The normal distribution

William Jasmine

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)
library(dplyr)
```

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

```
## # A tibble: 6 x 17
##   restaur~1 item  calor~2 cal_fat total~3 sat_fat trans~4 chole~5 sodium total~6
##   <chr>     <chr>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>
## 1 Mcdonalds Arti~     380      60       7       2     0        95   1110      44
## 2 Mcdonalds Sing~     840     410      45      17     1.5     130   1580      62
## 3 Mcdonalds Doub~    1130     600      67      27     3       220   1920      63
## 4 Mcdonalds Gril~     750     280      31      10     0.5     155   1940      62
## 5 Mcdonalds Cris~     920     410      45      12     0.5     120   1980      81
## 6 Mcdonalds Big ~     540     250      28      10     1        80    950      46
## # ... with 7 more variables: fiber <dbl>, sugar <dbl>, protein <dbl>,
## #   vit_a <dbl>, vit_c <dbl>, calcium <dbl>, salad <chr>, and abbreviated
## #   variable names 1: restaurant, 2: calories, 3: total_fat, 4: trans_fat,
## #   5: cholesterol, 6: total_carb
```

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.

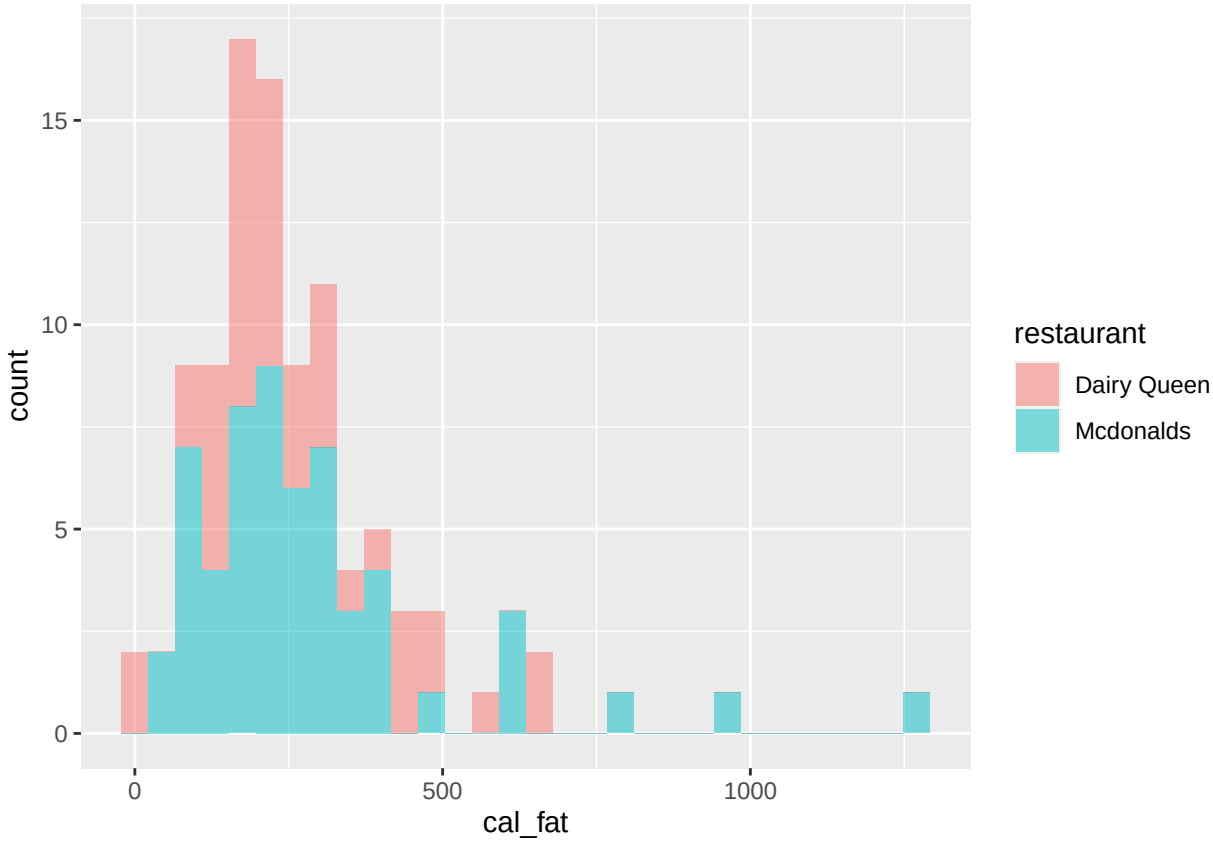
```
mcdonalds <- fastfood %>%
  filter(restaurant == "Mcdonalds")
dairy_queen <- fastfood %>%
  filter(restaurant == "Dairy Queen")
```

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?

WJ Response:

The chunk below provides the code to compare the histograms of the calories from fat values for each of the restaurant's (Dairy Queen (DQ) and McDonalds) menu items:

```
plt_data <- rbind(select(mcdonalds, restaurant, cal_fat),
                  select(dairy_queen, restaurant, cal_fat))
ggplot(data = plt_data, aes(x=cal_fat, fill=restaurant)) +
    geom_histogram(alpha=0.5)
```



The histograms have similar centers ands shapes, but the spreads differ slightly: while Dairy Queens's distribution has a greater amplitude and seems more tightly bound to its center, the McDonalds distribution seems slightly more compact and has a number of extreme values to the right side of the plot.

The normal distribution

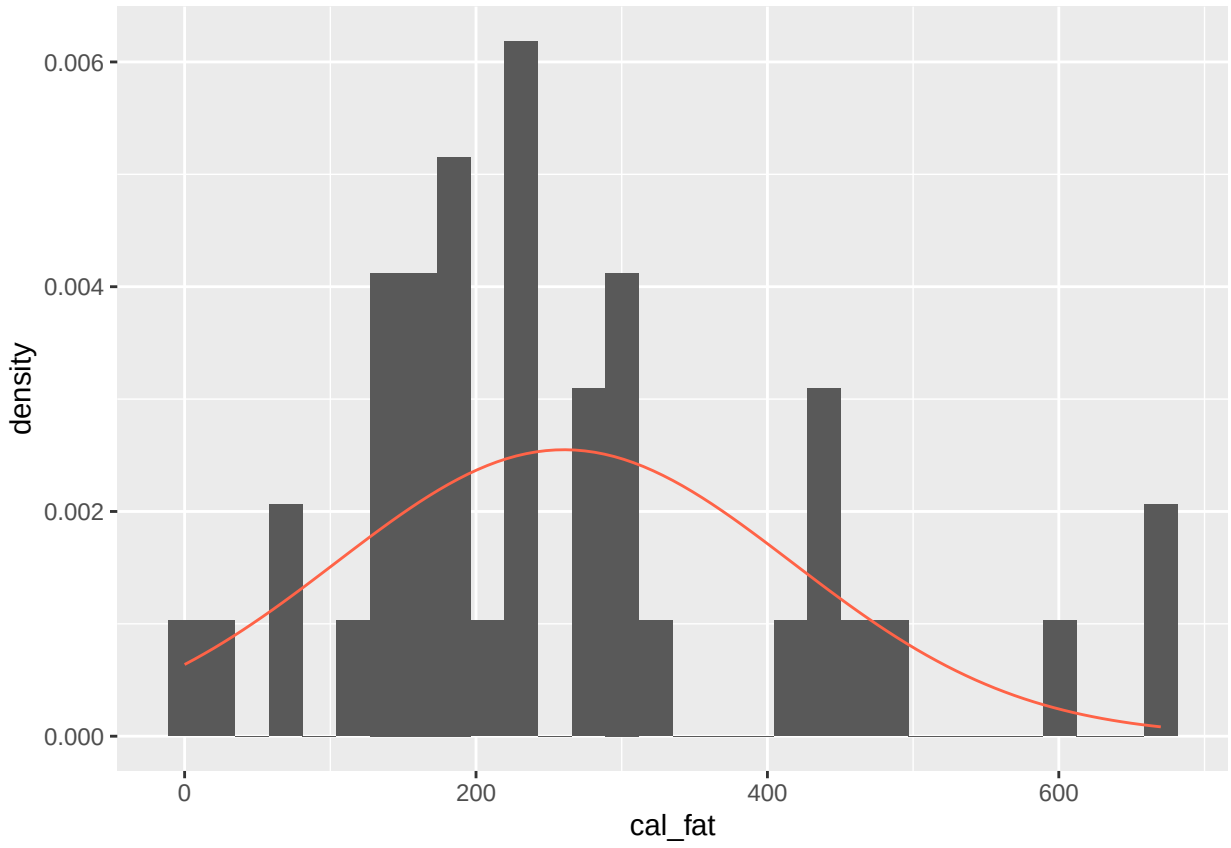
In your description of the distributions, did you use words like *bell-shaped*or *normal*? It's tempting to say so when faced with a unimodal symmetric distribution.

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



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?

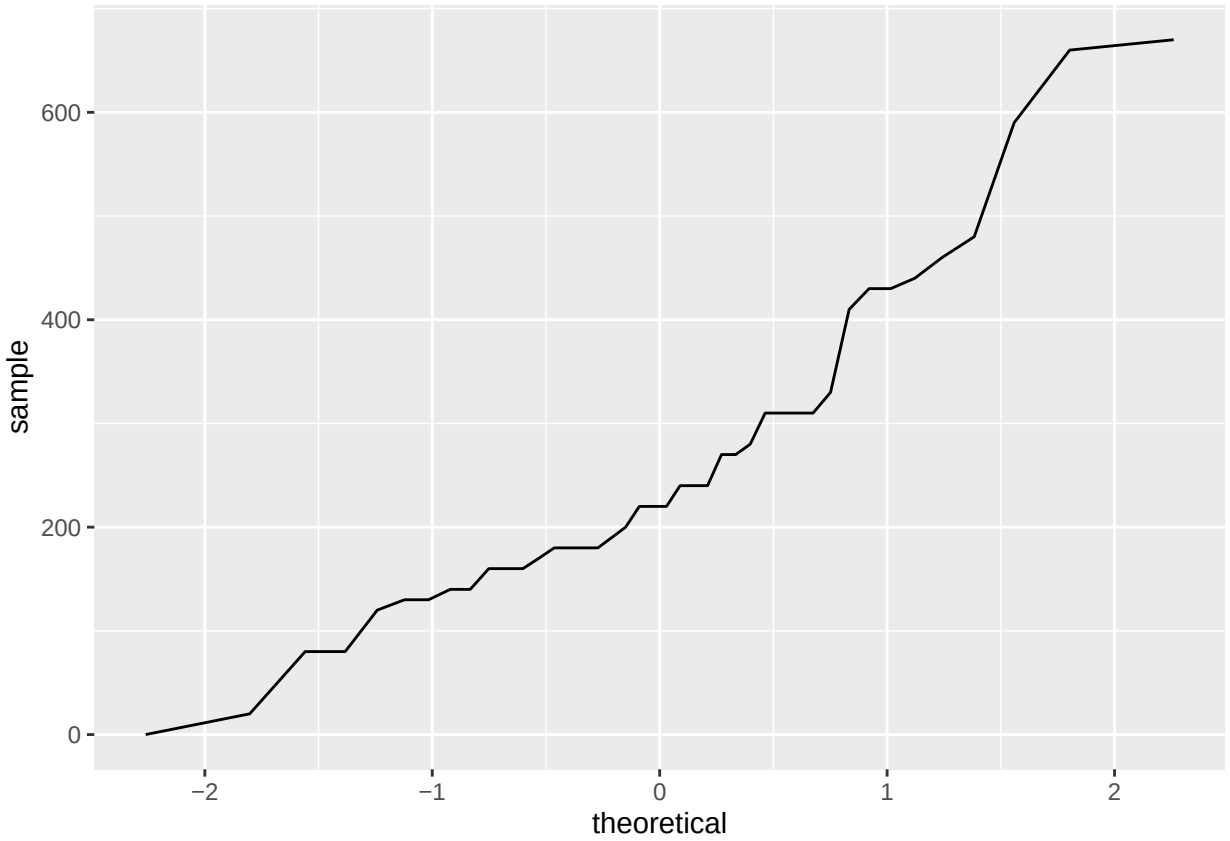
WJ Response:

While a lot of the bars in the histogram exceed the height of the theoretical normal distribution graph, the shape of the distribution does seem to align quite well. However, visual analysis alone does not seem to provide enough confidence that the distribution is or is not normal, and further testing would need to be done to provide evidence for either.

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".

```
ggplot(data = dairy_queen, aes(sample = cal_fat)) +
  geom_line(stat = "qq")
```



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)
```

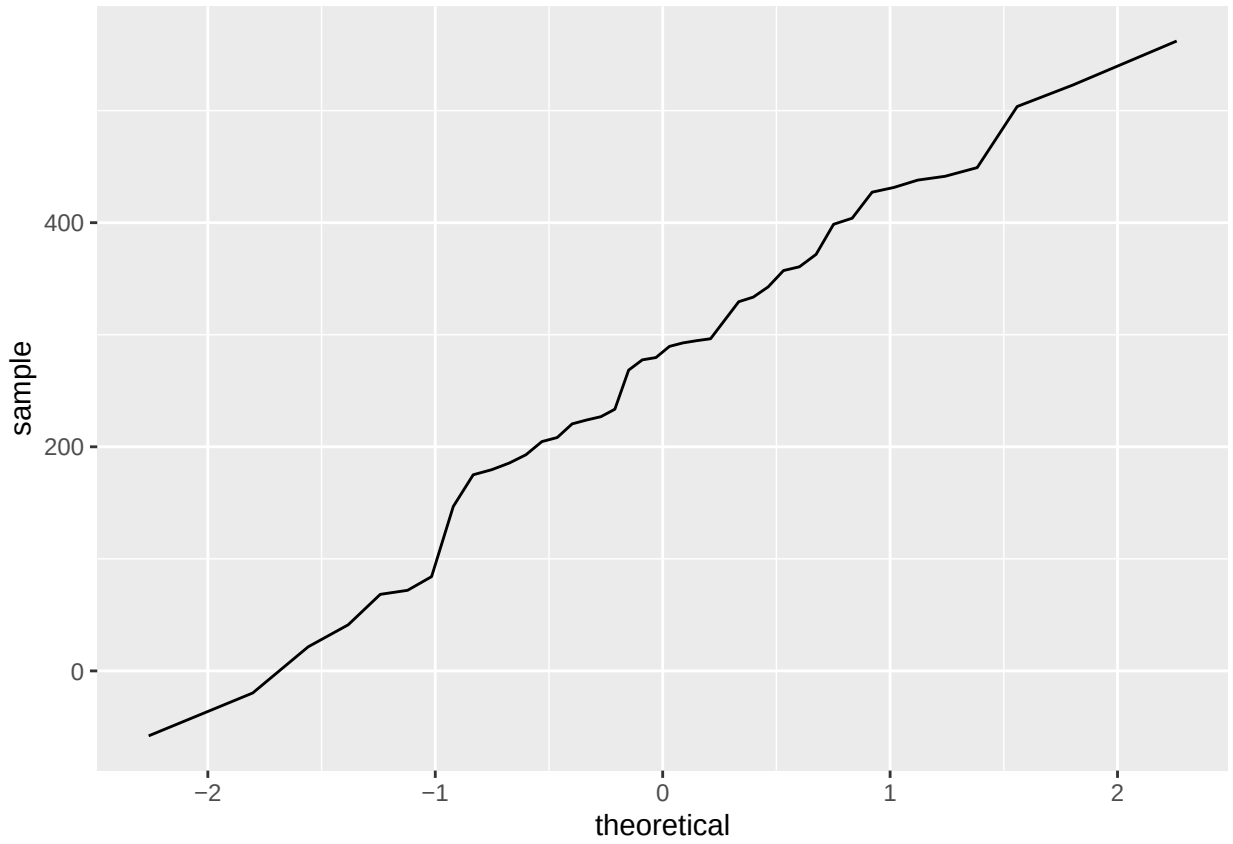
The first argument indicates how many numbers you'd like to generate, which we specify to be the same number of menu items in the `dairy_queen` data set using the `nrow()` function. The last two arguments determine the mean and standard deviation of the normal distribution from which the simulated sample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

WJ Response:

The chunk below generates a Q-Q plot of our simulated `cal_fat` data:

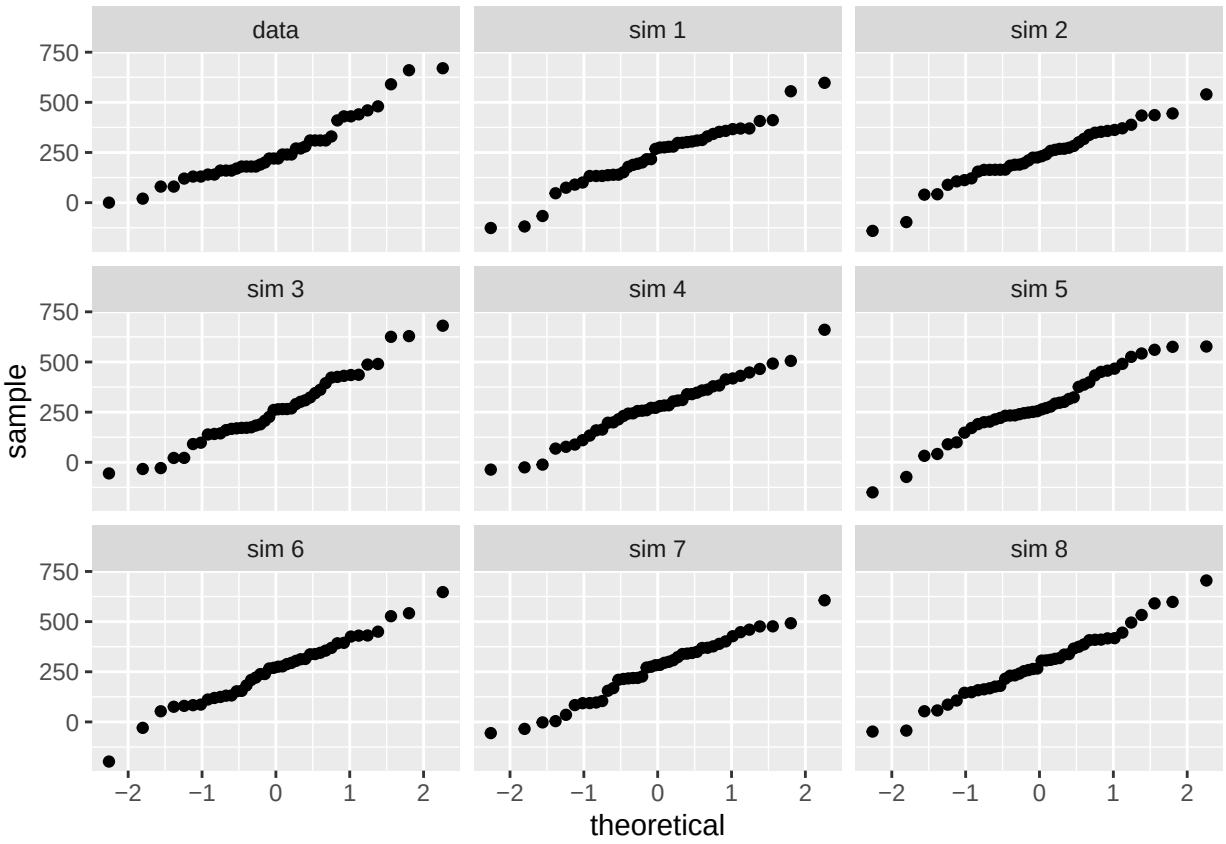
```
ggplot(, aes(sample=sim_norm)) +
  geom_line(stat = "qq")
```



The plot above does look closer to just a regular diagonal line compared to the real data (as expected), and comparing its shape to the Q-Q plot for the real data does inspire more confidence that we may be able to categorize it as normal.

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?

WJ Response:

Comparing the top left Q-Q plot to the other eight in the graphic above, it becomes more clear that the real and simulated data seem to resemble one another. They are all nearly diagonal, and all seem to follow the same basic structure: the majority of points are in the middle of the plot but include a smaller subset placed at the right and left tails, which are below and above this main section, respectively. Given that we are comparing the Q-Q plot for the real data now with 8 simulated Q-Q plots (as opposed to just one), this provides greater evidence to conclude that the real data is indeed normal.

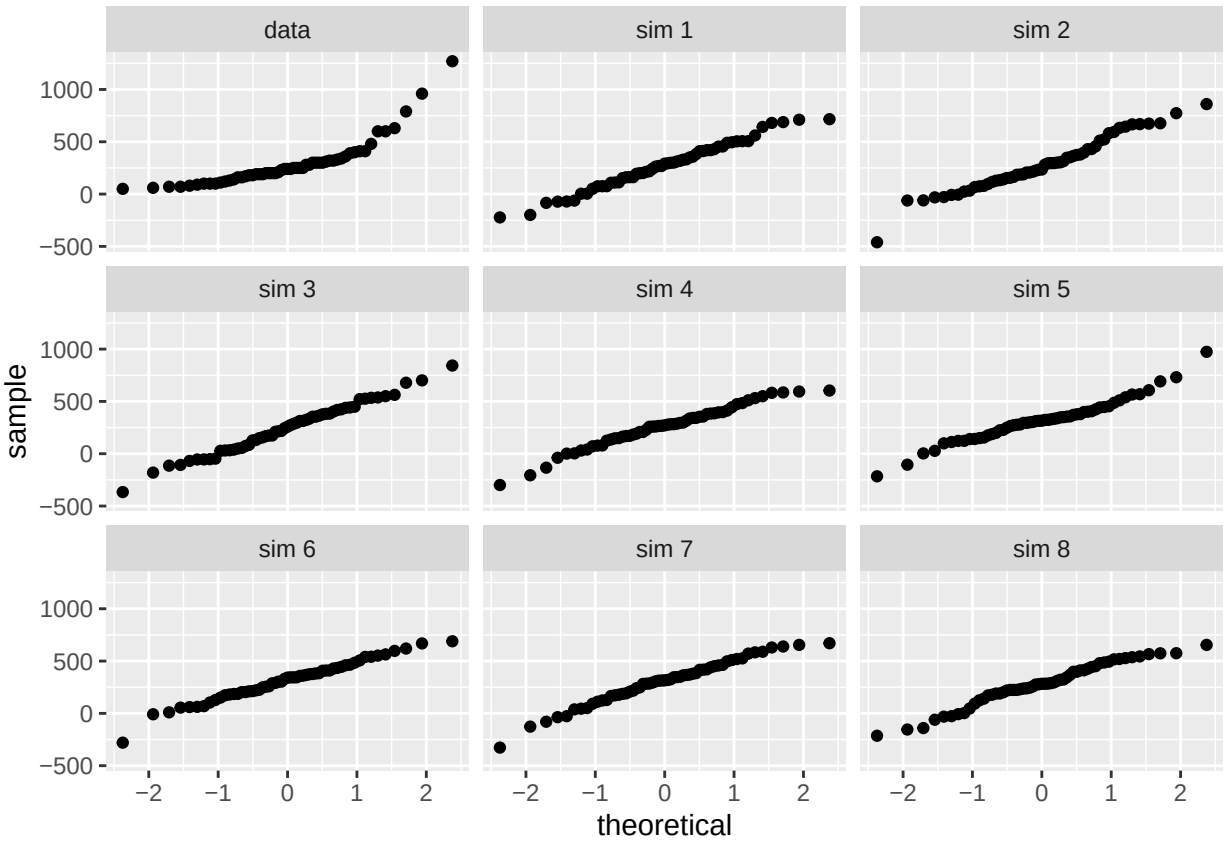
---

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

WJ Response:

The code chunk below replicates the output shown above with the `qqnormsim` function, but using the `mcdonalds` dataframe:

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



In this case, it appears as though the Q-Q plot produced from the real data does not match the 8 Q-Q plots generated using simulated, normally distributed data. While the simulated Q-Q plots all closely resemble diagonal lines (as is to be expected), the Q-Q plot of the real data has a different shape, exhibiting an almost exponential or parabolic shape. As such, this provides strong evidence for the fact that the calories from fat in the McDonalds menu items do not appear to come from a normal distribution.

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?

WJ Response:

Question 1: What is the probability that a single menu item chosen at random from Dairy Queen's menu has less than 200 calories from fat, or greater than 400 calories from fat?

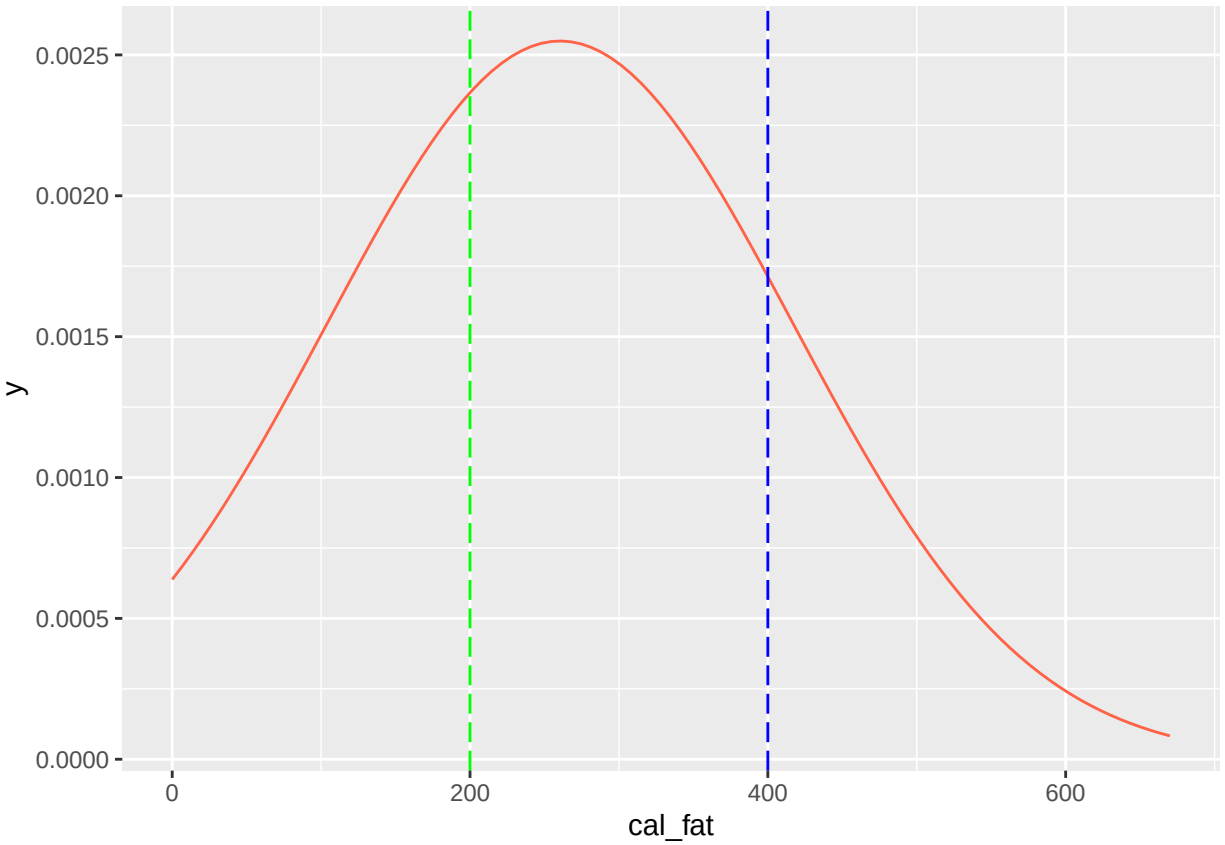
First, the theoretical probability using the `pnorm` function:

```
(1 - (pnorm(q = 400, mean = dqmean, sd = dqsd)) +
pnorm(q = 200, mean = dqmean, sd = dqsd)) * 100
```

```
## [1] 53.58764
```

The results above reveal that the normal distribution tells us there is about a 54% chance that a single DQ menu item has more than 400 calories or less than 200 calories. Answering this question can be broken up into two separate chunks (shown in the two lines of code above) to calculate the two separate probabilities (above 400, below 200) and adding them together (since the question includes "or"). This is explained graphically below: the value above is the result of adding the area under the normal distribution to the left of the green line and the area under the normal distribution to the right of the blue line.

```
ggplot(data = dairy_queen, aes(x = cal_fat)) +
        geom_blank() +
        stat_function(fun = dnorm, args = c(mean = dqmean, sd = dqsd), col = "tomato") +
    geom_vline(xintercept = 200, linetype="longdash", colour="green") +
    geom_vline(xintercept = 400, linetype="longdash", colour="blue")
```



The chunk below checks the actual data to confirm if the above prediction using the normal distribution is accurate:

```
(dairy_queen %>%
  filter(cal_fat < 200 | cal_fat > 400) %>%
    summarise(percent = n() / nrow(dairy_queen))) * 100
```

```
##    percent
## 1 64.28571
```

Using the real values from Dairy Queen, we see that ~64% of their menu items actually have a calories from fat value within the desired range. This is a little off compared to our prediction, but not outside the realm of possibility.

–

Question 2: Dietary guidelines stipulate that the calories fat should account for 25%-35% of the calories consumed every day. What is the probability a single Dairy Queen menu item chosen at random meets this guideline for an individual whom is consuming 2,000 calories a day?

First, the theoretical probability using the `pnorm` function:

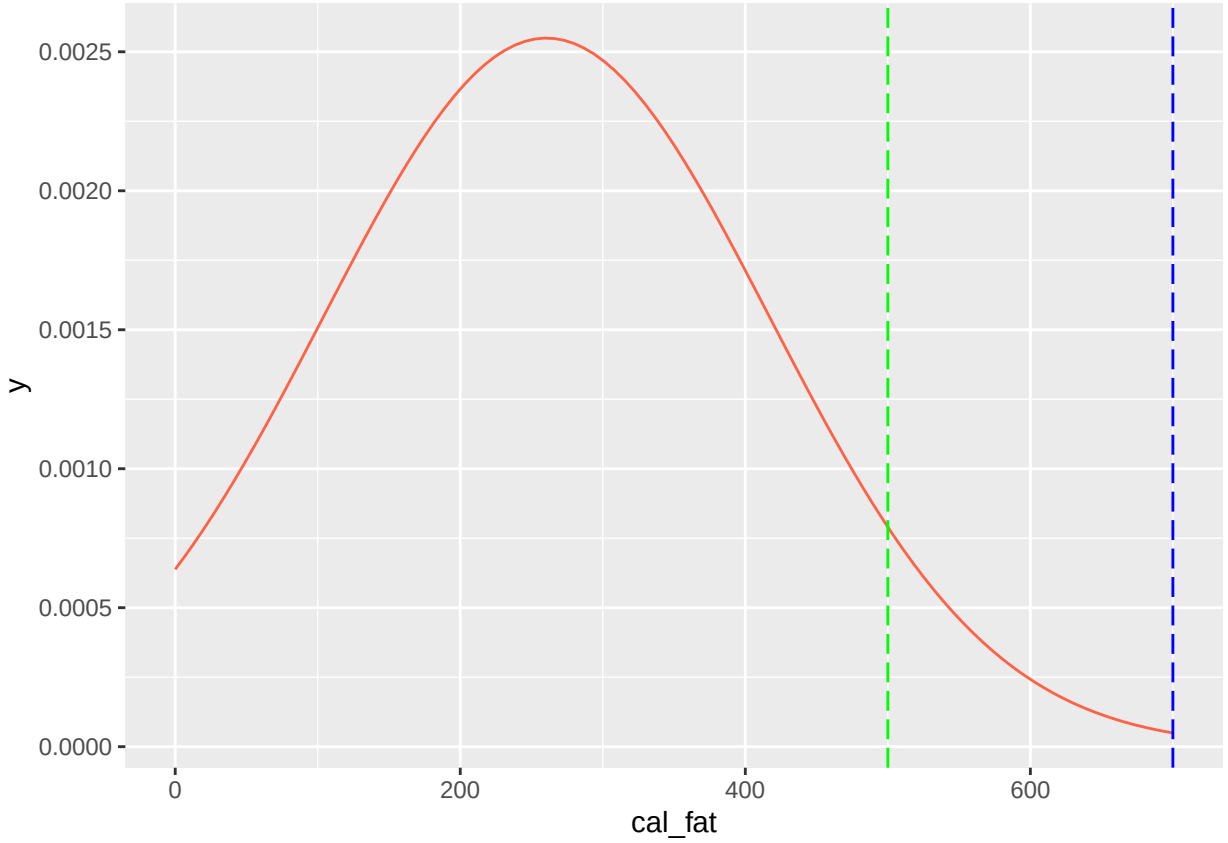
```
cal_max = 2000 * 0.35
cal_min = 2000 * 0.25

100 * (pnorm(q = cal_max, mean = dqmean, sd = dqsd) -
pnorm(q = cal_min, mean = dqmean, sd = dqsd))
```

```
## [1] 6.04411
```

The results above reveal that the normal distribution tells us there is about a 6% chance that a single DQ

menu item has the number of recommended calories from fat to consume in a single day. The subtraction shown in the chunk above is performed due to the fact that the `pnorm` function gets the area of the normal distribution to the left of the specified value. Thus, the area of our desired range can be found by simply subtracting the smaller area (generated from using the value at the lower end of the range) from the larger area (generated from using the value at the higher end of the range). This is explained graphically below: the value above is the result of subtracting the area under the normal distribution to the left of the green line from the area under the normal distribution to the left of the blue line (giving the area under the curve between the two lines).

```
ggplot(data = dairy_queen, aes(x = cal_fat)) +
        geom_blank() +
        stat_function(fun = dnorm, args = c(mean = dqmean, sd = dqsd), col = "tomato") +
    geom_vline(xintercept = cal_min, linetype="longdash", colour="green") +
    geom_vline(xintercept = cal_max, linetype="longdash", colour="blue")
```



The chunk below checks the actual data to confirm if the above prediction using the normal distribution is accurate:

```
(dairy_queen %>%
  filter(cal_fat < cal_max & cal_fat > cal_min) %>%
    summarise(percent = n() / nrow(dairy_queen))) * 100
```

```
##    percent
## 1 7.142857
```

Using the real values from Dairy Queen, we see that ~7% of their menu items actually have a calories from fat value within the daily recommended range. This is quite close to our prediction.

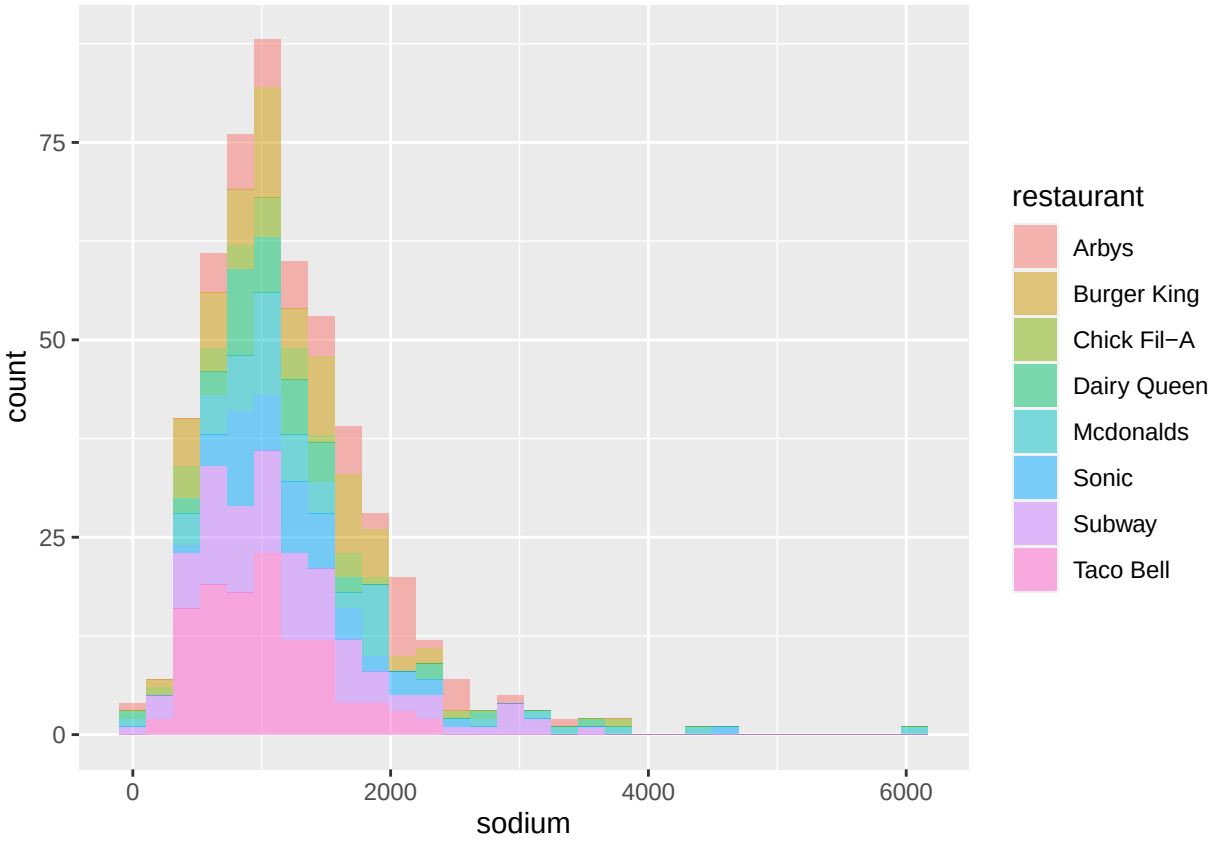
More Practice

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?

WJ Response:

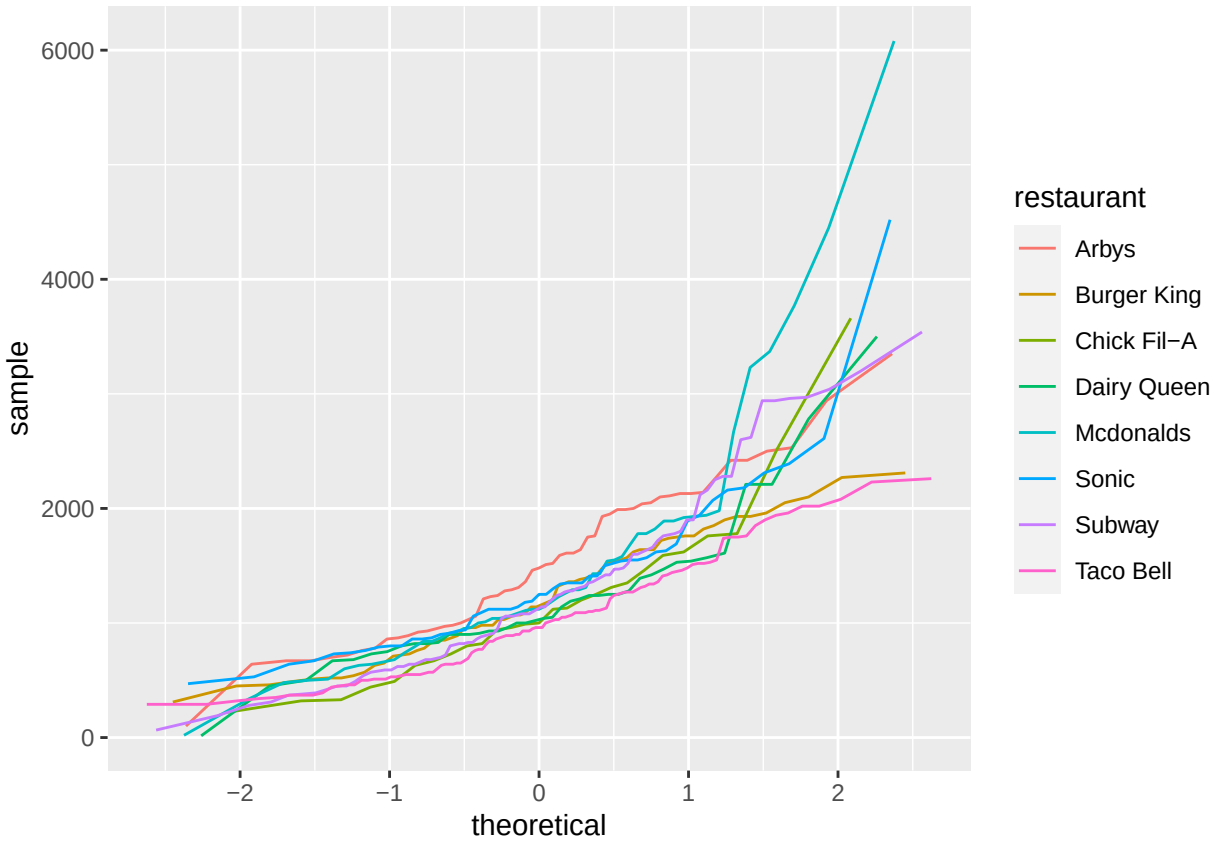
The code chunk below replicates the output shown above with the `qqnormsim` function, but using the `fastfood` dataframe:

```
ggplot(data = fastfood, aes(x=sodium, fill=restaurant)) +
    geom_histogram(alpha=0.5)
```



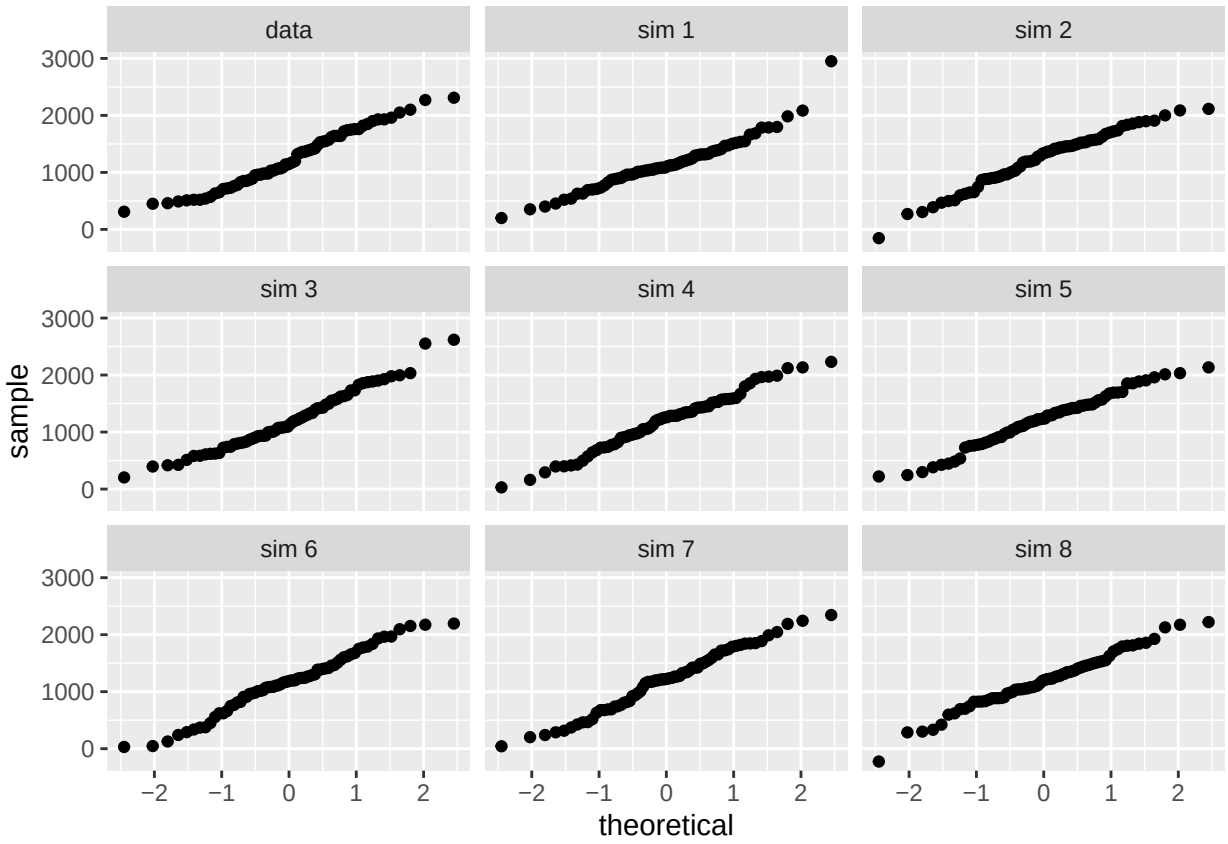
Comparing them in this way does not make it evidently clear which, if any of the restaurants have a normally distributed sodium values for their menu items. As such, the code chunk below creates Q-Q plots for each of them:

```
ggplot(data = fastfood,
       aes(sample = sodium, group=restaurant, colour=restaurant)) +
  geom_line(stat = "qq", )
```



Looking at the above graphic, it appears that Burger King has a Q-Q plot that is nearly diagonal. The below cell uses the `qqnormsim` function to confirm whether or not the this plot matches the Q-Q plots from simulated sets of normally distributed data using the mean and standard deviation of Burger King's sodium values:

```
qqnormsim(sample = sodium, data = filter(fastfood, restaurant=='Burger King'))
```



The graphic above provides strong evidence for the fact that the sodium values of Burger King's menu items are indeed normal: the Q-Q plot generated by the real data in the top left almost nearly matches eight other Q-Q plots generated using simulated data.

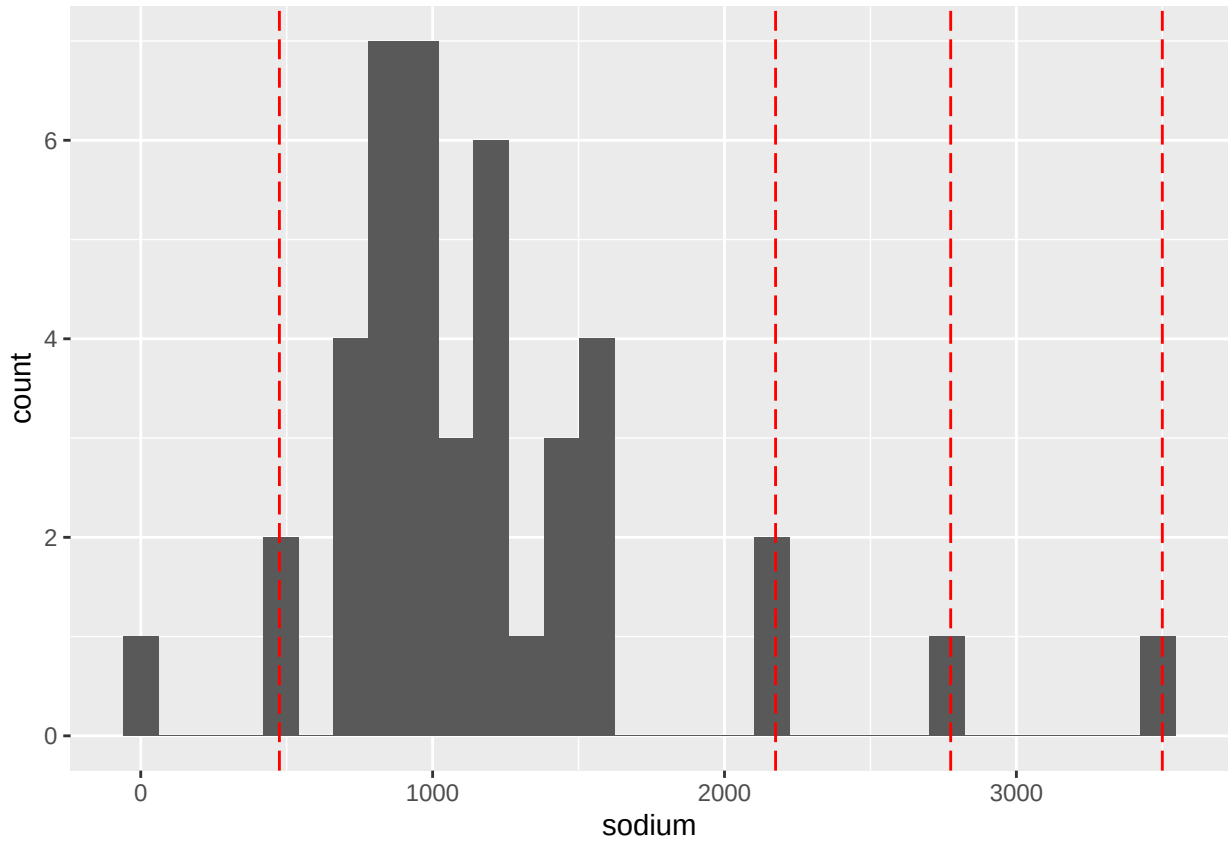
---

8. Note that some of the normal probability plots for sodium distributions seem to have a stepwise pattern. why do you think this might be the case?

WJ Response:

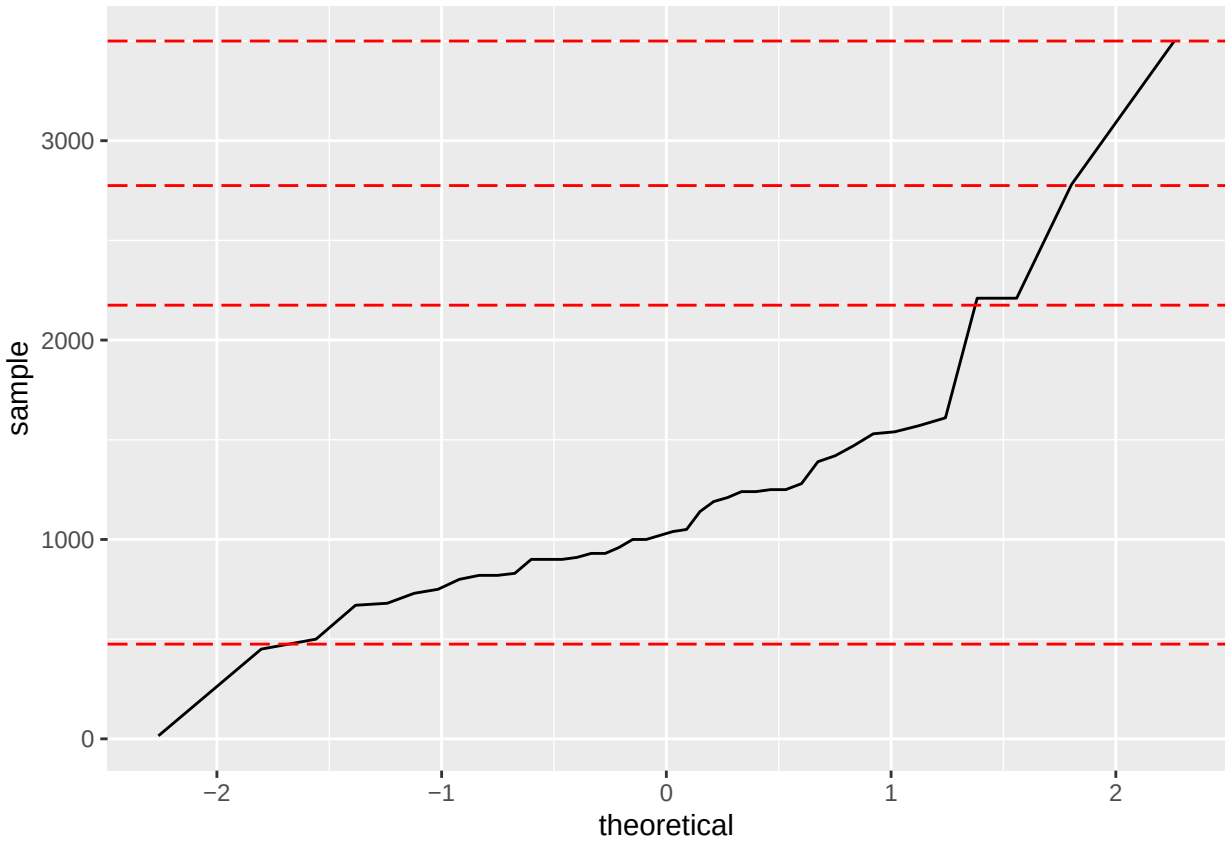
The code chunk below plots a histogram of sodium values of Dairy Queen's menu items:

```
restaurant_choice = 'Dairy Queen'

ggplot(data = filter(fastfood, restaurant==restaurant_choice),
       aes(x=sodium)) +
  geom_histogram() +
  geom_vline(xintercept = 475, linetype="longdash", colour="red") +
  geom_vline(xintercept = 2175, linetype="longdash", colour="red") +
  geom_vline(xintercept = 2775, linetype="longdash", colour="red") +
  geom_vline(xintercept = 3500, linetype="longdash", colour="red")
```



The red lines in the plot above estimate the position of the bins that are surrounded on either side by bins that are empty (with the exception of the bin at 0). The reason for this is to attempt to explain the step function line structure seen in the Q-Q plot generated for the same values, which is shown below:

```
restaurant_choice = 'Dairy Queen'

ggplot(data = filter(fastfood, restaurant==restaurant_choice),
       aes(sample = sodium)) +
  geom_line(stat = "qq", ) +
  geom_hline(yintercept = 475, linetype="longdash", colour="red") +
  geom_hline(yintercept = 2175, linetype="longdash", colour="red") +
  geom_hline(yintercept = 2775, linetype="longdash", colour="red") +
  geom_hline(yintercept = 3500, linetype="longdash", colour="red")
```



The horizontal lines shown above represent those same vertical lines that were superimposed on the previous histogram. It is clear from the above plot that the areas from the histogram that are surrounded by empty bins provide a good approximation for the areas on the Q-Q plot that have noticeable, long-lasting changes in its slope. As such, we can conclude that the step-like structure seen in a Q-Q plot is caused by data containing decently sized sub-ranges in which the frequency of values is low or 0.

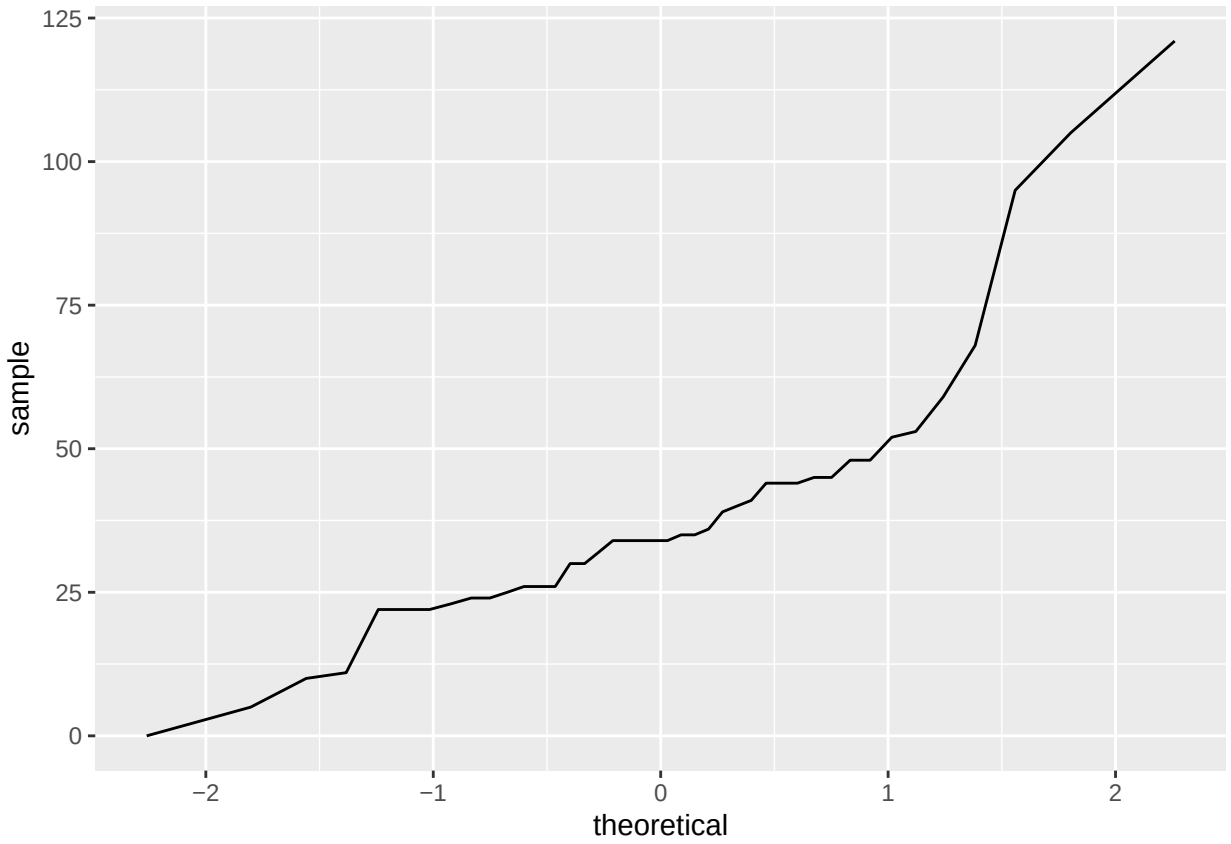
---

9. As you can see, normal probability plots can be used both to assess normality and visualize skewness. Make a normal probability plot for the total carbohydrates from a restaurant of your choice. Based on this normal probability plot, is this variable left skewed, symmetric, or right skewed? Use a histogram to confirm your findings.

WJ Response:

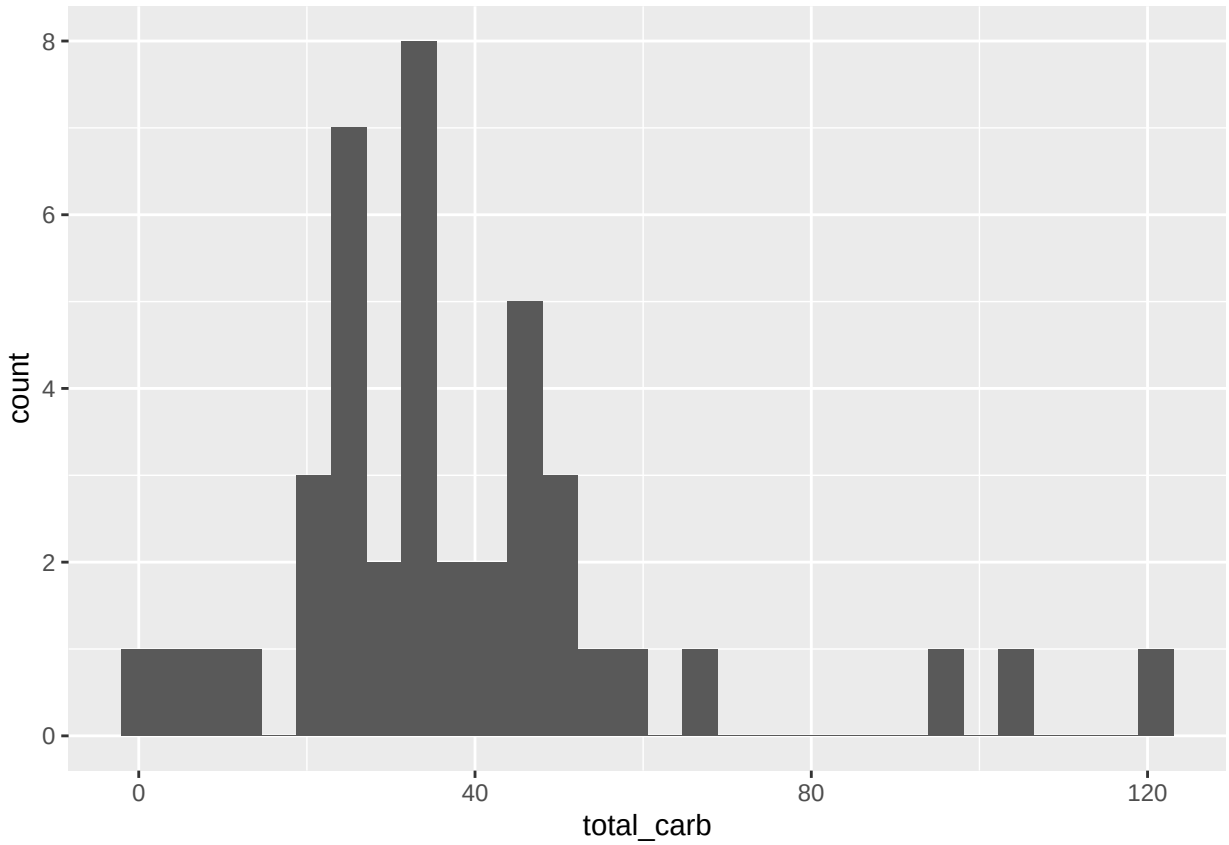
The code chunk below outputs a Q-Q plot of the carbohydrate values for Dairy Queens's menu items:

```
restaurant_choice = 'Dairy Queen'

ggplot(data = filter(fastfood, restaurant==restaurant_choice),
       aes(sample = total_carb)) +
  geom_line(stat = "qq", )
```



The Q-Q plot above exhibits a concave structure, in which it dips below the diagonal structure that should be expected if the data were truly normal. This means that the actual quantile values are below what is expected for much of the plot. The only way this can happen is if most of the data is clustered towards the smaller end of the range, seeing as a larger than expected percentage of the data is held within the lower quantiles. Graphically, this kind of Q-Q plot should then represent data that is right-skewed, in which the majority of the points land towards the left-hand side of the distribution. This can be confirmed by outputting the histogram, which is done below:

```
ggplot(data = filter(fastfood, restaurant==restaurant_choice),
       aes(x=total_carb)) +
  geom_histogram()
```



As is clear in the plot above, a majority of the total carbohydrate values for Dairy Queen's menu items are on the left hand side of the graph, confirming our hypothesis that the distribution would be right-skewed.