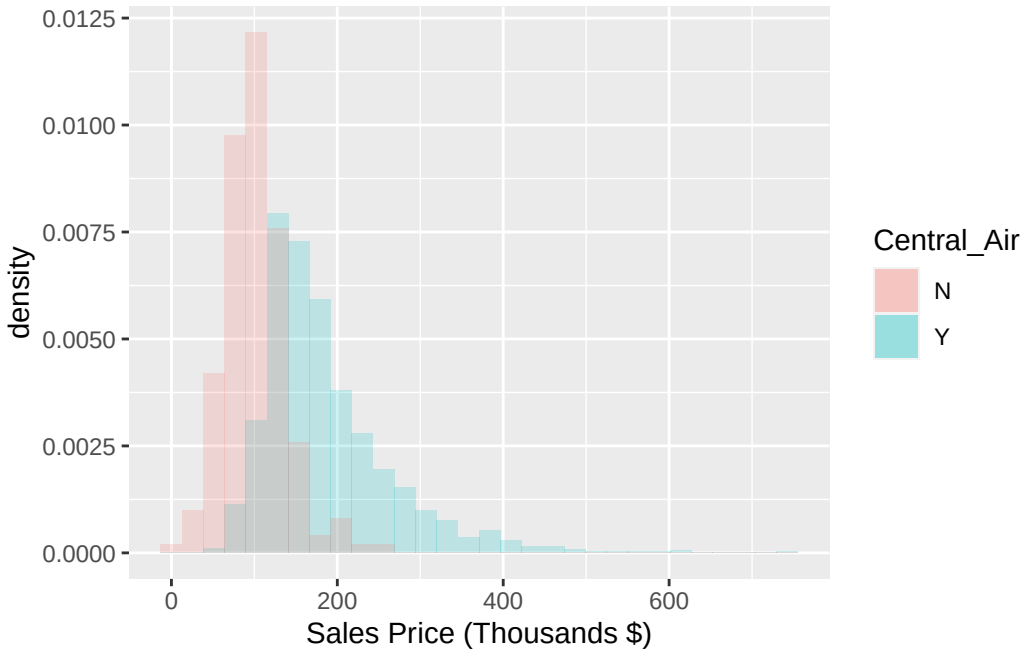

Figure 2.8: Histogram: Density of Sale_Price for varying qualities of Central_Air

To ease our differentiation of the histograms even further, we again employ a kernel density estimator as shown in Figure Figure **??**. This is an appealing alternative to the histogram for continuous data that is assumed to originate from some smooth underlying distribution.

```
ggplot(ames,aes(x=Sale_Price/1000)) +
    geom_density(data=subset(ames,Central_Air == 'Y'),aes(fill=Central_Air), alpha = 0.2)
    geom_density(data=subset(ames,Central_Air == 'N'),aes(fill=Central_Air), alpha = 0.2)
    labs(x = "Sales Price (Thousands $)")
```

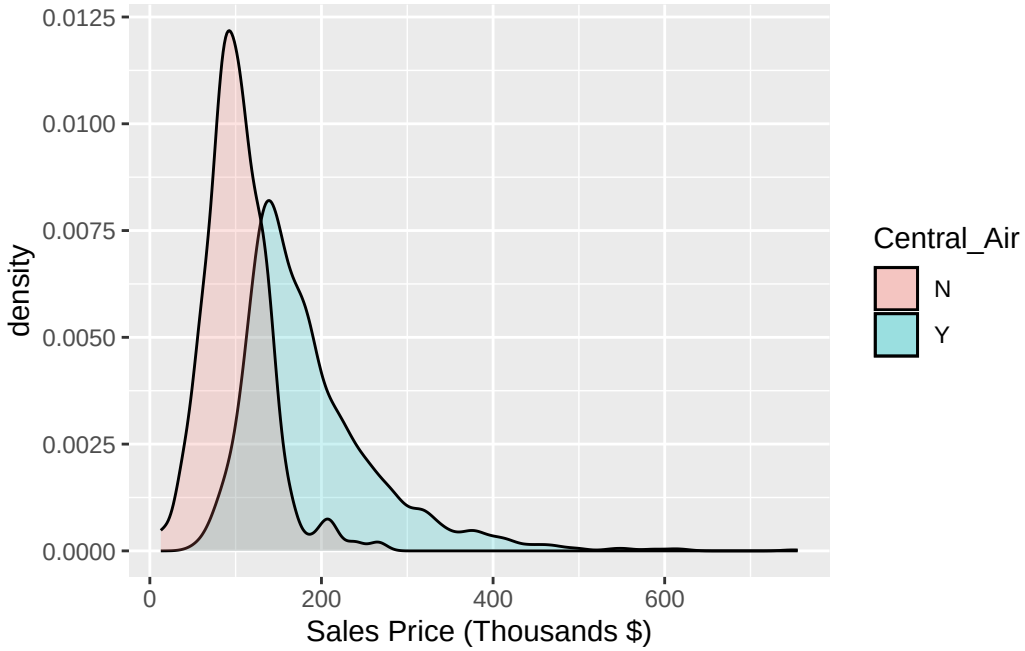

Figure 2.9: Histogram: Density of Sale_Price for varying qualities of Central_Air

Normal probability plots (QQ Plots)

A **normal probability plot** graphs the sorted data values against the values that one would expect if the same number of observations came from a theoretical normal distribution. The resulting image would look close to a straight line if the data was generated by a normal distribution. Strong deviations from a straight line indicate that the data distribution is not normal.

Figure Figure **??** shows a QQ plot for `Sale_Price`, and we can conclude that the variable is not normally distributed (in fact it is right skewed).

```
ggplot(data = ames, aes(sample = Sale_Price/1000)) +
  stat_qq() +
  stat_qq_line()
```

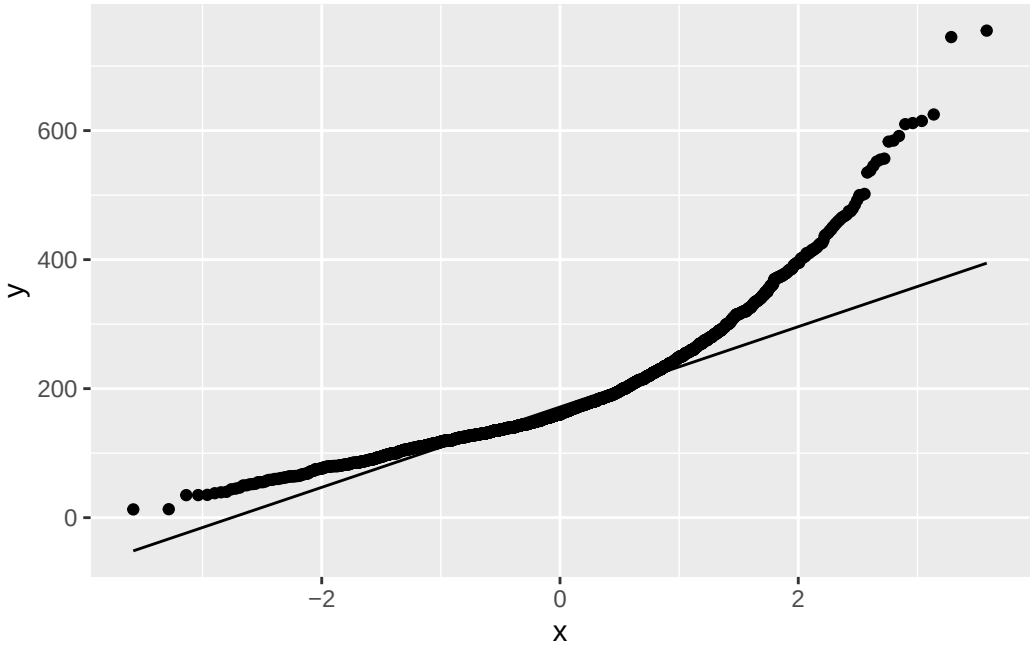

Figure 2.10: QQ-Plot: Quantiles of Sale_Price vs. quantiles of a theoretical normal distribution with same mean and standard deviation. Conclusion: Sale_Price is *not* normally distributed due to a problem with skew.

There are two main patterns that we expect to find when examining QQ-plots:

A quadratic shape, as seen in Figure Figure **??**. This pattern indicates a deviation from normality due to skewness to the data.

An S-shape (or cubic shape), as seen in Figure Figure **??**. This pattern indicates deviation from normality due to kurtosis.

```
df <- data.frame(j1 = rlaplace(10000,0,1))

ggplot(data = df, aes(sample=j1)) +
  stat_qq() +
  stat_qq_line()
```

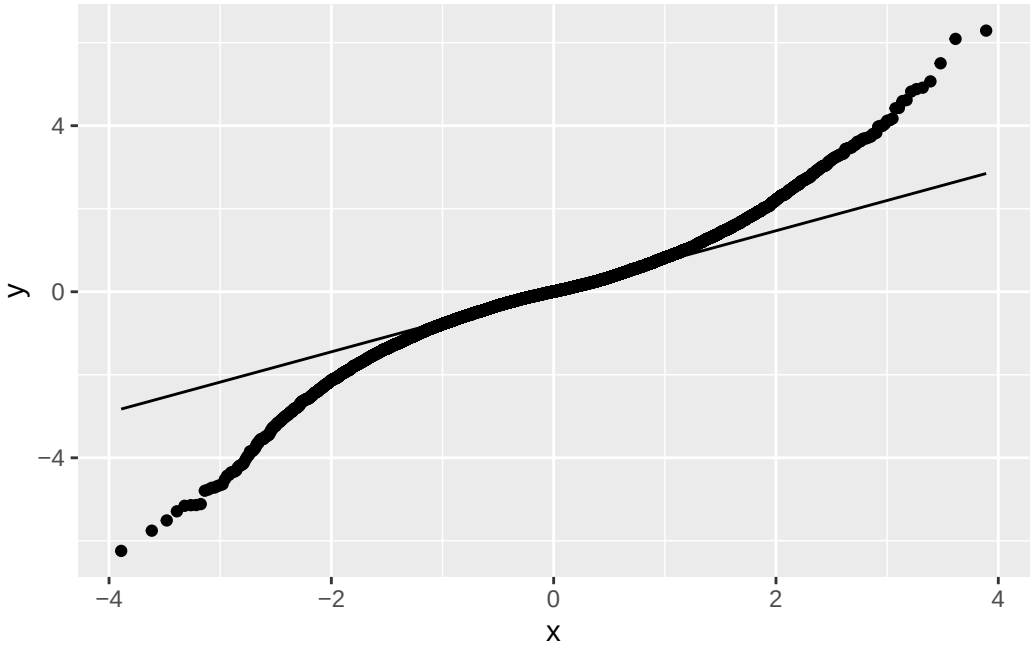

Figure 2.11: QQ-Plot: Quantiles of the Laplace distribution vs. quantiles of a theoretical normal distribution with same mean and standard deviation. Conclusion: Data is *not* normally distributed (in fact it is leptokurtic), due to a problem with kurtosis.

Box Plots

Box plots (sometimes called box-and-whisker plots) will not necessarily tell you about the *shape* of your distribution (for instance a bimodal distribution could have a similar box plot to a unimodal one), but it will give you a sense of the distribution's location and spread and potential skewness.

Many of us have become familiar with the *idea* of a box plot, but when pressed for the specific steps to create one, we realize our familiarity fades. The diagram in Figure Figure **??**) will remind the reader the precise information conveyed by a box plot.

Figure Figure **??** shows the boxplot of `Sale_Price`.

```
ggplot(data = ames, aes(y = Sale_Price/1000)) +
  geom_boxplot() +
  labs(y = "Sales Price (Thousands $)")
```

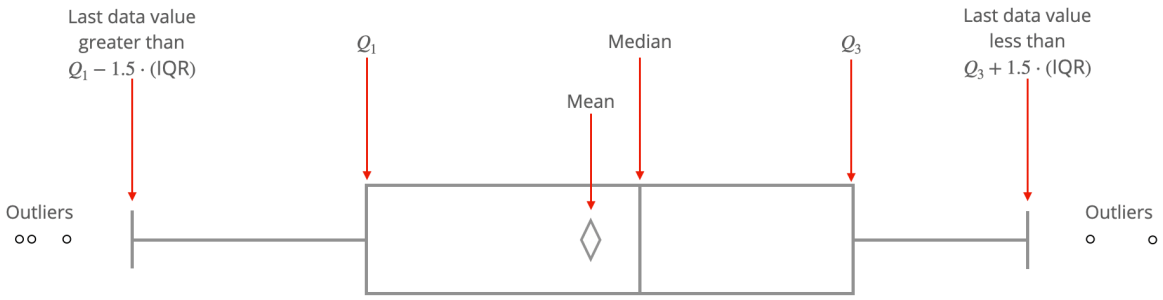

Figure 2.12: Anatomy of a Box Plot.

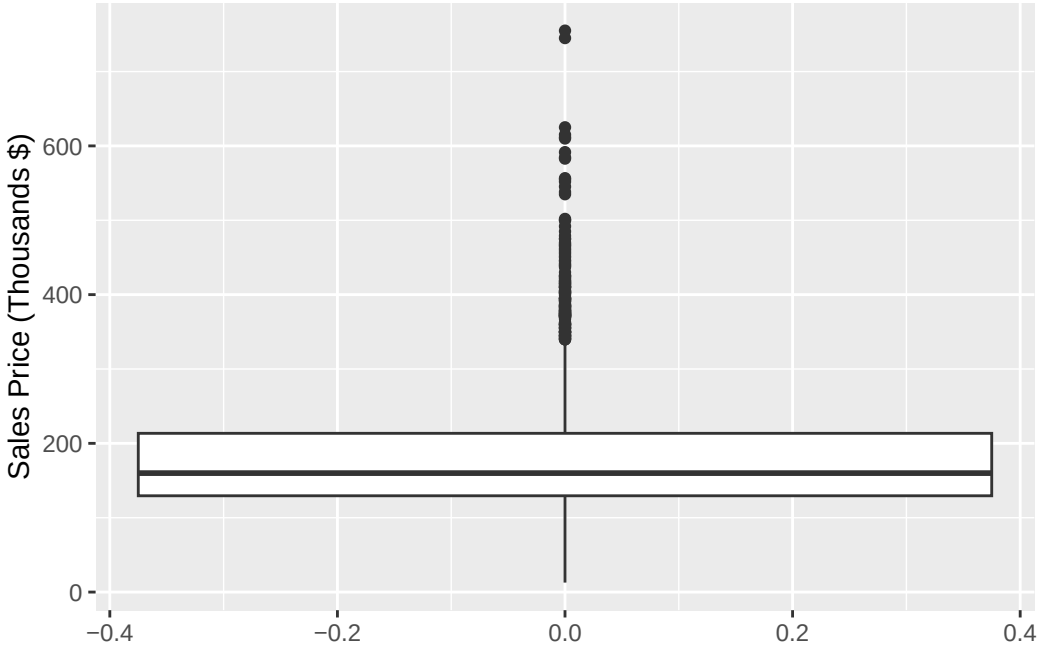

Figure 2.13: Box Plot of Sales Price

Furthermore, we might want to compare the boxplots of `Sale_Price` for different levels of a categorical variable, like `Central_Air` as we did with histograms and densities in Figures Figure **??** and Figure **??**.

The following code achieves this goal in Figure Figure **??**.

```
ggplot(data = ames, aes(y = Sale_Price/1000, x = Central_Air, fill = Central_Air)) +
  geom_boxplot() +
  labs(y = "Sales Price (Thousands $)", x = "Central Air") +
```

```
scale_fill_brewer(palette="Accent") + theme_classic() + coord_flip()
```

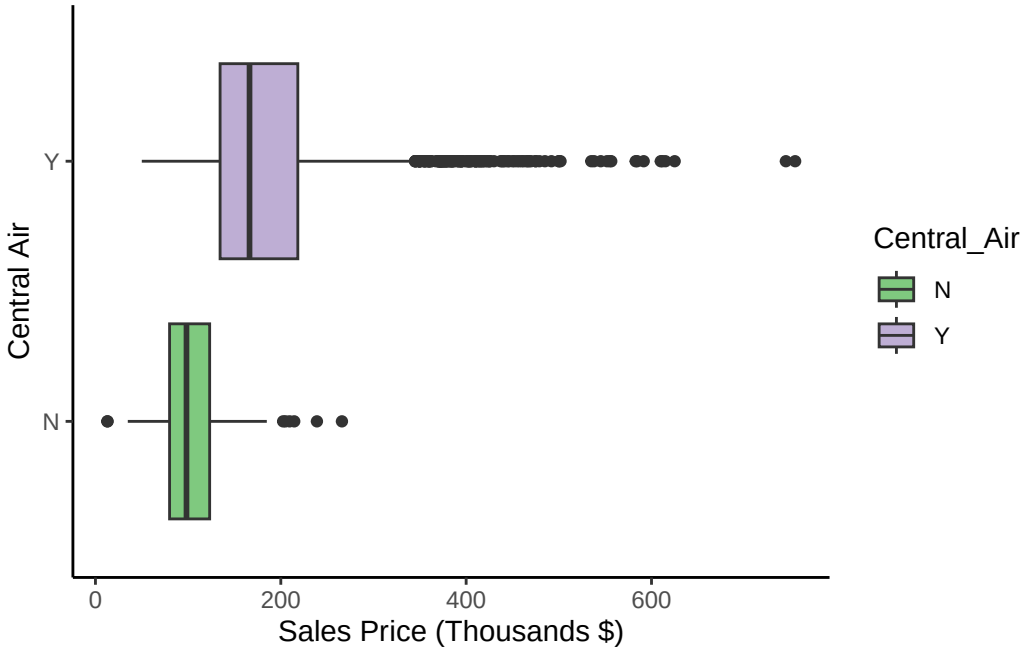

Figure 2.14: Box Plots of Sale_Price for each level of Exter_Qual

2.2 Point Estimates

All the statistics discussed so far have been *point estimates*. They are our *best* estimate at what the population parameter might be, but since we've taken a random sample of data from that population, there must be some uncertainty surrounding that estimate. In statistics, our real interest lies in drawing inferences about an entire population (which we couldn't possibly observe due to time, cost, and/or feasibility constraints) and our approach is to take a representative sample and try to understand what it might tell us about the population.

For the remainder of this text, we will assume our sample is representative of the population. Let's review some common statistical notation of *population parameters* (the true values we are unable to observe) and *sample statistics* (those values we calculate based on our sample)

Population Parmeter	Sample Statistic
Mean (μ)	Sample Average($\bar{x}$)
Variance (σ^2)	Sample Variance(s_x^2)

Population Parmeter	Sample Statistic
Standard deviation (σ^2)	Sample standard deviation(s_x)

Calculating point estimates should lead us to a natural question, one that embodies the field of statistics which aims to quantify uncertainty: *What's the margin of error for this estimate?* This will be the subject of interest in the next section.

2.3 Confidence Intervals

Let's imagine that we want to calculate the average gas mileage of American cars on the road today in order to analyze the country's carbon footprint. It should be clear to the reader that the calculation of the *population mean* would not be possible. The best we could do is take a large representative sample and calculate the sample average. Again, the next question should be: What is the margin of error for this estimate? If our sample average is 21.1 mpg, could the population mean reasonably be 21.2 mpg? how about 25 mpg? 42 mpg?

To answer this question, we reach for the notion of *confidence intervals*. A **confidence interval** is an interval that we believe contains the population mean with some degree of confidence. A confidence interval is associated with a *confidence level*, a percentage, which indicates the strength of our confidence that the interval created actually captured the true parameter.

It's an important nuance to remember that the population mean is a fixed number. The source of randomness in our estimation is our sample. When we construct a 95% confidence interval, we are claiming that, upon repetition of the sampling and interval calculation process, we expect 95% of our created intervals to contain the population mean.

To obtain a confidence interval for a mean in R, we can use the `t.test()` function, as shown below.

```
t.test(ames$Sale_Price, conf.level = 0.95)
```

```
    One Sample t-test

data:  ames$Sale_Price
t = 122.5, df = 2929, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 177902.3 183689.9
sample estimates:
mean of x
```

```
180796.1
```

We can gather based on the output that our 95% confidence interval for the mean of `Sale_Price` is [177902.3, 183689.9]. This function also outputs some extra information that relates to hypothesis testing which we will discuss in Section Section **??**). For now, if we only want to pull the output containing the confidence interval information, we can specify `$conf.int` to the object output from `t.test`:

```
t.test(ames$Sale_Price, conf.level = 0.95)$conf.int
```

```
[1] 177902.3 183689.9
attr(,"conf.level")
[1] 0.95
```

To learn the labels of the various pieces of output, you can list them with the `ls()` function, or by saving the output as an object (below, `results` is the object that stores the output) and exploring it in your environment (upper right panel in RStudio):

```
ls(t.test(ames$Sale_Price, conf.level = 0.95))
```

```
[1] "alternative" "conf.int"    "data.name"   "estimate"    "method"
[6] "null.value"  "p.value"     "parameter"   "statistic"   "stderr"
```

```
results <- t.test(ames$Sale_Price, conf.level = 0.95)
```

2.4 Hypothesis Testing

A confidence interval can help us test a hypothesis about the population mean. A **hypothesis** is merely a statement that we wish to investigate scientifically through the process of statistical inference. In Section Section **??** we proposed some potential hypotheses in the form of questions: If the sample average gas mileage is 21.1, is it possible that the population mean is 21.2? How about 42? The statistical **hypothesis test** can help us answer these questions.

To conduct a hypothesis test, we make an initial assumption. This initial assumption is called the **null hypothesis** and typically denoted as H_0. We then analyze the data and determine whether our observations are likely, given our assumption of the null hypothesis. If we determine that our observed data was unlikely *enough* (beyond some threshold that we set before hand - or beyond a "reasonable doubt" in the justice system) then we *reject* our

initial assumption in favor of the opposite statement, known as the **alternative hypothesis** denoted H_a. The threshold or **significance level** that we use to determine how much evidence is required to reject the null hypothesis is a proportion, α, which specifies how often we're willing to incorrectly reject the null hypothesis (this means that we are assuming the null hypothesis is true). Remember, in applied statistics there are no proofs. Every decision we make comes with some degree of uncertainty. α quantifies our allowance for that uncertainty. In statistical textbooks of years past, $\alpha = 0.05$ was the norm. Later in this text we will propose much smaller values for α depending on your sample size.

In order to quantify how unlikely it was that we observed a statistic as extreme or more extreme than we did, we calculate a **p-value**. The p-value is the area under the **null distribution** that represents the probability that we observed something as extreme or more extreme than we did (assuming the truth of the null hypothesis). If our p-value is less than our confidence level, α, we have enough evidence to reject the null hypothesis in favor of the alternative.

Let's take an example and actually *create* a null distribution. Suppose we flip a fair coin, having equal probability of landing on heads or tails. We can actually simulate this experience with code! The following line of code does just that. Go ahead and run it a few times until you observe a coin flip of each type.

```
sample(c('Heads','Tails'), 1)
```

```
[1] "Heads"
```

Now, let's suppose we do that *many* times and count the number of times we observe one outcome, say `Heads`. This can be done by sampling the values directly into a vector. Let `n` be the number of coin tosses.

```
n=100
outcomes = sample(c('Heads','Tails'), n, replace=T)
```

We can count the number of `Heads` we obtained as follows:

```
sum(outcomes=="Heads")
```

```
[1] 58
```

Every time you run the lines of code above you will find a different set of coin flips and a varying number of `Heads`; *however* the number of `Heads` will revolve around 50, because that is what we'd expect for a fair coin whose probability of `Heads` is 50% (Indeed, this simulates a draw from a binomial distribution with n=100 and p=0.5; the expected value of that distribution is $np = 50$ and the variance is $np(1-p) = 25$).

Thus, if were to do the above experiment thousands of times, we could map out a distribution of how many `Heads` one might reasonably receive by tossing a fair coin 100 times. Let's do that, using a `for` loop. Let `T` be the number of simulated experiments (each experiment tosses the coin 100 times), and let `number_heads` be a vector that stores the number of heads for each experiment. We can initialize `number_heads` with an empty vector. Notice that our loop overwrites the coin toss data in each step, after recording the number of heads.

```
T=10000
n=100
set.seed(11)
number_heads = vector()
for(i in 1:T){
outcomes = sample(c('Heads','Tails'), n, replace=T)
number_heads[i] = sum(outcomes=="Heads")
}

df = data.frame(number_heads)

ggplot(data = df) +
  geom_density(aes(x = number_heads)) +
  labs(x = "Number of heads in 100 tosses")
```

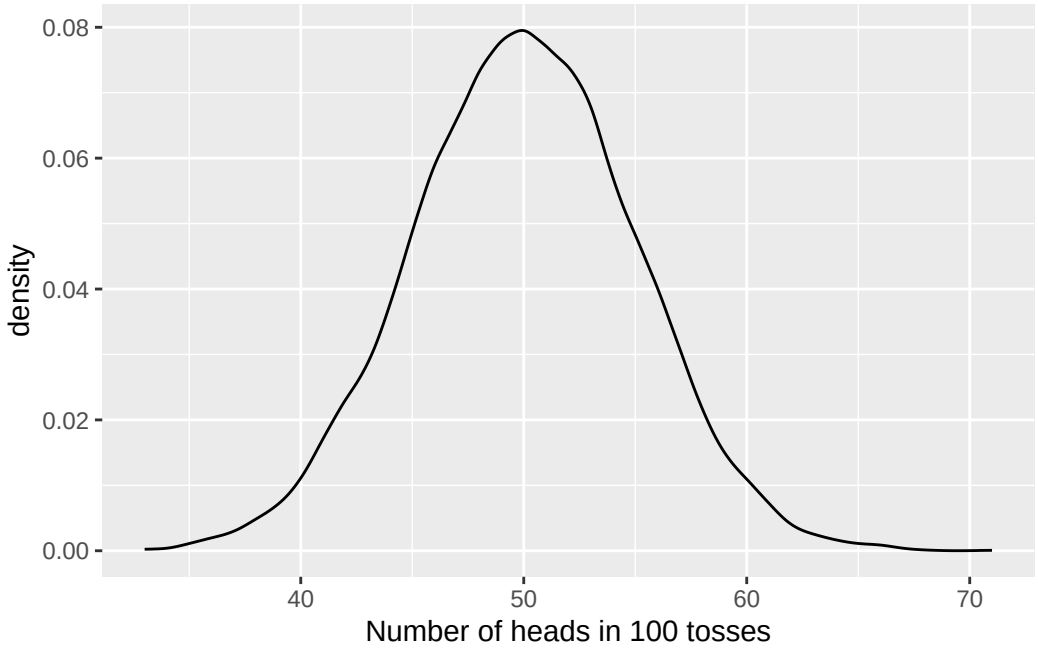

Figure 2.15: Null Distribution: Number of heads for fair coin tossed 100 times

Figure Figure **??** represents our null distribution of the number of heads from a fair coin tossed 100 times. What are the minimum and maximum values of this observed distribution?

```
summary(df$number_heads)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  33.00   47.00   50.00   49.99   53.00   71.00 
```

Example

Suppose now we obtain a new coin from a friend and our hypothesis is that it "feels unfair". We decide that we want a conservative signficance level of 0.01 before we accuse our friend of cheating, so we conduct a hypothesis test. Our null hypothesis must generate a known distribution to which we can compare. Thus our null hypothesis is that the coin is fair:

$$H_0 = \text{The coin is fair:} \quad P(Heads) = 0.5$$

Our alternative hypothesis is the opposite of this:

$$H_0 = \text{The coin is not fair:} \quad P(Heads) \neq 0.5$$

Suppose we flip the coin 100 times and count 65 heads. How likely is it that we would have obtained a result as extreme or more extreme than this if the coin was fair? Here we introduce the notion of a **two-tailed hypothesis test**. Since our hypothesis is that the coin is simply unfair, we want to know how likely it is that we obtained a result *so different from 50*. This is quantified by the *absolute* difference between what we observed and what we expected. Thus, when considering our null distribution, we want to look at the probability we'd obtain something greater than or equal to 65 ($= 50+15$) heads, *or* less than or equal to 35 ($= 50-15$) heads.

Let's take a look at this graphically through our simulated data:

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

We can use this simulated distribution to estimate the p-value associated with obtaining 65 heads (the red area highlighted in Figure Figure **??**. We'd simply calculate the proportion of times we observed values equal to or more extreme than 65 - this is the very definition of a p-value. In the following line of code, `|` represents the logical "or" operator.

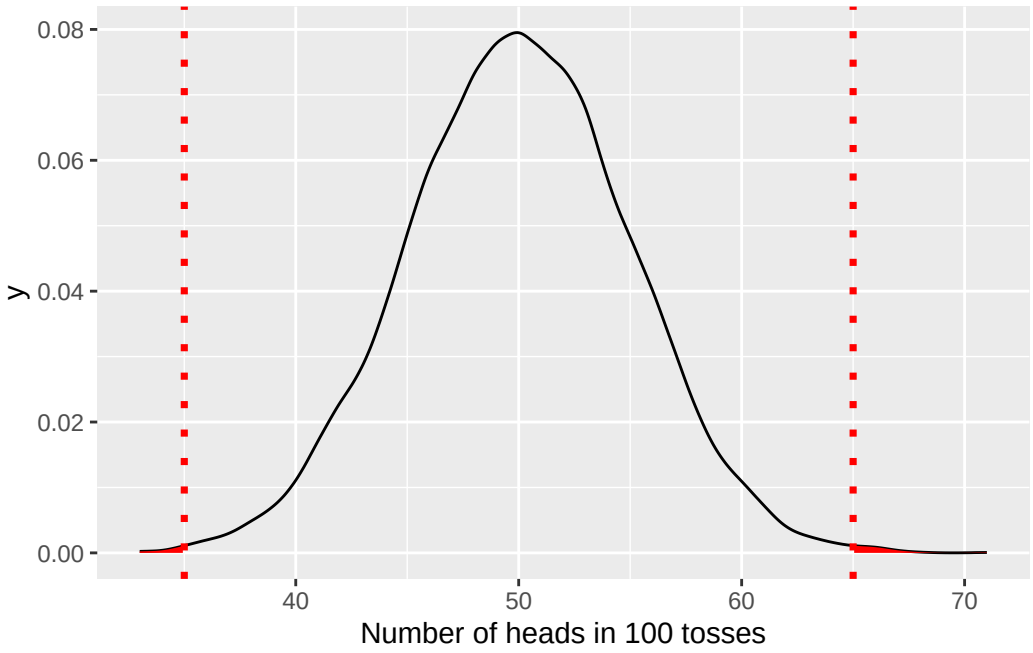

Figure 2.16: Null Distribution: Number of heads for fair coin tossed 100 times

```
sum(number_heads>=65 | number_heads<=35)/T
```

```
[1] 0.0041
```

Conclusion: We said at the outset that we wanted a significance level of 0.01, or 1%, for our test before we accused our friend of cheating. Based on our simulations, there is a 0.4% chance that we'd obtain the result we did, or something more extreme, if the coin was fair. Therefore, we have no choice but to *reject our null hypothesis in favor of the alternative*. Our friend has some explaining to do!

Before we move on, we can compare the simulated result we just developed to one based on a theoretical distribution. This can be done using the `prop.test()` function to test a proportion. The formal test confirms our conclusion.

```
prop.test(65, 100, p = 0.5,
          alternative = c("two.sided"),
          conf.level = 0.99)
```

```
    1-sample proportions test with continuity correction

data:  65 out of 100
X-squared = 8.41, df = 1, p-value = 0.003732
alternative hypothesis: true p is not equal to 0.5
99 percent confidence interval:
 0.5162768 0.7643236
sample estimates:
   p
0.65
```

The nice thing about a simulation study like the one above is that it allows the user to explore how changes in the underlying procedure might affect the outcome. We'll next consider two pieces that of the simulation study and how they affect the p-value: the *sample size* (the number of coin flips) and the *effect size* (the observed deviation from 50% heads).

What happens if we increase/decrease the number of coin flips in our experiment, but keep the effect size the same, fixed at 65% heads? If we only flipped the coin 10 times, would 6-7 heads be improbable to witness from a fair coin? If we flipped the coin 1000 times, would 650 heads be *more* or *less* improbable than that same ratio in 10 tosses? In other words, which of these situations would entail a smaller p-value? We hope that the reader now has some intuition to answer this question. If not, we encourage them to answer it by altering the value of `n` in the simulation code, and seeing how the changes affect the distribution of the null hypothesis.

What happens if we fix the sample size at 100 tosses and decrease the effect size from 65 heads to 60 heads? We've already generated the data to answer this question - our p-value would *increase* because it would be *more* probable to obtain a smaller effect size from a fair coin. On the flip side (pun intended) the p-value would *decrease* for a larger effect size.

2.4.1 One-Sample T-Test

If we want to test whether the mean of continuous variable is equal to hypothesized value, we can use the `t.test()` function. The following code tests whether the average sale price of homes from Ames, Iowa over the data time period is $178,000. For now, we'll use the classic $\alpha = 0.05$ as our significance level. If we have enough evidence to reject this null hypothesis, we will conclude that the mean sale price is significantly different than $178,000 for a two-tailed test (the default):

```r
t.test(ames$Sale_Price, mu = 178000)
```

```
    One Sample t-test

data:  ames$Sale_Price
t = 1.8945, df = 2929, p-value = 0.05825
alternative hypothesis: true mean is not equal to 178000
95 percent confidence interval:
 177902.3 183689.9
sample estimates:
mean of x
 180796.1
```

Because our *p-value is greater than our alpha* level of 0.05, we *fail to reject* the null hypothesis. We do not quite have sufficient evidence to say the mean is different from 178,000.

If we're instead interested in testing whether the `Sale_Price` is *higher* than $178,000, we can specify this in the `alternative=` option.

```
t.test(ames$Sale_Price, mu = 178000, alternative = 'greater')
```

```
    One Sample t-test

data:  ames$Sale_Price
t = 1.8945, df = 2929, p-value = 0.02913
alternative hypothesis: true mean is greater than 178000
95 percent confidence interval:
 178367.7      Inf
sample estimates:
mean of x
 180796.1
```

In this second test, we see that we actually *do have enough evidence* to claim that the true mean is greater than $178,000 at the $\alpha = 0.05$ level.

2.5 Two-Sample t-tests

If we have a hypothesis about a difference in the means of two groups of observations, a **two-sample t-test** can tell us whether that difference is statistically significant. By *statistically significant*, we mean the observed difference in sample means is greater than what we would

expect to find if the population means were truly equal. In other words, statistical significance is a phrase that describes when our p-value falls below our significance level, α. Typically, the groups of interest are formed by levels of a binary variable, and the t-test is a way of testing whether there is a relationship between that binary variable and the continuous variable.

To conduct a two-sample t-test, our data should satisfy 3 fundamental assumptions:

The observations are independent

The data from each group is normally distributed

The variances for each group are equal

If our data does not satisfy these assumptions, we must adapt our test to the situation. If the 3^{rd} assumption of equal variances is not met, we simply add the option `var.equal=F` to the `t.test()` function to use the Welch or Satterthwaite approximation to degrees of freedom (it's becoming increasingly common for practitioners to use this option even when variances are equal).

If the 2^{nd} assumption is not met, one must opt for a *nonparametric* test like the Mann-Whitney-U test (also called the Mann–Whitney–Wilcoxon or the Wilcoxon rank-sum test).

The 1^{st} assumption is not easily checked unless the data is generated over time (time-series) and is instead generally implied by careful data collect and the application of domain expertise.

2.5.1 Testing Normality of Groups

We can test the normality assumption either graphically or through formal statistical tests. The best graphical test of normality is a QQ-Plot, though histograms are often used as well. We saw in Figures Figure **??** and Figure **??** that the distribution of `Sale_Price` was not perfectly normal, however the deviations from normality were not egregious. In Figure Figure **??** we again examine the normality of Sale Price, this time for each value of `Central_Air`:

```
ggplot(data = ames, aes(sample = Sale_Price, color = Central_Air)) +
     stat_qq() +
     stat_qq_line()
```

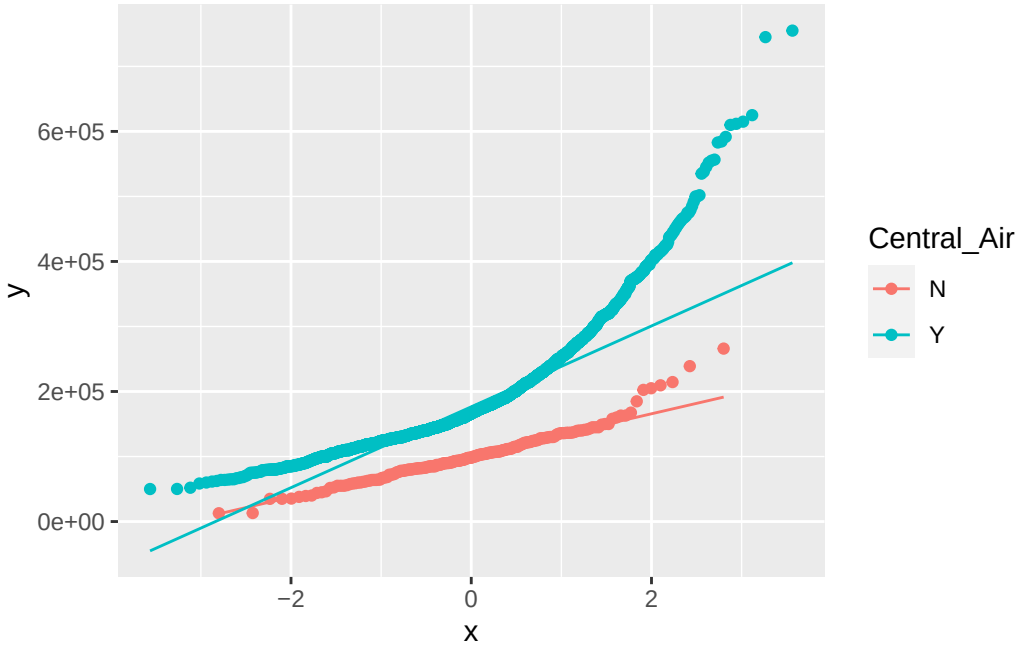

Figure 2.17: QQ-Plot: Quantiles of Sale_Price vs. quantiles of a theoretical normal distribution with same mean and standard deviation, for each level of Central_Air. Conclusion: Sale_Price is *not* normally distributed for Central_Air='Yes' and is *more* normally distributed for Central_Air = 'No'.

For formal tests of normality, we most often use the Shapiro-Wilk test, although many other formal tests exist, each with their own advantages and disadvantages. All of these tests have the null hypothesis of normality:

$$H_0: \text{ the data is normally distributed}$$

$$H_a: \text{ the data is NOT normally distributed}$$

We conduct formal tests as follows:

```
shapiro.test(ames$Sale_Price[ames$Central_Air=='Y'])
```

```
    Shapiro-Wilk normality test

data:  ames$Sale_Price[ames$Central_Air == "Y"]
W = 0.86295, p-value < 2.2e-16
```

```r
shapiro.test(ames$Sale_Price[ames$Central_Air=='N'])
```

```
    Shapiro-Wilk normality test

data:  ames$Sale_Price[ames$Central_Air == "N"]
W = 0.95315, p-value = 4.739e-06
```

The formal test rejects the null hypotheses of normality and confirms what we saw in our visual analysis.

For the sake of illustration, we will proceed in this example under the assumption of normality (JUST for illustration), though a non-parametric approach will also be demonstrated later in our analysis in Section Section **??**.

2.5.2 Testing Equality of Variances

We can conduct a formal test to confirm that the 3^{rd} assumption is met. This test for equal variances is known as an F-Test. The null hypothesis is that the variances are equal, the alternative being that they are not:

$$H_0 : \sigma_1^2 = \sigma_2^2$$
$$H_a : \sigma_1^2 \neq \sigma_2^2$$

The F-Test is invoked with the `var.test` function, which takes as input a `formula`. A formula is an R object most often created using the `~` operator, for example `y ~ x + z`, interpreted as a specification that the response `y` is to be predicted with the inputs `x` and `z`. For our present discussion on two-sample t-tests, we might think of predicting our continuous attribute with our binary attribute, provided the means are different between the two groups.

The following code checks whether the distributions of Sale_Price for houses with and without central air have the same variance.

```r
var.test(Sale_Price ~ Central_Air, data = ames)
```

```
    F test to compare two variances

data:  Sale_Price by Central_Air
F = 0.2258, num df = 195, denom df = 2733, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal to 1
```

```
95 percent confidence interval:
 0.1854873 0.2800271
sample estimates:
ratio of variances
         0.2257977
```

They do not. Thus, we must opt for the `var.equal=FALSE` option in the `t.test()` procedure.

2.5.3 Testing Equality of Means

Assuming the first two assumptions are met, the two-sample t-test is performed by including a binary grouping variable with into the `t.test()` function. Below we test whether the mean sales price is different for houses with and without `Central_Air`, and we include the `var.equal=FALSE` option because we determined that the variances of the two groups were unequal in Section Section **??**. The null hypothesis for the t-test is that the means of the two groups are equal:

$$H_0 : \mu_1 = \mu_2$$
$$H_a : \mu_1 \neq \mu_2$$

```
t.test(Sale_Price ~ Central_Air, data = ames, var.equal = FALSE)
```

```
    Welch Two Sample t-test

data:  Sale_Price by Central_Air
t = -27.433, df = 336.06, p-value < 2.2e-16
alternative hypothesis: true difference in means between group N and group Y is not equal to
95 percent confidence interval:
 -90625.69 -78498.92
sample estimates:
mean in group N mean in group Y
       101890.5        186452.8
```

Our final conclusion from the t-test is the *rejection of the null hypothesis* and the conclusion that *houses with and without central air should be expected to sell for different prices.*

2.5.4 Mann-Whitney-Wilcoxon Test

As we pointed out in Section Section **??**, the distribution of `Sale_Price` was not precisely normal. The most principled way to proceed in this case would be with a non-parametric test. The Mann-Whitney-Wilcoxon test is not identical to the t-test in its null and alternative hypotheses, but it aims to answer the same question about an association between the binary attribute and the continuous attribute.

The null and alternative hypotheses for this test are typically defined as follows, so long as the two groups *are* identically distributed (having the same *shape* and variance):

$$H_0 : \text{ the medians of the two groups are equal}$$

$$H_a : \text{ the medians of the two groups are NOT equal}$$

If those identical distributions are also symmetric, then Mann-Whitney-Wilcoxon can be interpretted as testing for a difference in means. When the data is not identically distributed, or when the distributions are not symmetric, Mann-Whitney-Wilcoxon is a test of **dominance** between distributions. **Distributional dominance** is the notion that one group's distribution is located at larger values than another, probabilistically speaking. Formally, a random variable A has dominance over random variable B if $P(Ax) \geq P(B \geq x)$ for all x, and for some x, $P(A \geq x) > P(B \geq x)$.

We summarize this information in the following table:

Conditions	Interpretation of Significant Mann-Whitney-Wilcoxon Test
Group distributions are identical in shape, variance, and symmetric	Difference in means
Group distributions are identical in shape, variance, but not symmetric	Difference in medians
Group distributions are identical in shape, variance, but not symmetric	Difference in location. (distributional dominance)

To perform this test, we use the `wilcox.test()` function, whose inputs are identical to the `t.test()` function.

```
wilcox.test(Sale_Price ~ Central_Air, data = ames)
```

```
    Wilcoxon rank sum test with continuity correction

data:  Sale_Price by Central_Air
```

```
W = 63164, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

Thus, we make the same conclusion with our non-parametric test. Houses with and without central air should be expected to sell for different prices.

3 Introduction to ANOVA and Linear Regression

This Chapter aims to answer the following questions:

What is a predictive model versus an explanatory model?

How to perform an honest assessment of a model.

How to estimate associations.

Continuous-Continuous

Continuous-Categorical

Pearson's correlation

Test of Hypothesis

Effect of outliers

Correlation Matrix

How to perform ANOVA.

Testing assumptions

Kruskal-Wallis

Post-hoc tests

How to perform Simple Linear Regression.

Assumptions

Inference

In this chapter, we introduce one of the most commonly used tools in data science: the linear model. We'll start with some basic terminology. A **linear model** is an equation that typically takes the form

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{x_1} + \cdots + \beta_k \mathbf{x_k} + \varepsilon \tag{3.1}$$

The left-hand side of this equation, $\mathbf{y}$ is equivalently called the **target**, **response**, or **dependent** variable. The right-hand side is a linear combination of the columns $\{\mathbf{x_i}\}_{i=1}^{k}$ which are commonly referred to as **explanatory**, **input**, **predictor**, or **independent** variables.

3.1 Prediction vs. Explanation

The purpose of a linear model like Equation (Equation **??**) is generally two-fold:

The model is **predictive** in that it can estimate the value of y for a given combination of the x attributes.

The model is **explanatory** in that it can estimate how y changes for a unit increase in a given x attribute, holding all else constant (via the slope parameters β).

However, it's common for *one* of these purposes to be more aligned with the specific goals of your project, and it is common to approach the building of such a model differently for each purpose.

In predictive modeling, you are most interested in how much error your model has on *holdout data*, that is, validation or test data. This is a notion that we introduce next in Section Section **??**. If good predictions are all you want from your model, you are unlikely to care how many variables (including polynomial and interaction terms) are included in the final model.

In explanatory modeling, you foremost want a model that is simple to interpret and doesn't have too many input variables. It's common to avoid many polynomial and interaction terms for explanatory models. While the error rates on holdout data will still be useful reporting metrics for explanatory models, it will be more important to craft the model for ease of interpretation.

3.2 Honest Assessment

When performing predictive or explanatory modeling, we *always* divide our data into subsets for training, validation, and/or final testing. Because models are prone to discovering small, spurious patterns on the data that is used to create them (the *training data*), we set aside the *validation* and *testing data* to get a clear view of how they might perform on new data that they've never seen before. This is a concept that will be revisited several times throughout this text, highlighting its importance to honest assessment of models.

There is no single right answer for how this division should occur for every data set - the answer depends on a multitude of factors that are beyond the scope of our present discussion. Generally speaking, one expects to keep about 70% of the data for model training purposes, and the remaining 30% for validation and testing. These proportions may change depending on the amount of data available. If one has millions of observations, they can often get away with a much smaller proportion of training data to reduce computation time and increase confidence in validation. If one has substantially fewer observations, it may be necessary to increase the training proportion in order to build a sound model - trading validation confidence for proper training.

Below we demonstrate one technique for separating the data into just two subsets: training and test. These two subsets will suffice for our analyses in this text. We'll use 70% of our data for the training set and the remainder for testing.

Since we are taking a random sample, each time you run this functions you will get a different result. This can be difficult for team members who wish to keep their analyses in sync. To avoid that variation of results, we can provide a "seed" to the internal random number generation process, which ensures that the randomly generated output is the same to all who use that seed.

The following code demonstrates sampling via the `tidyverse`. This method requires the use of an id variable. If your data set has a unique identifier built in, you may omit the first line of code (after `set.seed()`) and use that unique identifier in the third line.

```
library(tidyverse)

set.seed(123)

ames <- ames %>% mutate(id = row_number())

train <- ames %>% sample_frac(0.7)

test <- anti_join(ames, train, by = 'id')

dim(train)
```

```
[1] 2051   82
```

```
dim(test)
```

```
[1] 879  82
```

3.3 Bivariate EDA

As stated in Chapter Chapter **??**, exploratory data analysis is the foundation of any successful data science project. As we move on to the discussion of modeling, we begin to explore *bivariate* relationships in our data. In doing so, we will often explore the input variables' relationships with the target. **Such exploration should only be done on the training data; we should never let insights from the validation or test data inform our decisions about modeling.**

Bivariate exploratory analysis is often used to assess relationships between two variables. An **association** or **relationship** exists when the expected value of one variable changes at different levels of the other variable. A **linear relationship** between two continuous variables can be inferred when the general shape of a scatter plot of the two variables is a straight line.

3.3.1 Continuous-Continuous Associations

Let's conduct a preliminary assessment of the relationship between the size of the house in square feet (via `Gr_Liv_Area`) and the `Sale_Price` by creating a scatter plot (only on the training data). Note that we call this a preliminary assessment because we should not declare a statistical relationship without a formal hypothesis test (see Section Section **??**).

```
ggplot(data = train) +
  geom_point(mapping = aes(x = Gr_Liv_Area, y = Sale_Price/1000)) +
  labs(y = "Sales Price (Thousands $)", x = "Greater Living Area (Sqft)")
```

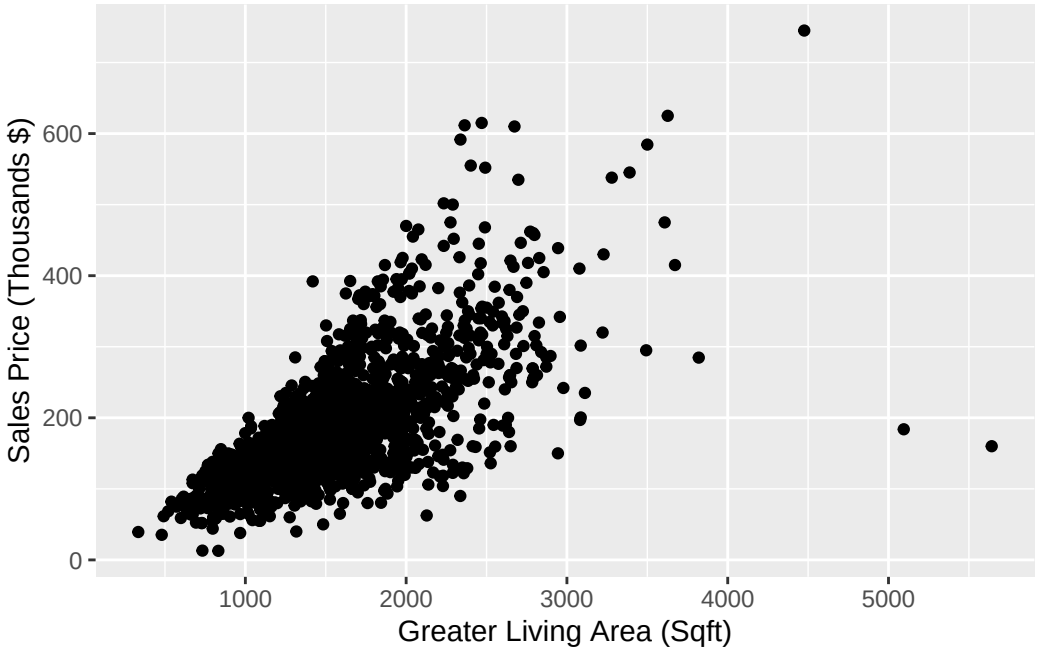

Figure 3.1: Scatter plot demonstrating a positive linear relationship

3.3.2 Continuous-Categorical Associations

We'll also revisit the plots that we created in Section Section **??**, this time being careful to use only our training data since our goal is eventually to use a linear model to predict `Sale_Price`.

We start by exploring the relationship between the external quality rating of the home (via the ordinal variable `Exter_Qual` and the `Sale_Price`).

The simplest graphic we may wish to create is a bar chart like Figure Figure **??** that shows the average sale price of homes with each value of exterior quality.

```
ggplot(train) +
  geom_bar(aes(x=Exter_Qual,y= Sale_Price),
           position = "dodge", stat = "summary", fun = "mean") +
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE)) # Modify formatti
```

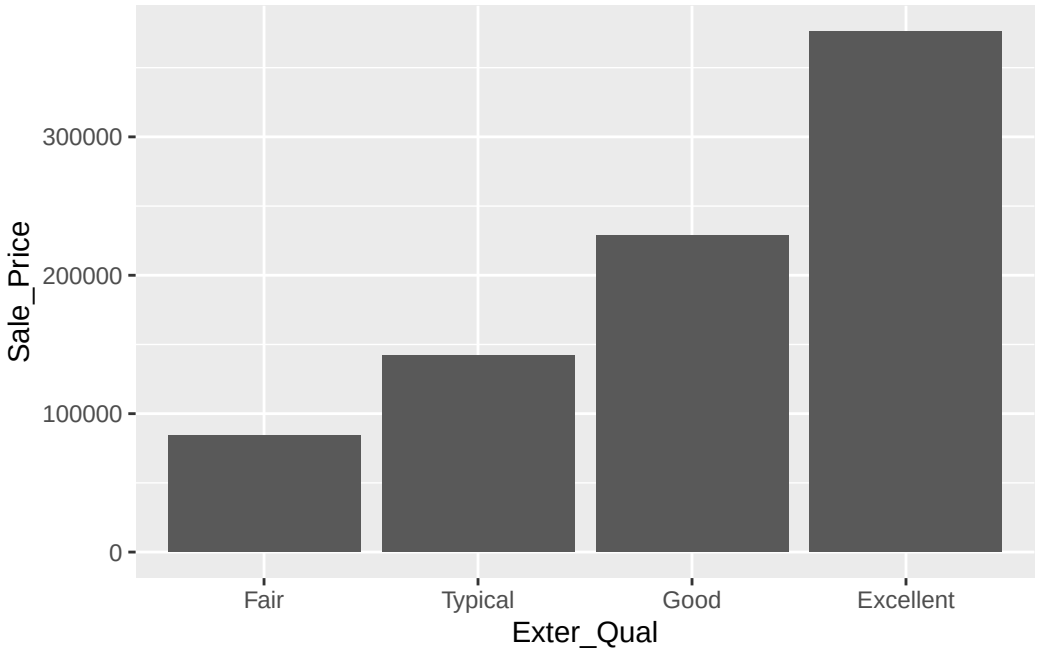

Figure 3.2: Bar Chart Comparing Average Sale Price of Homes with each Level of Exterior Quality

This gives us the idea that there may be an association between these two attributes, but it can be tricky to rely solely on this graph without exploring the overall distribution in sale price for each group. While this chart is great for the purposes of *reporting* (once we've verified the

relationship), it's not the best one for exploratory analysis. The next two charts allow us to have much more information on one graphic.

The frequency histogram in Figure Figure ?? allows us to see that much fewer of the homes have a rating of `Excellent` versus the other tiers, a fact that makes it difficult to compare the distributions. To normalize that quantity and compare the raw probability densities, we can change our axes to density (which is analogous to percentage) and employ a kernel density estimator with the `geom_density` plot as shown in Figure @ref(fig:overhistogramdensitykernel). We can then clearly see that as the exterior quality of the home "goes up" (in the ordinal sense, not in the linear sense), the sale price of the home also increases.

```
ggplot(train,aes(x=Sale_Price/1000, fill=Exter_Qual)) +
  geom_histogram(alpha=0.2, position="identity") +
  labs(x = "Sales Price (Thousands $)")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

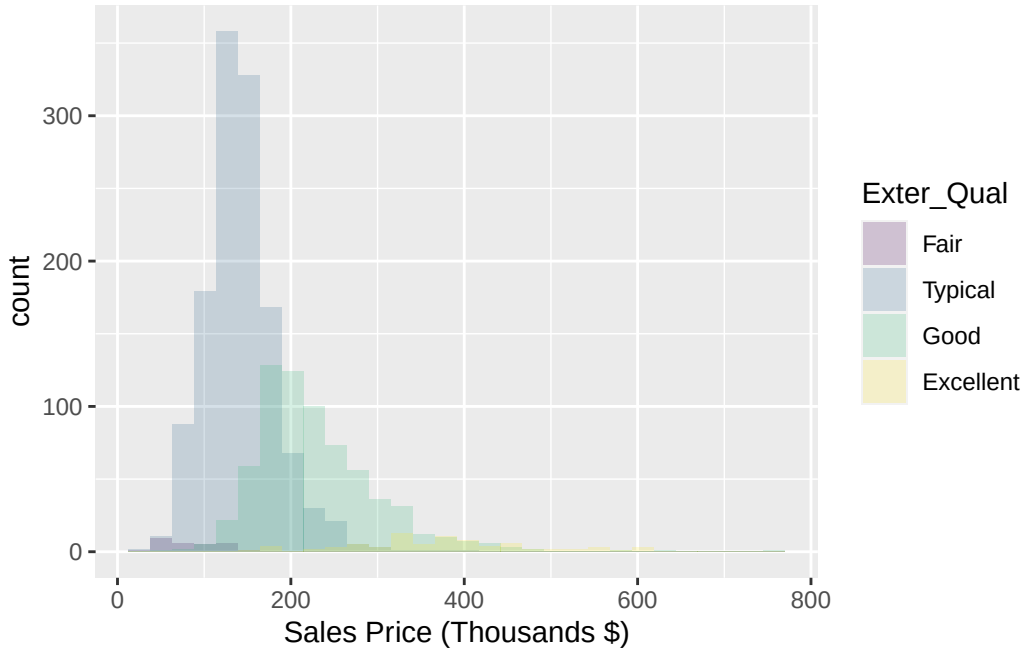

Figure 3.3: Histogram: Frequency of Sale_Price for varying qualities of home exterior

```
ggplot(ames,aes(x=Sale_Price/1000, fill=Exter_Qual)) +
  geom_density(alpha=0.2, position="identity") +
  labs(x = "Sales Price (Thousands $)")
```

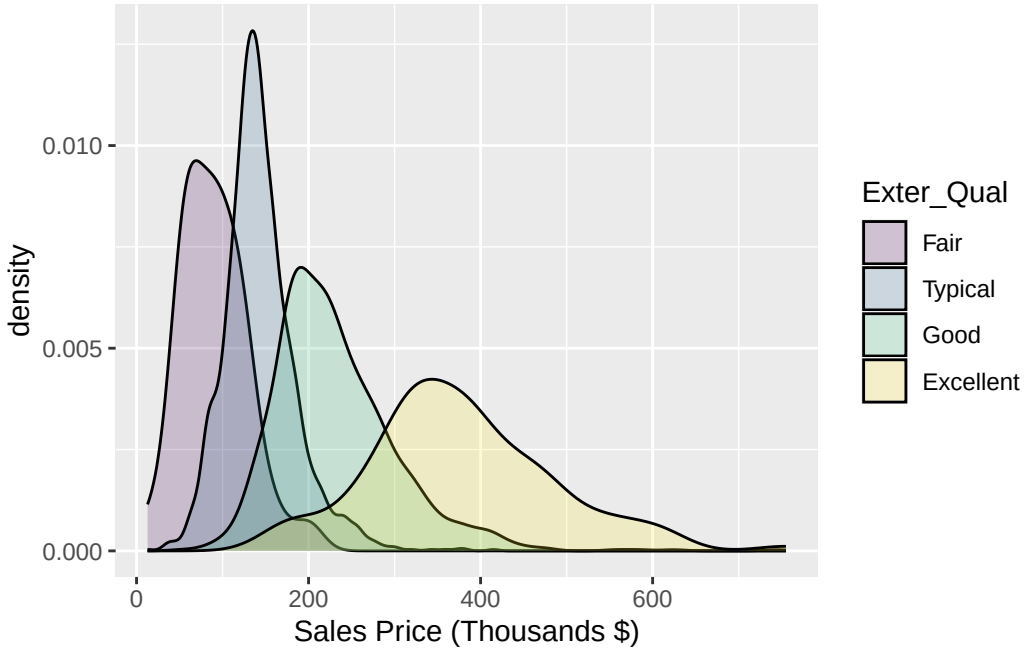

Figure 3.4: Histogram: Density of Sale_Price for varying qualities of home exterior

To further explore the location and spread of the data, we can create box-plots for each group using the following code:

```
ggplot(data = train, aes(y = Sale_Price/1000, x = `Exter_Qual`, fill = `Exter_Qual`)) +
  geom_boxplot() +
  labs(y = "Sales Price (Thousands $)", x = "Exterior Quality Category") +
  stat_summary(fun = mean, geom = "point", shape = 20, size = 5, color = "red", fill = "re
  scale_fill_brewer(palette="Blues") + theme_classic() + coord_flip()
```

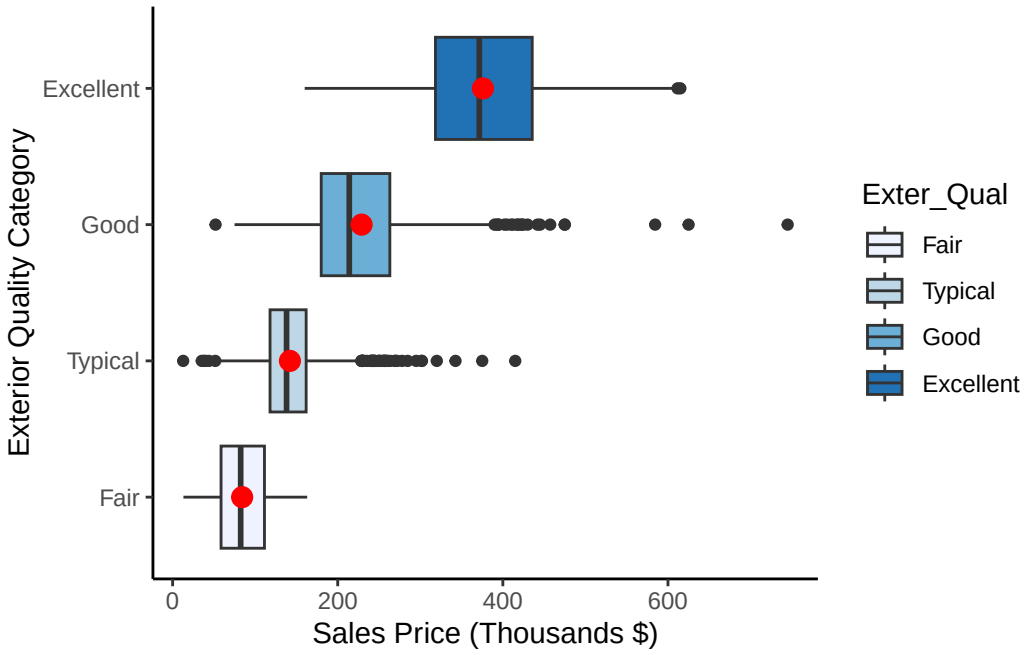

Figure 3.5: Box Plots of Sale_Price for each level of Exter_Qual

Notice that we've highlighted the mean on each box-plot for the purposes of comparison. We now have a hypothesis that we may want to formally test. After all, it is not good practice to look at Figures Figure **??**) and Figure **??**) and *declare* that a statistical difference exists. While we do, over time, get a feel for which visually apparent relationships turn out to be statistically significant, it's *imperative* that we conduct formal testing before declaring such insights to a colleague or stakeholder.

If we want to test whether the `Sale_Price` is different for the different values of `Exter_Qual`, we have to reach for the multi-group alternative to the 2-sample t-test. This is called **Analysis of Variance**, or **ANOVA** for short.

3.4 One-Way ANOVA

One-way ANOVA aims to determine whether there is a difference in the mean of a continuous attribute across levels of a categorical attribute. Sound like a two-sample t-test? Indeed, it's the extension of that test to more than two groups. Performing ANOVA with a binary input variable is mathematically identical to the two-sample t-test, as are it's **assumptions**:

The observations are independent

The model residuals are normally distributed

The variances for each group are equal

A one-way ANOVA refers to a single hypothesis test, which is $H_0 : \mu_1 = \mu_2 = ...\mu_k$ for a predictor variable with k levels against the alternative of at least one difference. Although a one-way ANOVA is designed to assess whether or not there is a significant difference within the mean values of the response with respect to the different levels of the predictor variable, we can draw some parallel to the regression model. For example, if we have k=4, then we can let (x_a), (x_b), and (x_c) be 3 reference-coded dummy variables for the levels: `a`, `b`, `c`, and `d`. Note that we only have 3 dummy variables because one gets left out for the reference level, in this case it is `d`. The linear model is of the following form:

$$y = \beta\_0 + \beta\_ax_a + \beta\_bx_b + \beta\_cx_c + \varepsilon \tag{3.2}$$

If we define x_a as 1 if the observation belongs to level `a` and 0 otherwise, and the same definition for x_b and x_c, then this is called *reference-level coding* (this will change for effects-level coding). The predicted values in Equation **??** is basically the predicted mean of the response within the 4 levels of the predictor variable.

β_0 represents the mean of reference group, group `d`.

$\beta_a, \beta_b, \beta_c$ all represent the *difference* in the respective group means compared to the reference level. Positive values thus reflect a group mean that is higher than the reference group, and negative values reflect a group mean lower than the reference group.

ε is called the **error**.

A *one-way* ANOVA model only contains a single input variable of interest. Equation Equation **??**, while it has 3 dummy variable inputs, only contains a single nominal attribute. In Chapter **??**, we will add more inputs to the equation via two-way ANOVA and multivariate regression models.

ANOVA is used to test the following hypothesis:

$$H_0 : \beta_a = \beta_b = \beta_c = 0 \quad \text{(i.e. all group means are equal)}$$

$$H_0 : \beta_a \neq 0 \text{ or } \beta_b \neq 0 \text{ or } \beta_c \neq 0 \quad \text{(i.e. at least one is different)}$$

Both the `lm()` function and the `aov()` function will provide the p-values to test the hypothesis above, the only difference between the two functions is that `lm()` will also provide the user with the coefficient of determinination, R^2, which tells you how much of the variation in y is accounted for by your categorical input.

```
ames_lm <- lm(Sale_Price ~ Exter_Qual, data = train)

anova(ames_lm)
```

```
Analysis of Variance Table

Response: Sale_Price
             Df     Sum Sq    Mean Sq F value    Pr(>F)
Exter_Qual    3 6.6913e+12 2.2304e+12  701.83 < 2.2e-16 ***
Residuals  2047 6.5054e+12 3.1780e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(ames_lm)
```

```
Call:
lm(formula = Sale_Price ~ Exter_Qual, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-215904  -32910   -6147   24793  516090

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    207785       3176  65.416  < 2e-16 ***
Exter_Qual.L   215078       8353  25.749  < 2e-16 ***
Exter_Qual.Q    44553       6353   7.013 3.15e-12 ***
Exter_Qual.C     6994       3308   2.114   0.0346 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 56370 on 2047 degrees of freedom
Multiple R-squared:  0.507, Adjusted R-squared:  0.5063
F-statistic: 701.8 on 3 and 2047 DF,  p-value: < 2.2e-16
```

```r
# ames_aov <- aov(Sale_Price ~ Exter_Qual, data = train) # Same thing with aov() function
# summary(ames_aov) # R-squared not reported here.
```

The p-value for the ANOVA hypothesis that all the groups have the same mean sale price is incredibly small, at 2.2×10^{-16}. This means it is extraordinarily improbable that we would have observed these differences in means, or a more extreme difference, if the population group means were equal. Thus, we reject our null hypothesis and conclude that there is an association between the exterior quality of a home and the price of the home.

We note, based on the R^2 statistics, that the exterior quality rating can account for almost half the variation in sales price! Adjusted R^2 is a statistic that takes into account the number of variables in the model. The difference between R^2 and adjusted R^2 will be more thoroughly discussed in Chapter Chapter **??**.

We can also confirm what we know about the predictions from ANOVA, that there are only k unique predictions from an ANOVA with k groups (the predictions being the group means), using the `predict` function.

```
train$pred_anova <- predict(ames_lm, data = train)

train$resid_anova <- resid(ames_lm, data = train)

(model_output = train %>% dplyr::select(Sale_Price, pred_anova, resid_anova))
```

```
# A tibble: 2,051 x 3
   Sale_Price pred_anova resid_anova
        <int>      <dbl>       <dbl>
 1     232600    228910.       3690.
 2     166000    228910.     -62910.
 3     170000    142107.      27893.
 4     252000    228910.      23090.
 5     134000    142107.      -8107.
 6     164700    228910.     -64210.
 7     193500    142107.      51393.
 8     118500    142107.     -23607.
 9      94000    142107.     -48107.
10     111250    142107.     -30857.
# ... with 2,041 more rows
```

3.4.1 Testing Assumptions

We can use the default plots from the `lm()` function to check the normality assumption.

```
par(mfrow=c(2,2))
plot(ames_lm)
par(mfrow=c(1,1))
```

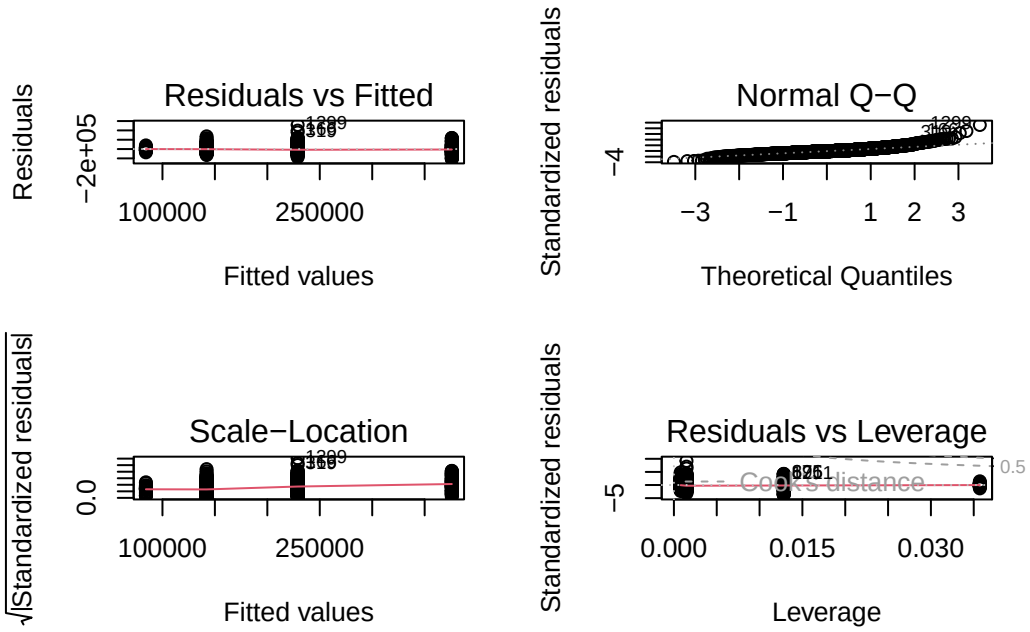

Figure 3.6: Of the 4 default plots from lm(), we are presently interested in the top-right QQ plot that tests our assumption of normally distributed residuals.

In the top-right plot in Figure Figure **??** we verify again the *approximate* normality of sale price. To test for the third assumption of equal variances, we opt for a formal test like Levene's (which depends on normality and can be found in the `car` package) or Fligner's (which does not depend on normality and exists in the `stats` package). In both cases, the null hypothesis is equal variances:

$$H_0 : \sigma_a^2 = \sigma_b^2 = \sigma_c^2 = \sigma_d^2 \quad \text{i.e., the groups have equal variance}$$

$$H_a : \text{at least one group's variance is different}$$

```
library(car)
library(stats)

leveneTest(Sale_Price ~ Exter_Qual, data = train) # Most popular, but depends on Normality
```

```
Levene's Test for Homogeneity of Variance (center = median)
        Df F value    Pr(>F)
group    3  76.879 < 2.2e-16 ***
      2047
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
fligner.test(Sale_Price ~ Exter_Qual, data = train) # DOES NOT depend on Normality
```

```
    Fligner-Killeen test of homogeneity of variances

data:  Sale_Price by Exter_Qual
Fligner-Killeen:med chi-squared = 206.26, df = 3, p-value < 2.2e-16
```

And in both cases, we're forced to reject the null hypothesis of equal variances. A non-parametric version of the ANOVA test, the Kruskal-Wallis test, is more suitable to this particular case. Non-parametric tests do not have the same *statistical power* to detect differences between groups. **Statistical power** is the probability of detecting an effect, if there is a true effect present to detect. We should opt for these tests in situations where our data is ordinal or otherwise violates the assumptions of normality or equal variances in ways that cannot be fixed by logarithmic or other similar transformation.

3.4.2 Kruskal-Wallis

The Kruskal-Wallis test, proposed in 1952, is equivalent to a parametric one-way ANOVA where the data values have been replaced with their ranks (i.e. largest value = 1, second largest value = 2, etc.). When the data *is not* normally distributed but *is* identically distributed (having the same shape and variance), the Kruskal-Wallis test can be considered a test for differences in medians. If those identical distributions are also symmetric, then Kruskal-Wallis can be interpretted as testing for a difference in means. When the data is not identically distributed, or when the distributions are not symmetric, Kruskal-Wallis is a test of **dominance** between distributions. **Distributional dominance** is the notion that one group's distribution is located at larger values than another, probabilistically speaking. Formally, a random variable A has dominance over random variable B if $P(A \geq x) \geq P(B \geq x)$ for all x, and for some x, $P(A \geq x) > P(B \geq x)$.

We summarize this information in the following table:

Conditions	Interpretation of Significant Kruskal-Wallis Test
Group distributions are identical in shape, variance, and symmetric	Difference in means

Group distributions are identical in shape, variance, but not symmetric	Difference in medians
Group distributions are not identical in shape, variance, and are not symmetric	Difference in location. (distributional dominance)

Implementing the Kruskal-Wallis test in R is simple:

```
kruskal.test(Sale_Price ~ Exter_Qual, data = train)
```

```
    Kruskal-Wallis rank sum test

data:  Sale_Price by Exter_Qual
Kruskal-Wallis chi-squared = 975.98, df = 3, p-value < 2.2e-16
```

Our conclusion would be that the distribution of sale price is different across different levels of exterior quality.

3.5 ANOVA Post-hoc Testing

After performing an ANOVA and learning that there is a difference between the groups of data, our next natural question ought to be *which groups of data are different, and how?* In order to explore this question, we must first consider the notion of **experimentwise error**. When conducting multiple hypothesis tests simultaneously, the experimentwise error rate is the proportion of time you expect to make an error in at least one test.

Let's suppose we are comparing grocery spending on 4 different credit card rewards programs. If we'd like to compare the rewards programs pairwise, that entails 6 different hypothesis tests (each is a two-sample t-test). If we keep a confidence level of $\alpha = 0.05$ and subsequently view "being wrong in one test" as a random event happening with probability $p = 0.05$ then our probability of being wrong *in at least* one test out of 6 could be as great as 0.26!

To control this experiment-wise error rate, we must lower our significance thresholds to account for it. Alternatively, we can view this as an adjustment of our p-values higher while keeping our significance threshold fixed as usual. This is typically the approach taken, as we prefer to fix our significance thresholds in accordance with previous literature or industry standards. There are many methods of adjustment that have been proposed over the years for this purpose. Here, we consider two popular methods: Tukey's test for pairwise comparisons and Dunnett's test for control group comparisons. If the reader finds themselves in a situation that doesn't fit the prescription of either of these methods, we suggest looking next at the modified Bonferroni correction or the notion of false discovery rates proposed by Benjamini and Hochberg in 1995.

3.5.1 Tukey-Kramer

If our objective is to compare each group to every other group then Tukey's test of honest significant differences, also known as the Tukey-Kramer test is probably the most widely-available and popular corrections in practice. However, it should be noted that Tukey's test *should not* be used if one does not plan to make all pairwise comparisons. If only a subset of comparisons are of interest to the user (like comparisons only to a control group) then one should opt for the Dunnett or a modified Bonferroni correction.

To employ Tukey's HSD in R, we must use the `aov()` function to create our ANOVA object rather than the `lm()` function. The output of the test shows the difference in means and the p-value for testing the null hypothesis that the means are equal (i.e. that the differences are equal to 0).

```
ames_aov <- aov(Sale_Price ~ Exter_Qual, data = train)
tukey.ames <- TukeyHSD(ames_aov)
print(tukey.ames)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Sale_Price ~ Exter_Qual, data = train)

$Exter_Qual
                        diff       lwr       upr p adj
Typical-Fair       57887.91  30194.31  85581.52 5e-07
Good-Fair         144690.25 116739.87 172640.63 0e+00
Excellent-Fair    291684.79 259752.41 323617.16 0e+00
Good-Typical       86802.34  79910.03  93694.64 0e+00
Excellent-Typical 233796.87 216886.62 250707.12 0e+00
Excellent-Good    146994.54 129666.98 164322.10 0e+00
```

```
par(mar=c(4,10,4,2))
plot(tukey.ames, las = 1)
```

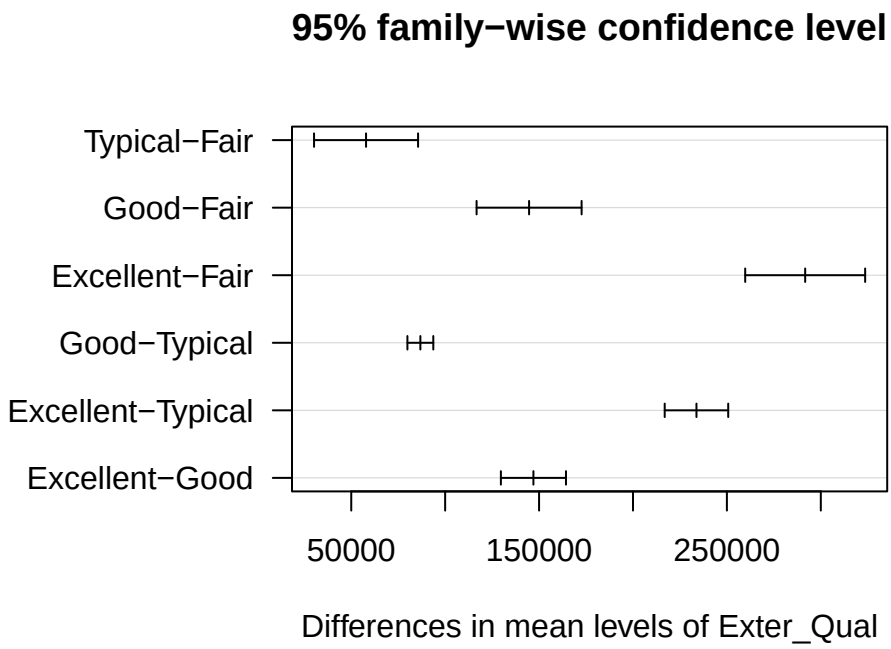

Figure 3.7: Confidence intervals for mean differences adjusted via Tukey-Kramer

The p-values in this table have been adjusted higher to account for the possible experimentwise error rate. For every pairwise comparison shown, we reject the null hypothesis and conclude that the mean sales price of the homes is different for each level of `Exter_Qual`. Furthermore, Figure Figure **??** shows us experiment-wise (family-wise) adjusted confidence intervals for the differences in means for each pair. The plot option `las=1` guides the axis labels. Type `?par` for a list of plot options for base R, including an explanation of `las`.

3.5.2 Dunnett's Test

If the plan is to make fewer comparisons, specifically just $k-1$ comparisons where k is the number of groups in your data (indicating you plan to compare all the groups to one specific group, usually the control group), then Dunnett's test would be preferrable to the Tukey-Kramer test. If all pairwise comparisons are not made, the Tukey-Kramer test is overly conservative, creating a confidence level that is much lower than specified by the user. Dunnett's test factors in fewer comparisons and thus should not be used for tests of all pairwise comparisons.

To use Dunnett's test, we must add the `DescTools` package to our library. The control group to which all other groups will be compared is designated by the `control=` option.

```
library(DescTools)
DunnettTest(x = train$Sale_Price, g = train$Exter_Qual, control = 'Typical')
```

```
  Dunnett's test for comparing several treatments with a control :
    95% family-wise confidence level

$Typical
                        diff    lwr.ci    upr.ci    pval
Fair-Typical      -57887.91 -83628.55 -32147.28 2.6e-07 ***
Good-Typical       86802.34  80396.08  93208.59 < 2e-16 ***
Excellent-Typical 233796.87 218079.15 249514.60 < 2e-16 ***

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the output from Dunnett's test, we notice the p-value comparing `Fair` to `Typical` exterior qualities is lower than it was in the Tukey-Kramer test. This is consistent with our expectations for a test that isn't controlling for as many comparisons; it makes a smaller upward adjustment of p-values to satisfy a given experiment-wise error rate.

3.6 Pearson Correlation

ANOVA is used to formally test the relationship between a categorical variable and a continuous variable. To formally test the (linear) relationship between two continuous attributes, we introduce **Pearson correlation**, commonly referred to as simply **correlation**. Correlation is a number between -1 and 1 which measures the strength of a linear relationship between two continuous attributes.

Negative values of correlation indicate a *negative linear relationship*, meaning that as one of the variables increases, the other tends to decrease. Similarly, positive values of correlation indicate a *positive linear relationship* meaning that as one of the variables increases, the other tends to increase. Absolute values of correlation equal to 1 indicate a perfect linear relationship. For example, if our data had a column for "mile time in minutes" and a column for "mile time in seconds", these two columns would have a correlation of 1 due to the fact that there are 60 seconds in a minute. A correlation value near 0 indicates that the variables have no linear relationship.

It's important to emphasize that Pearson correlation is only designed to detect *linear* associations between variables. Even when a correlation between two variables is 0, the two variables