Data 606 Lab 4: The Normal Distribution

Kevin Kirby

In this lab, you'll investigate the probability distribution that is most central to statistics: the normal distribution. If you are confident that your data are nearly normal, that opens the door to many powerful statistical methods. Here we'll use the graphical tools of R to assess the normality of our data and also learn how to generate random numbers from a normal distribution.

Getting Started

Load packages

In this lab, we will explore and visualize the data using the **tidyverse** suite of packages as well as the **openintro** package.

Let's load the packages.

```
library(tidyverse)
library(openintro)
```

The data

This week you'll be working with fast food data. This data set contains data on 515 menu items from some of the most popular fast food restaurants worldwide. Let's take a quick peek at the first few rows of the data.

Either you can use `glimpse` like before, or `head` to do this.

```
library(tidyverse)
library(openintro)
library(e1071)
data("fastfood", package='openintro')
head(fastfood)
```

```
## # A tibble: 6 x 17
##   restaurant item      calories cal_fat total_fat sat_fat trans_fat cholesterol
##   <chr>      <chr>        <dbl>   <dbl>     <dbl>   <dbl>     <dbl>       <dbl>
## 1 Mcdonalds  Artisan G~     380      60         7       2         0          95
## 2 Mcdonalds  Single Ba~     840     410        45      17       1.5         130
## 3 Mcdonalds  Double Ba~    1130     600        67      27         3         220
## 4 Mcdonalds  Grilled B~     750     280        31      10       0.5         155
## 5 Mcdonalds  Crispy Ba~     920     410        45      12       0.5         120
## 6 Mcdonalds  Big Mac        540     250        28      10         1          80
## # i 9 more variables: sodium <dbl>, total_carb <dbl>, fiber <dbl>, sugar <dbl>,
## #   protein <dbl>, vit_a <dbl>, vit_c <dbl>, calcium <dbl>, salad <chr>
```

You'll see that for every observation there are 17 measurements, many of which are nutritional facts.

You'll be focusing on just three columns to get started: restaurant, calories, calories from fat.

Let's first focus on just products from McDonalds and Dairy Queen.

```
mcdonalds <- fastfood %>%
  filter(restaurant == "Mcdonalds")
dairy_queen <- fastfood %>%
  filter(restaurant == "Dairy Queen")

#Kevin addition
mcd_dq <- rbind(mcdonalds %>% mutate(restaurant = "McDonald's"),
                dairy_queen %>% mutate(restaurant = "Dairy Queen"))
```

1. Make a plot (or plots) to visualize the distributions of the amount of calories from fat of the options from these two restaurants. How do their centers, shapes, and spreads compare?

Answer:

I've included two charts that show calories from fat for both McDonalds and Dairy Queen one is a density plot distribution and the second is a box plot chart, one plot for each chain.

Combined Density Plot: This shows that both of them have remarkably fat calorie distributions. McDonald's has a center that's a higher calorie amount than Dairy Queen, which is show by yhr gentle nudge to the right. Dairy Queen has more volatility coming off the peak but also doesn't have the aggressive tail that McDonald's has. McDonald's has a wider spread, which is probably a function of being a much larger and more successful fast food place with more options.

Box Plots: shows pretty much the same story while also more clearly shwoing that that the middle of the respective chains calories from fat operate in the same sphere. The extra dots off the top for Mcdonalds shows the longer tail described above.

```
ggplot(mcd_dq, aes(x = cal_fat, fill = restaurant)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(name = "Brand", values = c('McDonald\'s' = 'skyblue', 'Dairy Queen' = 'purple')) +
  theme_minimal()
```
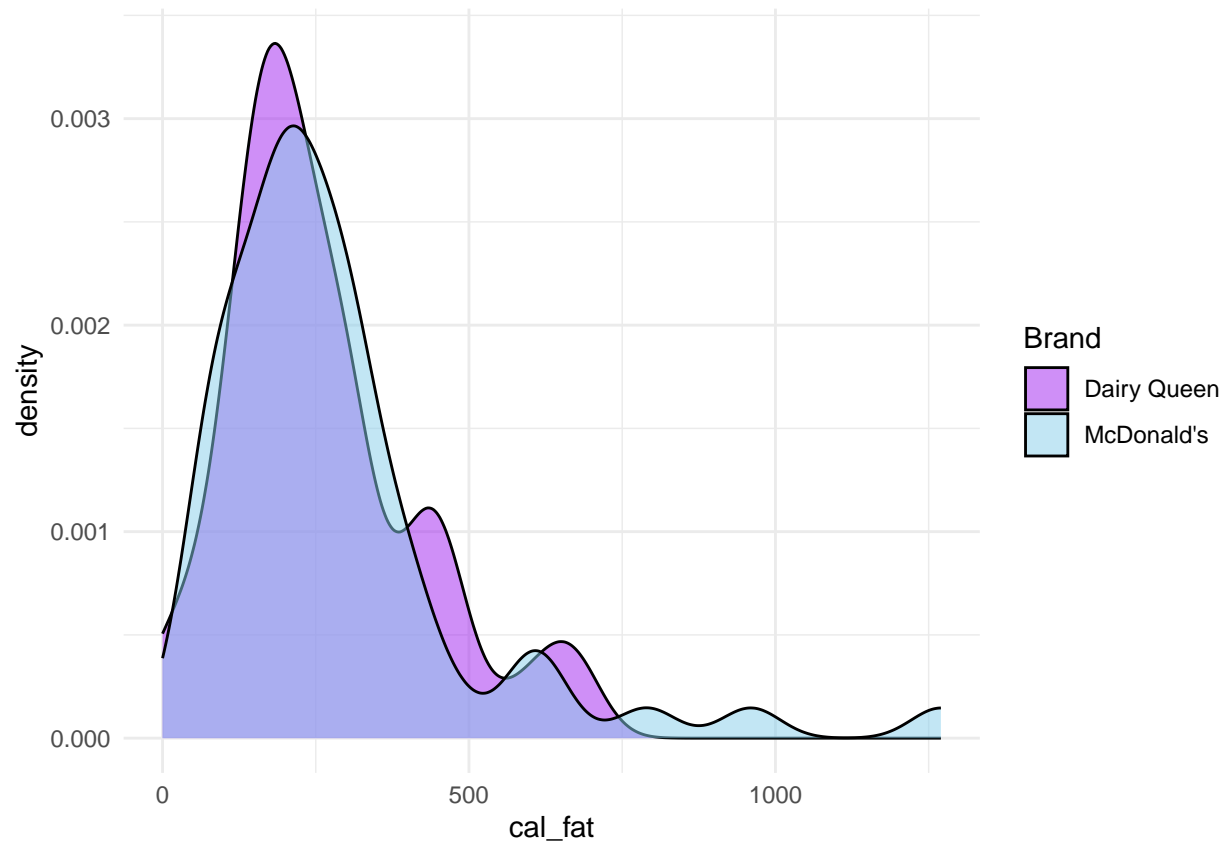
```
ggplot(mcd_dq, aes(x = restaurant, y = cal_fat, fill = restaurant)) +
  geom_boxplot() +
  labs(title = "Calories from Fat: McDonald's vs Dairy Queen",
       x = "Restaurant",
       y = "Calories from Fat") +
  theme_minimal()
```

Calories from Fat: McDonald's vs Dairy Queen
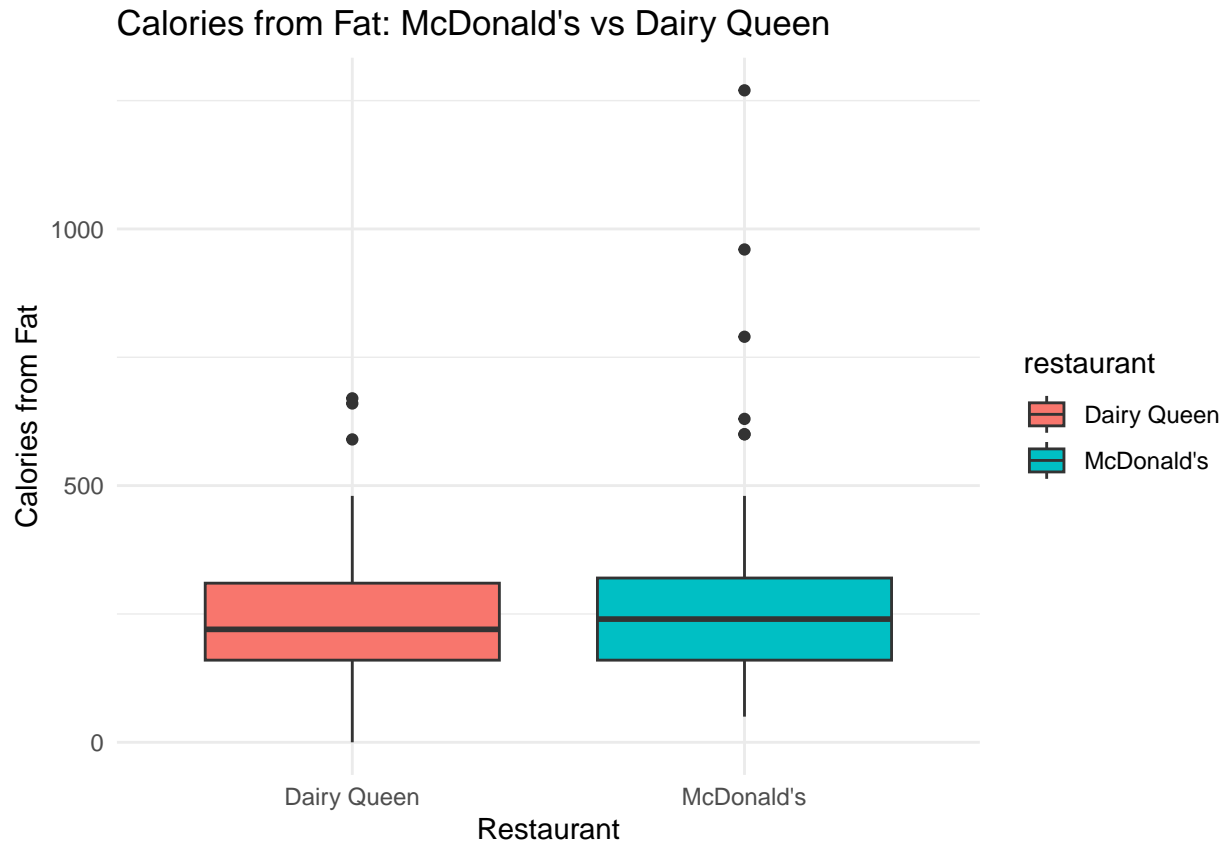
The normal distribution

In your description of the distributions, did you use words like *bell-shaped* or *normal*? It's tempting to say so when faced with a unimodal symmetric distribution.

Kevin Note: I didn't and I prefer to use more descriptive terms since the chart was normal-ish. The "ish" warrants a dive.

To see how accurate that description is, you can plot a normal distribution curve on top of a histogram to see how closely the data follow a normal distribution. This normal curve should have the same mean and standard deviation as the data. You'll be focusing on calories from fat from Dairy Queen products, so let's store them as a separate object and then calculate some statistics that will be referenced later.

```
dqmean <- mean(dairy_queen$cal_fat)
dqsd   <- sd(dairy_queen$cal_fat)
```

Next, you make a density histogram to use as the backdrop and use the `lines` function to overlay a normal probability curve. The difference between a frequency histogram and a density histogram is that while in a frequency histogram the *heights* of the bars add up to the total number of observations, in a density histogram the *areas* of the bars add up to 1. The area of each bar can be calculated as simply the height *times* the width of the bar. Using a density histogram allows us to properly overlay a normal distribution curve over the histogram since the curve is a normal probability density function that also has area under the curve of 1. Frequency and density histograms both display the same exact shape; they only differ in their y-axis. You can verify this by comparing the frequency histogram you constructed earlier and the density histogram created by the commands below.

```
ggplot(dairy_queen, aes(x = cal_fat)) +
  geom_blank() +
  geom_histogram(aes(y = ..density..), binwidth = 20) +
  stat_function(fun = dnorm, args = c(mean = dqmean, sd = dqsd), col = "skyblue")
```
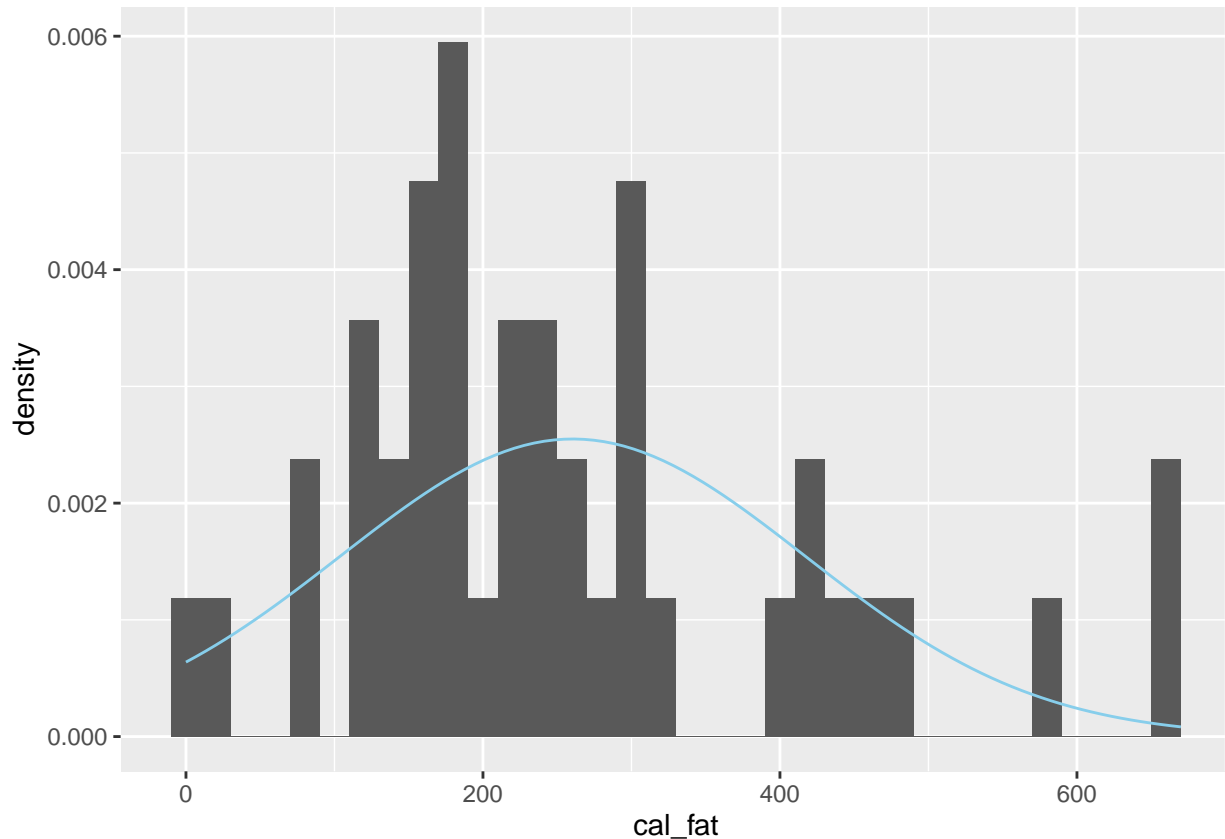

After initializing a blank plot with `geom_blank()`, the `ggplot2` package (within the `tidyverse`) allows us to add additional layers. The first layer is a density histogram. The second layer is a statistical function – the density of the normal curve, `dnorm`. We specify that we want the curve to have the same mean and standard deviation as the column of fat calories. The argument `col` simply sets the color for the line to be drawn. If we left it out, the line would be drawn in black.

2. Based on the this plot, does it appear that the data follow a nearly normal distribution?

Yes, the data appears normal. While the black bars plotting the density have gaps between them, this a function of dataset size. With enough data, it would start to fill in a normal curve shape. The line drawn on the graph is a normal curve but the peak would be higher for the data than the line plotted.

Evaluating the normal distribution

Eyeballing the shape of the histogram is one way to determine if the data appear to be nearly normally distributed, but it can be frustrating to decide just how close the histogram is to the curve. An alternative approach involves constructing a normal probability plot, also called a normal Q-Q plot for "quantile-quantile".

5

```
ggplot(data = dairy_queen, aes(sample = cal_fat)) +
  geom_line(stat = "qq") +
  labs(title = "Normal Quantile-Quantile Chart of Dairty Queen Fat Calories")
```

Normal Quantile–Quantile Chart of Dairty Queen Fat Calories
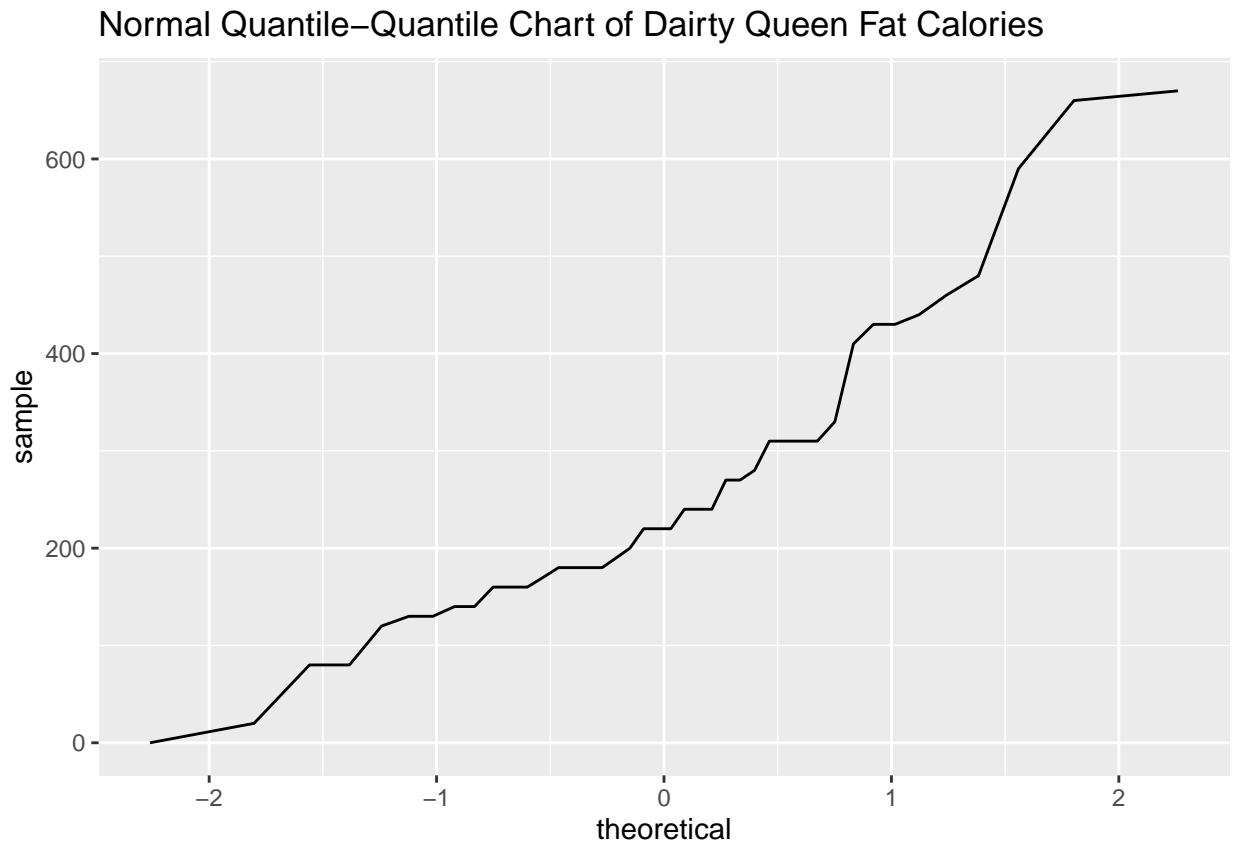
This time, you can use the `geom_line()` layer, while specifying that you will be creating a Q-Q plot with the `stat` argument. It's important to note that here, instead of using x instead `aes()`, you need to use `sample`.

The x-axis values correspond to the quantiles of a theoretically normal curve with mean 0 and standard deviation 1 (i.e., the standard normal distribution). The y-axis values correspond to the quantiles of the original unstandardized sample data. However, even if we were to standardize the sample data values, the Q-Q plot would look identical. A data set that is nearly normal will result in a probability plot where the points closely follow a diagonal line. Any deviations from normality leads to deviations of these points from that line.

The plot for Dairy Queen's calories from fat shows points that tend to follow the line but with some errant points towards the upper tail. You're left with the same problem that we encountered with the histogram above: how close is close enough?

A useful way to address this question is to rephrase it as: what do probability plots look like for data that I *know* came from a normal distribution? We can answer this by simulating data from a normal distribution using `rnorm`.

```
sim_norm <- rnorm(n = nrow(dairy_queen), mean = dqmean, sd = dqsd)
```

The first argument indicates how many numbers you'd like to generate, which we specify to be the same number of menu items in the `dairy_queen` data set using the `nrow()` function. The last two arguments determine the mean and standard deviation of the normal distribution from which the simulated sample will

be generated. You can take a look at the shape of our simulated data set, `sim_norm`, as well as its normal probability plot.

3. Make a normal probability plot of `sim_norm`. Do all of the points fall on the line? How does this plot compare to the probability plot for the real data? (Since `sim_norm` is not a data frame, it can be put directly into the `sample` argument and the `data` argument can be dropped.)
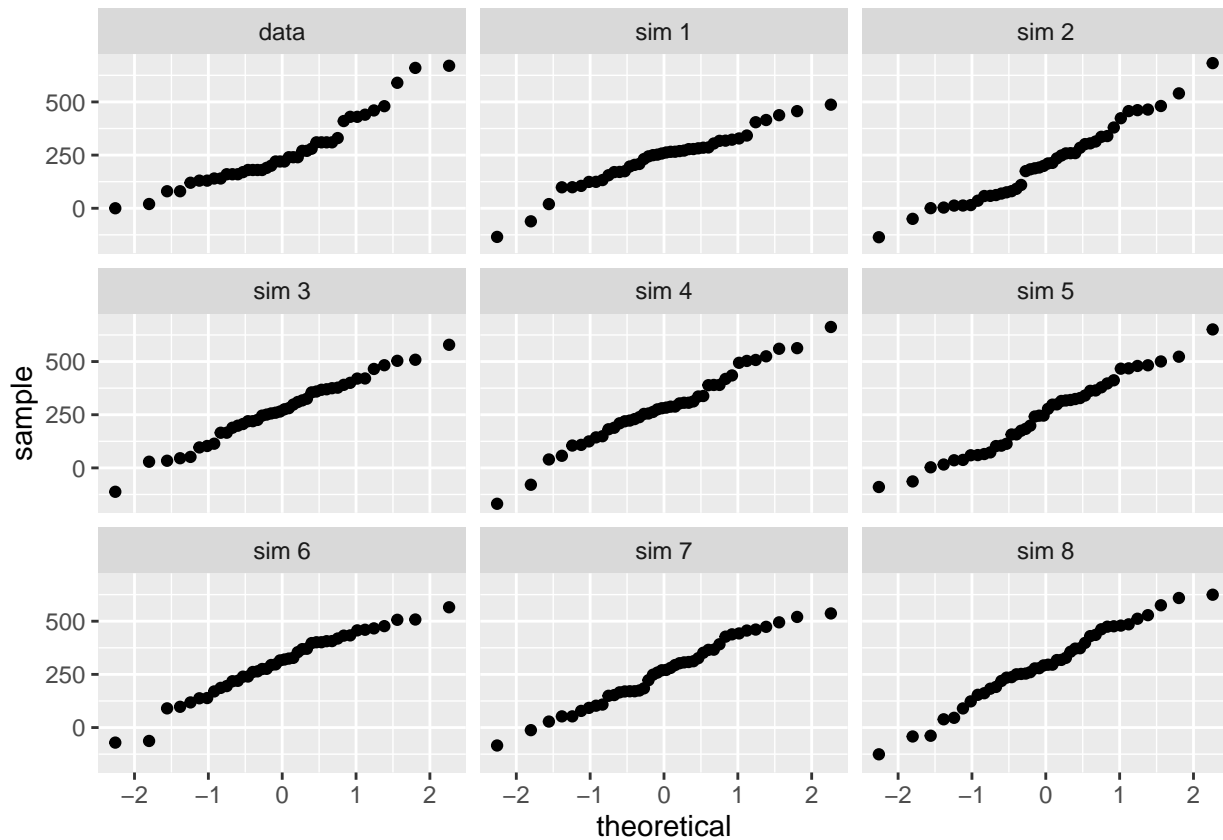
Answer:

Simulated Data: The Q-Q plot for simulated normal data shows points starting out away from the normal line at 2 quantiles below the mean of zero, with one quantile being about one standard deviation. The points then converge with normalize around the mean before deviating again starting at 1 quantile above. The actual dataset above doesn't follow a perfecr normal curve, with roughness in the approach and the tails. If anything, the plot points are accurately representing the data's natural state of not quite normal.

```
ggplot() +
  geom_qq(aes(sample = sim_norm)) +
  geom_qq_line(aes(sample = sim_norm)) +
  labs(title = "Normal QQ Plot of Simulated Data",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles")
```
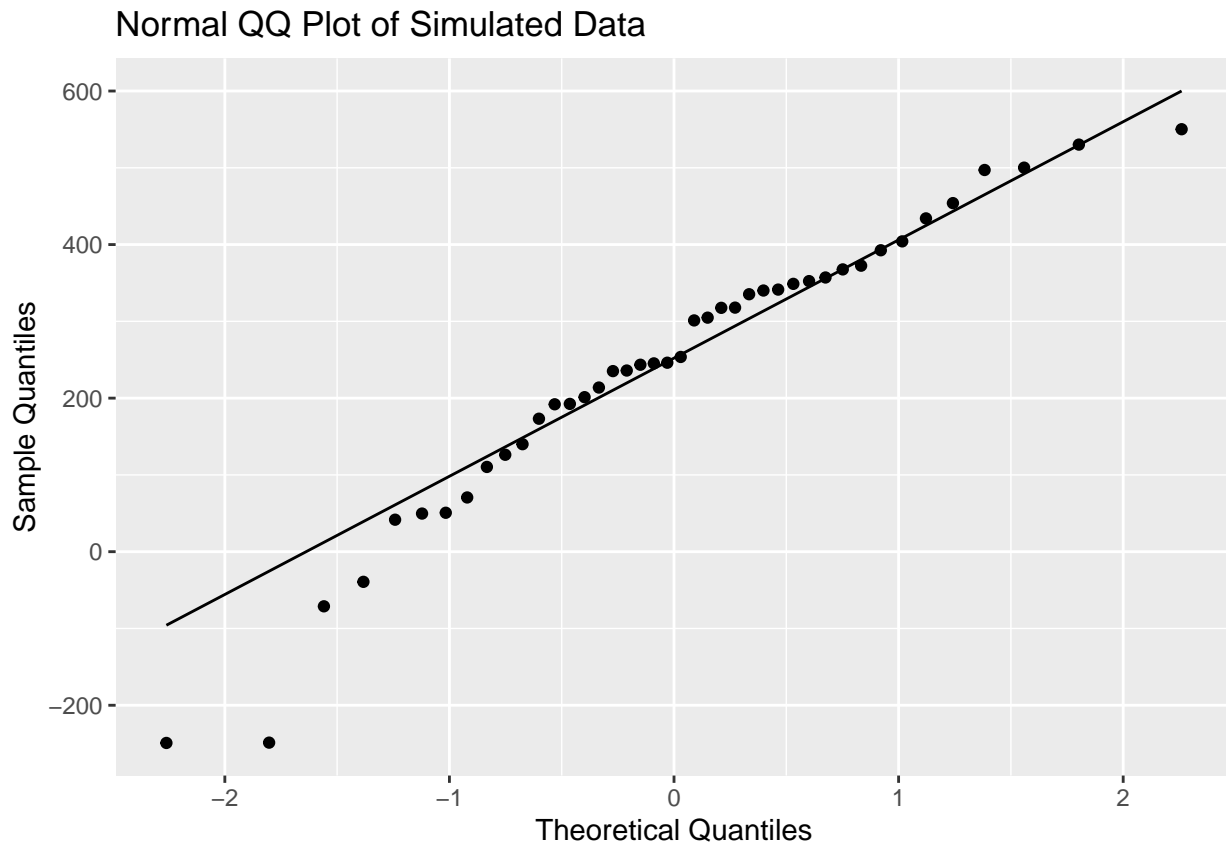

Even better than comparing the original plot to a single plot generated from a normal distribution is to compare it to many more plots using the following function. It shows the Q-Q plot corresponding to the original data in the top left corner, and the Q-Q plots of 8 different simulated normal data. It may be helpful to click the zoom button in the plot window.

```
qqnormsim(sample = cal_fat, data = dairy_queen)
```


4. Does the normal probability plot for the calories from fat look similar to the plots created for the simulated data? That is, do the plots provide evidence that the calories are nearly normal?

Answer: I love these hedge terms like "nearly." There's too much deviation in the normal QQ Plot (Data) chart starting around the first quantile above zero for me to use the term "nearly." I would characterize the regular data as having normal features at times and potentially returnable to normal with more data. Fast food will always have a long tail of higher calorie foods.

5. Using the same technique, determine whether or not the calories from McDonald's menu appear to come from a normal distribution.

Answer: The Mcondalds data is very normal until the first quantile above the 0=mean of the standard curve. The data then deviates pretty dramatically, which was also highlighted with the extended box plot values above. All of the simulations are basically normal, which is interesting. It seems to be handling the non-extreme values well but it can't quite nail the outliers or a skew.

"'{mcd-normality-bonanza}

mcdmean <- mean(mcdonalds$cal_fat)mcdsd < -sd(mcdonalds$cal_fat)

ggplot(mcdonalds, aes(x = cal_fat)) + geom_blank() + geom_histogram(aes(y = ..density..), binwidth = 20) + stat_function(fun = dnorm, args = c(mean = mcdmean, sd = mcdsd), col = "skyblue")

ggplot(data = mcdonalds, aes(sample = cal_fat)) + geom_line(stat = "qq") + labs(title = "Normal Quantile-Quantile Chart of McDonalds Fat Calories")

8

mcd_sim_norm <- rnorm(n = nrow(mcdonalds), mean = mcdmean, sd = mcdsd)

ggplot() + geom_qq(aes(sample = mcd_sim_norm)) + geom_qq_line(aes(sample = mcd_sim_norm)) +
labs(title = "Normal QQ Plot of Mcdonalds Simulated Data", x = "Theoretical Quantiles", y = "Sample
Quantiles")

```r
qqnormsim(sample = cal_fat, data = mcdonalds)
```
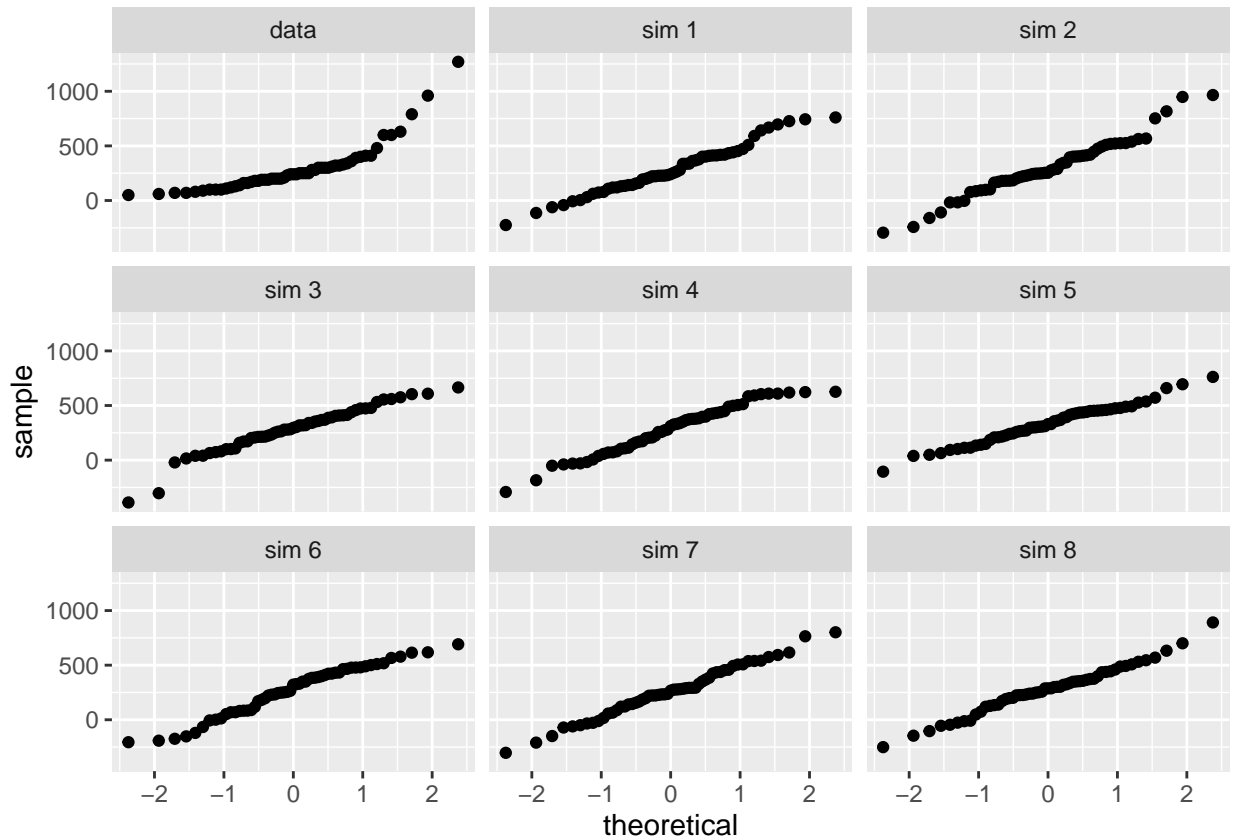

Normal probabilities

Okay, so now you have a slew of tools to judge whether or not a variable is normally distributed. Why should you care?

It turns out that statisticians know a lot about the normal distribution. Once you decide that a random variable is approximately normal, you can answer all sorts of questions about that variable related to probability. Take, for example, the question of, "What is the probability that a randomly chosen Dairy Queen product has more than 600 calories from fat?"

If we assume that the calories from fat from Dairy Queen's menu are normally distributed (a very close approximation is also okay), we can find this probability by calculating a Z score and consulting a Z table (also called a normal probability table). In R, this is done in one step with the function `pnorm()`.

```
1 - pnorm(q = 600, mean = dqmean, sd = dqsd)
```

```
## [1] 0.01501523
```

Note that the function `pnorm()` gives the area under the normal curve below a given value, `q`, with a given mean and standard deviation. Since we're interested in the probability that a Dairy Queen item has more than 600 calories from fat, we have to take one minus that probability.

Assuming a normal distribution has allowed us to calculate a theoretical probability. If we want to calculate the probability empirically, we simply need to determine how many observations fall above 600 then divide this number by the total sample size.

```
dairy_queen %>%
  filter(cal_fat > 600) %>%
  summarise(percent = n() / nrow(dairy_queen))
```

```
## # A tibble: 1 x 1
##   percent
##     <dbl>
## 1  0.0476
```

Although the probabilities are not exactly the same, they are reasonably close. The closer that your distribution is to being normal, the more accurate the theoretical probabilities will be.

6. Write out two probability questions that you would like to answer about any of the restaurants in this dataset. Calculate those probabilities using both the theoretical normal distribution as well as the empirical distribution (four probabilities in all). Which one had a closer agreement between the two methods?

Answer: 1. What is the probability that a Taco Bell meal has more than 400 calories?

This question ended up having the closer agreement, with only ~.02 separating theoretical from empirical.

Theoretical probability: 0.5936 Empircal: 0.557

```
tcbdata <- subset(fastfood, restaurant == "Taco Bell")

tcb_mean <- mean(tcbdata$calories)
tcb_sd <- sd(tcbdata$calories)

tcb_sim_norm <- rnorm(n = nrow(tcbdata), mean = tcb_mean, sd = tcb_sd)

ggplot() +
  geom_qq(aes(sample = tcb_sim_norm)) +
  geom_qq_line(aes(sample = tcb_sim_norm)) +
  labs(title = "Normal QQ Plot of Taco Bell Simulated Data",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles")
```

Normal QQ Plot of Taco Bell Simulated Data
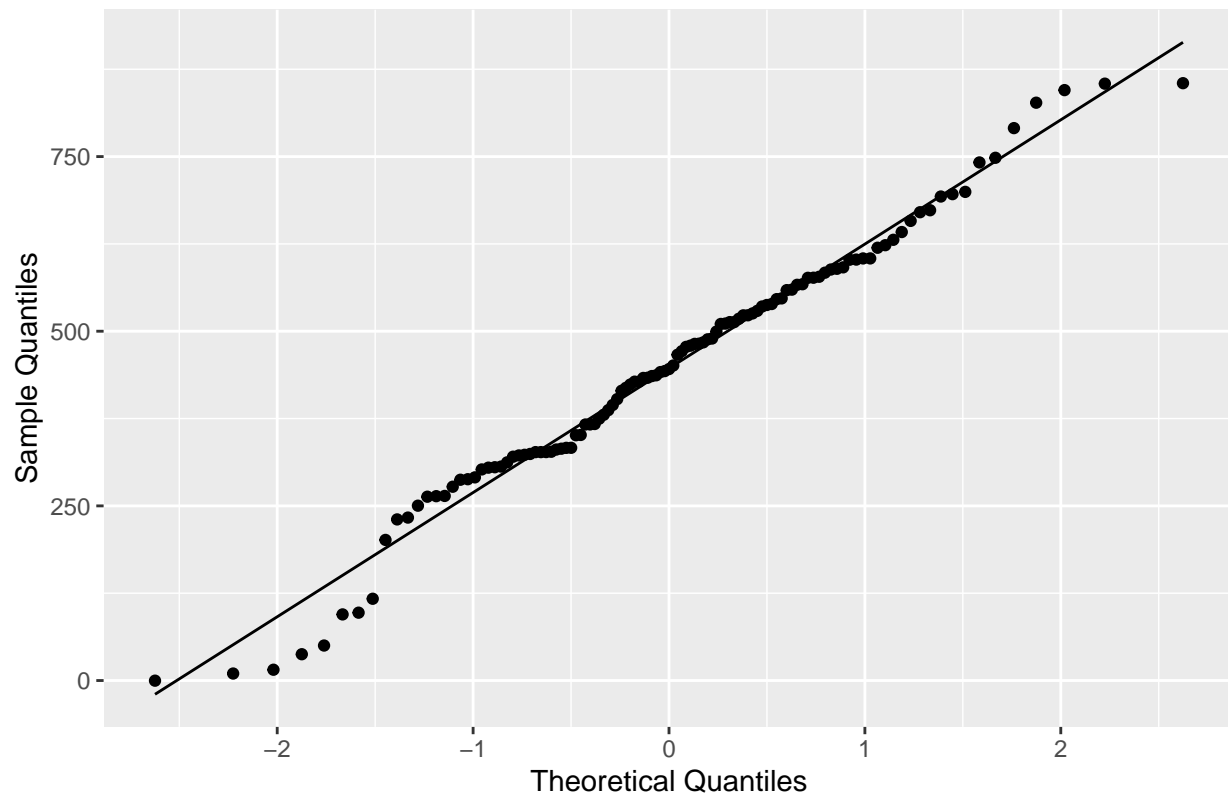

```r
#theoretical
1 - pnorm(q = 400, mean = tcb_mean, sd = tcb_sd)
```

```
## [1] 0.5935926
```

```r
#empircal
tcbdata %>%
  filter(calories > 400) %>%
  summarise(percent = n() / nrow(tcbdata))
```

```
## # A tibble: 1 x 1
##    percent
##      <dbl>
## 1    0.557
```

2. What is the probability that a Subway meal has between 400 and 600 calories?

Theoretical: 0.277 Emperical: 0.198

```r
eatfresh <- subset(fastfood, restaurant == "Subway")

sub_mean <- mean(eatfresh$calories)
sub_sd <- sd(eatfresh$calories)
```

```
sub_sim_norm <- rnorm(n = nrow(eatfresh), mean = sub_mean, sd = sub_sd)

ggplot() +
  geom_qq(aes(sample = sub_sim_norm)) +
  geom_qq_line(aes(sample = sub_sim_norm)) +
  labs(title = "Normal QQ Plot of Subway Simulated Data",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles")
```

Normal QQ Plot of Subway Simulated Data
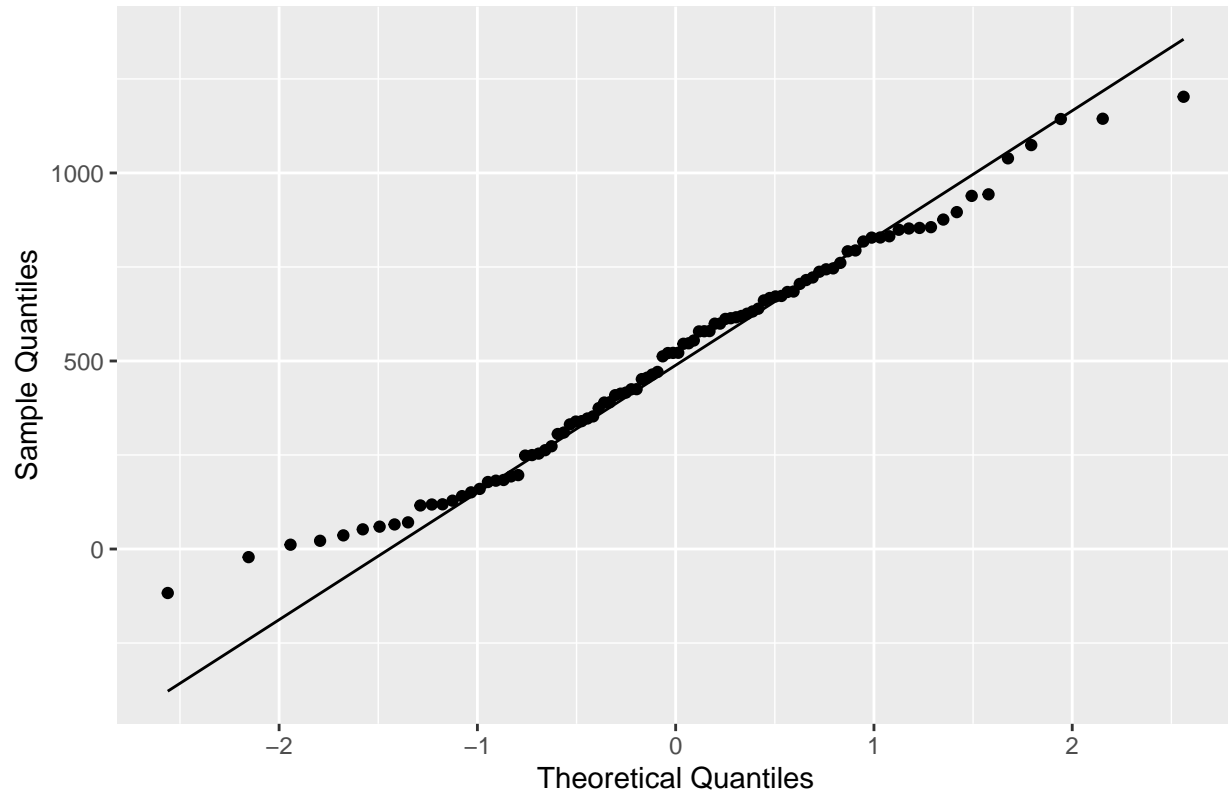

```
#theoretical probability
pnorm(600, mean = sub_mean, sd = sub_sd) - pnorm(400, mean = sub_mean, sd = sub_sd)
```

```
## [1] 0.2768949
```

```
#empirical probability
eatfresh %>%
  filter(calories > 400 & calories <= 600) %>%
  summarise(percent = n() / nrow(eatfresh))
```

```
## # A tibble: 1 x 1
##   percent
##     <dbl>
## 1   0.198
```

More Practice

7. Now let's consider some of the other variables in the dataset. Out of all the different restaurants, which ones' distribution is the closest to normal for sodium?

```r
sodium_cut <- fastfood %>%
  filter(!is.na(sodium)) %>%
  mutate(log_sodium = log(sodium))

sod_mean <- mean(sodium_cut$sodium)
sod_sd <- sd(sodium_cut$sodium)

ggplot(sodium_cut, aes(x = log_sodium, color = restaurant, fill = restaurant)) +
  geom_density(alpha = 0.3) +
  labs(title = "Sodium Distribution Across Restaurantes - Log Normalized",
       x = "Sodium",
       y = "Density") +
  theme_minimal()
```
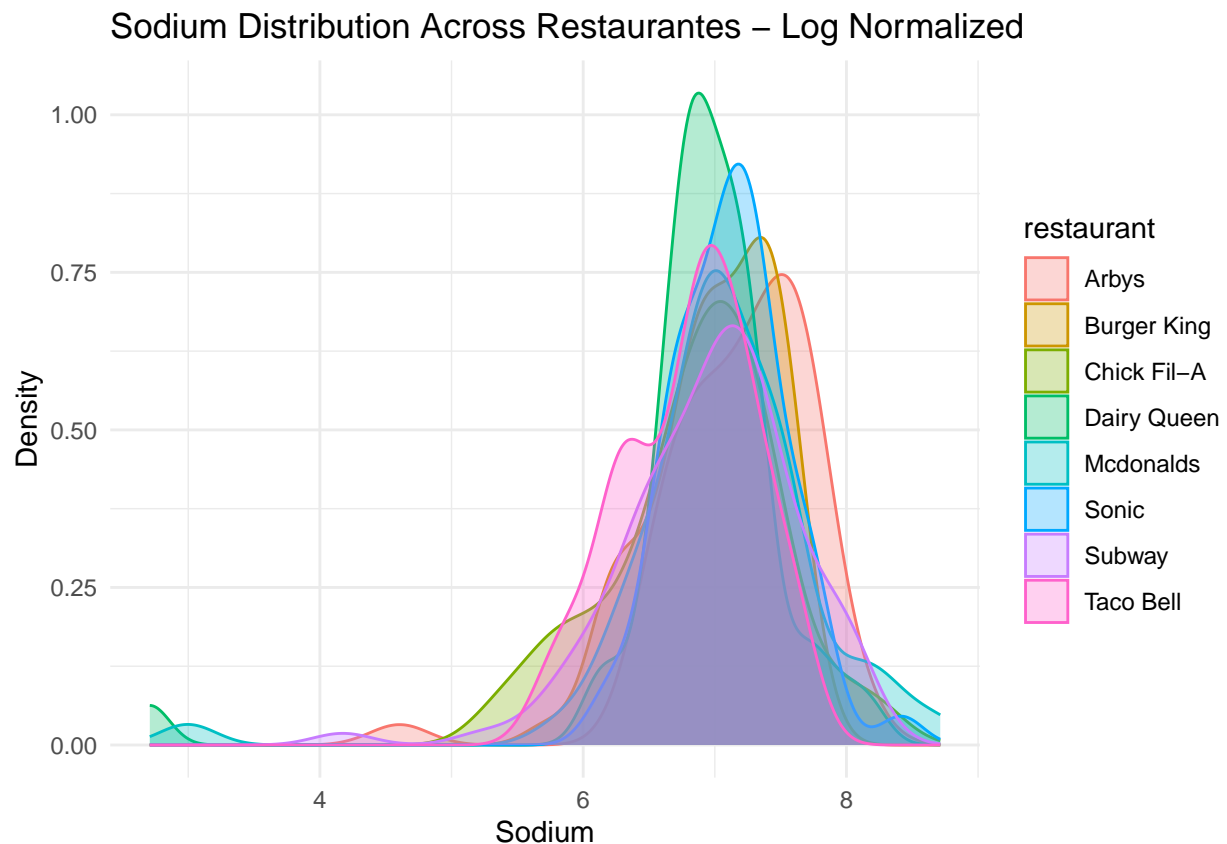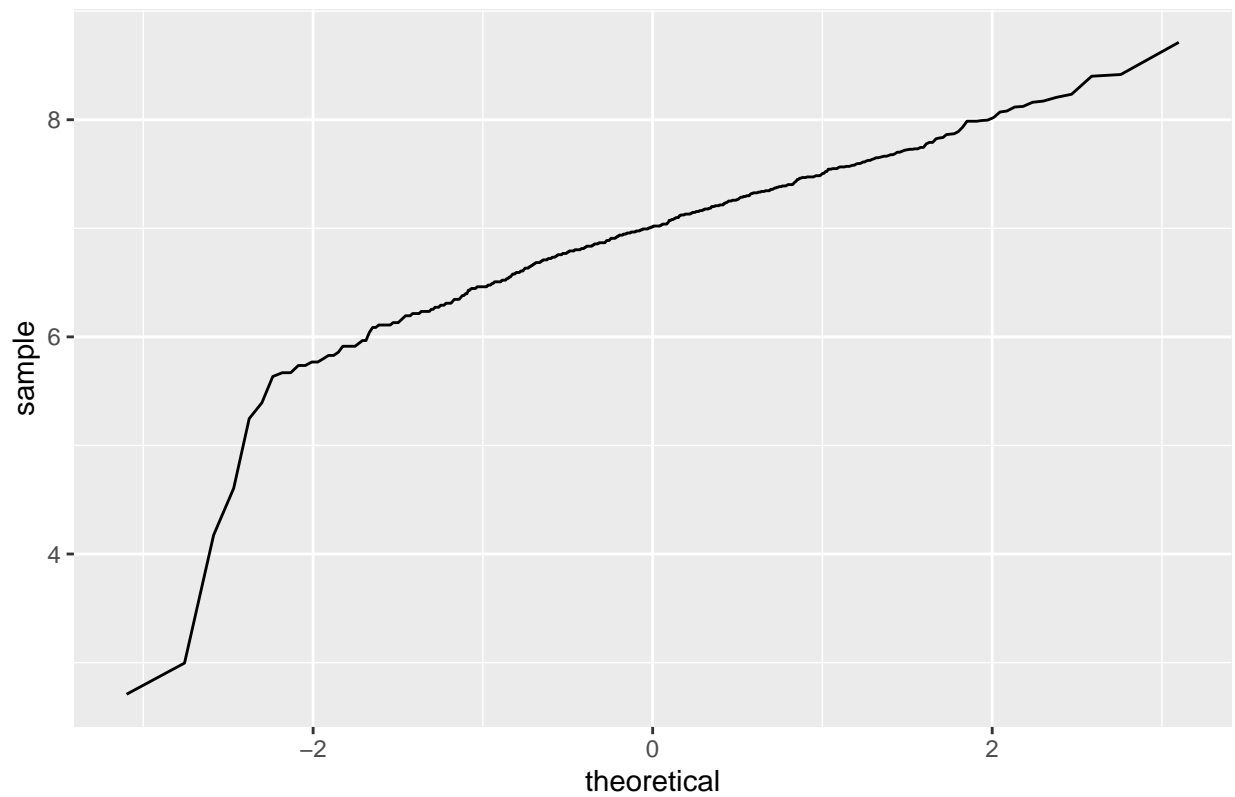


```r
ggplot(sodium_cut, aes(x = log_sodium)) +
  geom_blank() +
  geom_histogram(aes(y = ..density..), binwidth = 20) +
  stat_function(fun = dnorm, args = c(mean = sod_mean, sd = sod_sd), col = "skyblue")
```
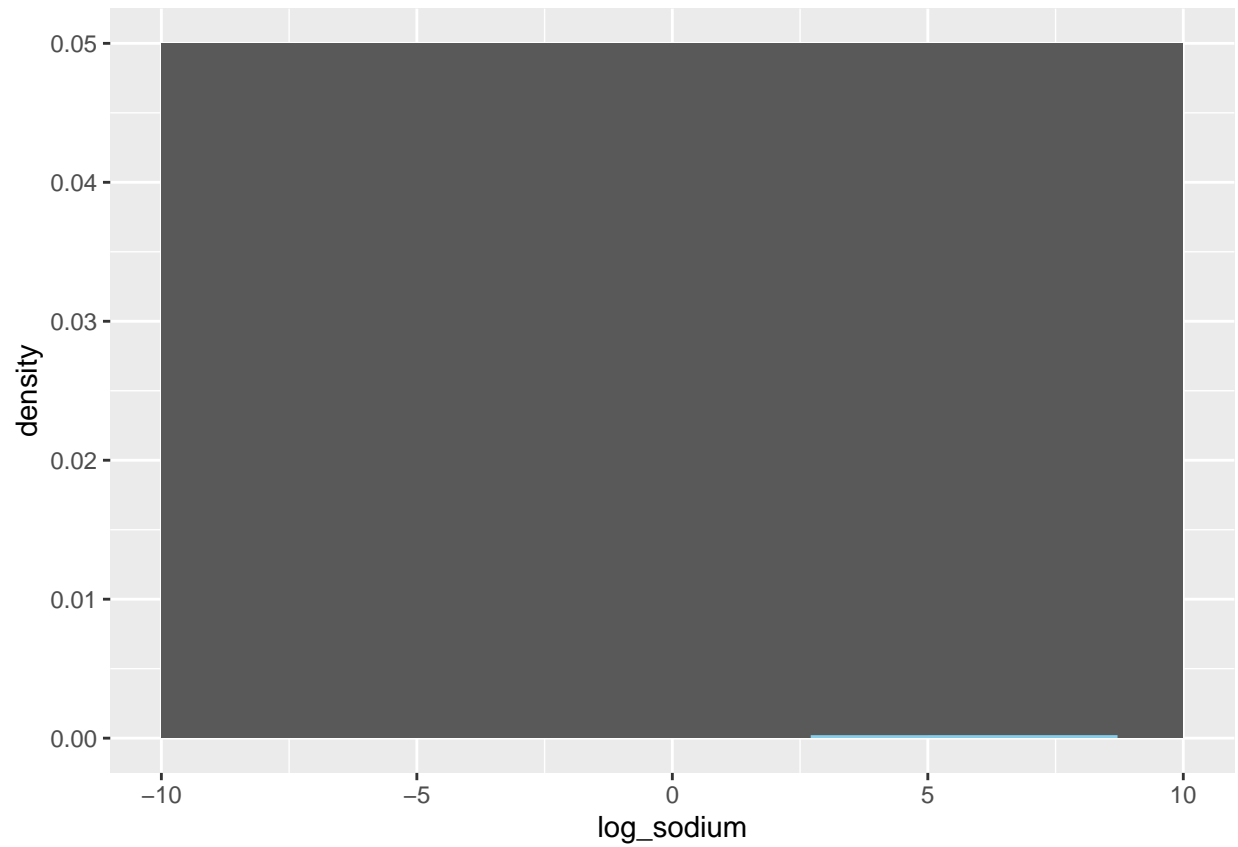
```
ggplot(data = sodium_cut, aes(sample = log_sodium)) +
  geom_line(stat = "qq") +
  labs(title = "Normal Quantile-Quantile Chart of Sodium Levels")
```

Normal Quantile–Quantile Chart of Sodium Levels


```r
sod_sim_norm <- rnorm(n = nrow(sodium_cut), mean = sod_mean, sd = sod_sd)

ggplot() +
  geom_qq(aes(sample = sod_sim_norm)) +
  geom_qq_line(aes(sample = sod_sim_norm)) +
  labs(title = "Normal QQ - Sodiun",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles")
```

Normal QQ – Sodiun
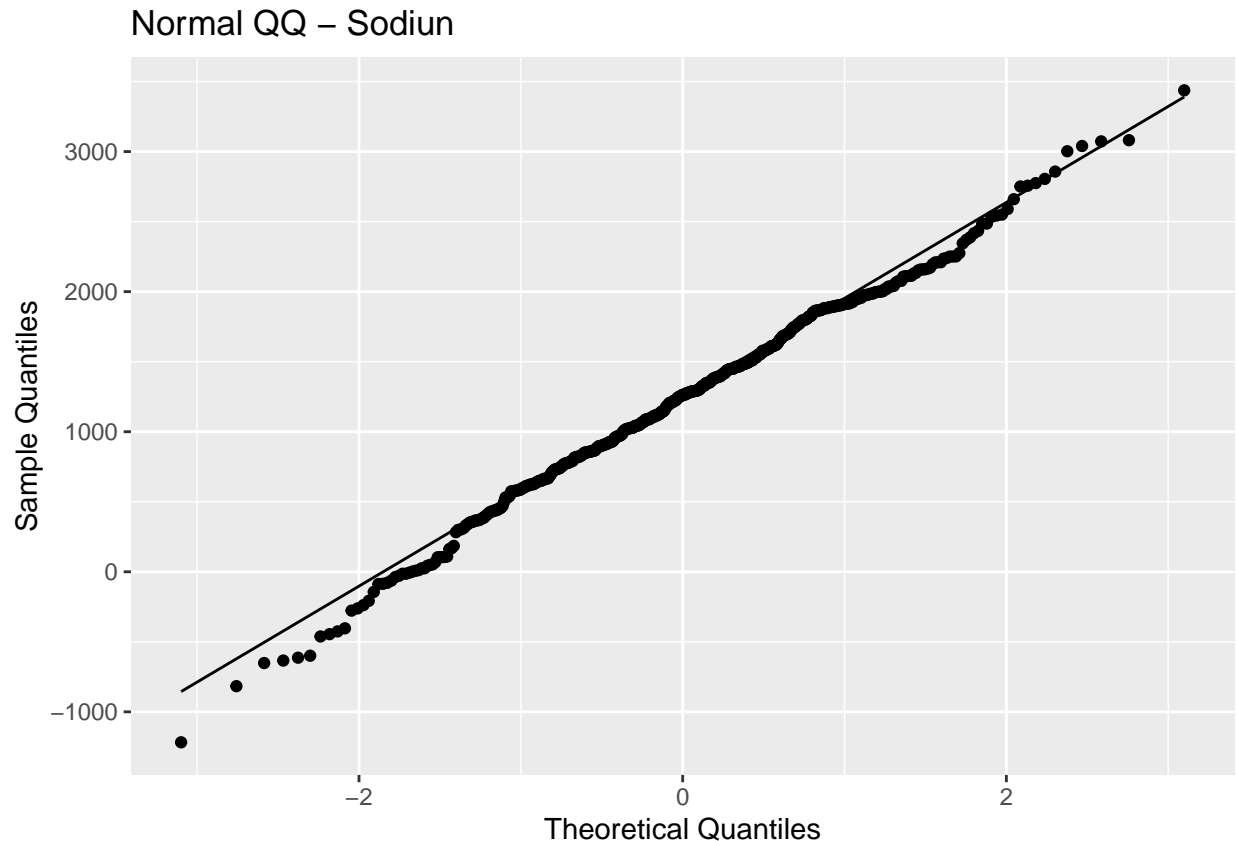
Based on the above graphs and the below skewness and kurtosis charts, Arby's and Burger King have the normal normal distributions. Burger King's skewness of 0.188 indicates a nearly symmetric distribution. Arby's low kurtosis and skewness means it's not as weak overall, whereas Mcdonald's skewness of 2.27 is continued evidence fo what we saw earlier, where the data is not really normal.

"'{skew-kurtote}

sod_stats <- fastfood %>% group_by(restaurant) %>% summarise(mean_sodium = mean(sodium, na.rm = TRUE), sd_sodium = sd(sodium, na.rm = TRUE), sod_skewness = skewness(sodium, na.rm = TRUE), sod_kurtosis = kurtosis(sodium, na.rm = TRUE) # Corrected this line)

ggplot(sod_stats, aes(x = restaurant, y = sod_skewness, fill = restaurant)) + geom_bar(stat = "identity") + labs(title = "Skewness by Restaurant - Sodium", x = "Restaurant", y = "Skewness") + theme_minimal()

ggplot(sod_stats, aes(x = restaurant, y = sod_kurtosis, fill = restaurant)) + geom_bar(stat = "identity") + labs(title = "Kurtosis by Restaurant - Sodium", x = "Restaurant", y = "Kurtosis") + theme_minimal()

8. Note that some of the normal probability plots for sodium distributions seem to have a stepwise patt

Answer:

I believe this is due to the data being considered discrete rather than continuous. These health stats i

It would also be impacted by multiple lines sharing the same sodium values, like 1890 or 1900. These tie

9. As you can see, normal probability plots can be used both to assess normality and visualize skewness

Answer:

Based on this QQ plot, I suspect this data is rightward skewed. The data clusters tightly around the lo

```r
sonic <- fastfood %>%
  filter(restaurant == "Sonic")

ggplot(sonic, aes(sample = total_carb)) +
  geom_qq() +
  geom_qq_line() +
  labs(title = "Normal QQ Plot of Total Carbs - Sonic",
       x = "Theoretical Quantiles",
       y = "Sample Quantiles") +
  theme_minimal()
```


Normal QQ Plot of Total Carbs – Sonic
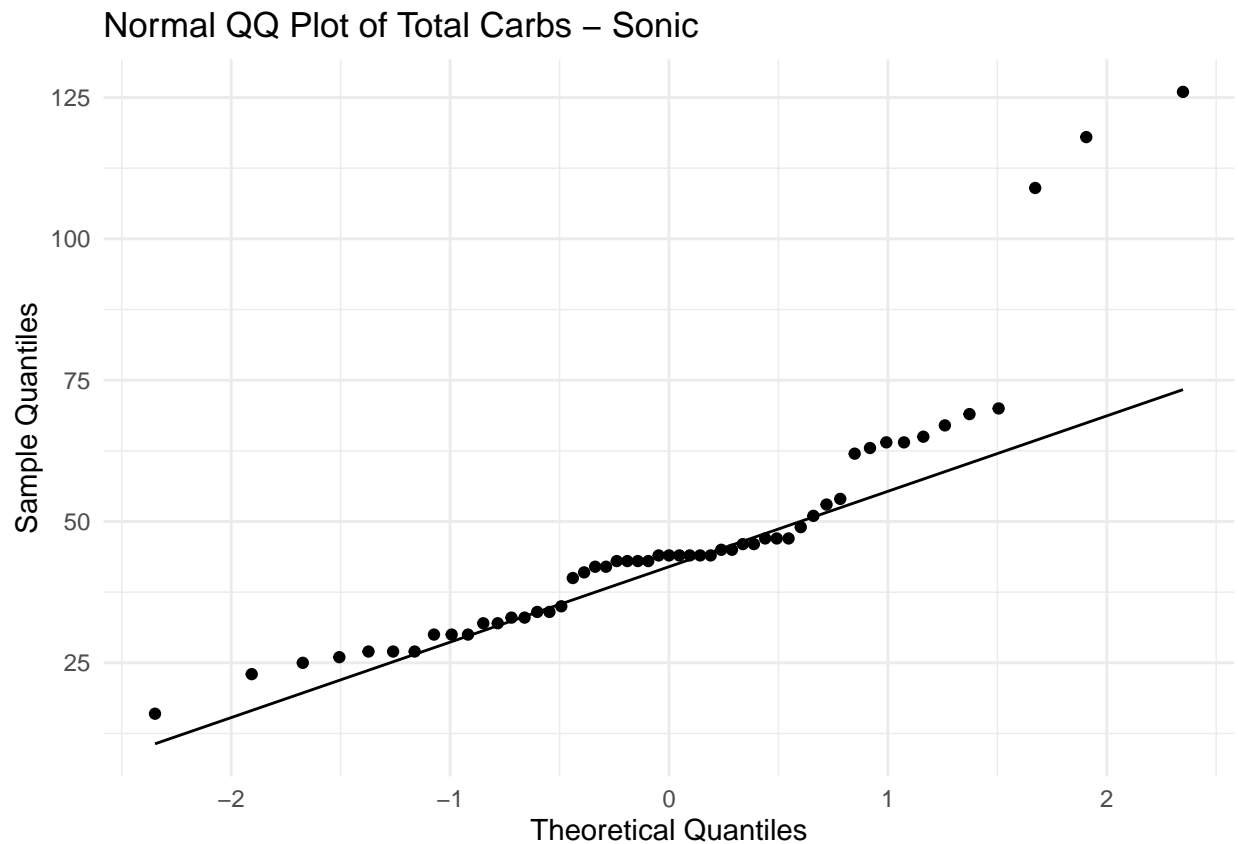
Here's the histogram to prove what I suspected based on the QQ plot. It's clustered tightly in a seemingly normal form and then there's a long tail in a rightward skew format.

```r
ggplot(sonic, aes(x = total_carb)) +
  geom_histogram(binwidth = 10, fill = "skyblue", color = "black") +
  labs(title = "Histogram of Total Carbs - Sonic",
       x = "Grams",
```

```
    y = "Frequency") +
theme_minimal()
```

Histogram of Total Carbs – Sonic