The normal distribution

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)
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.

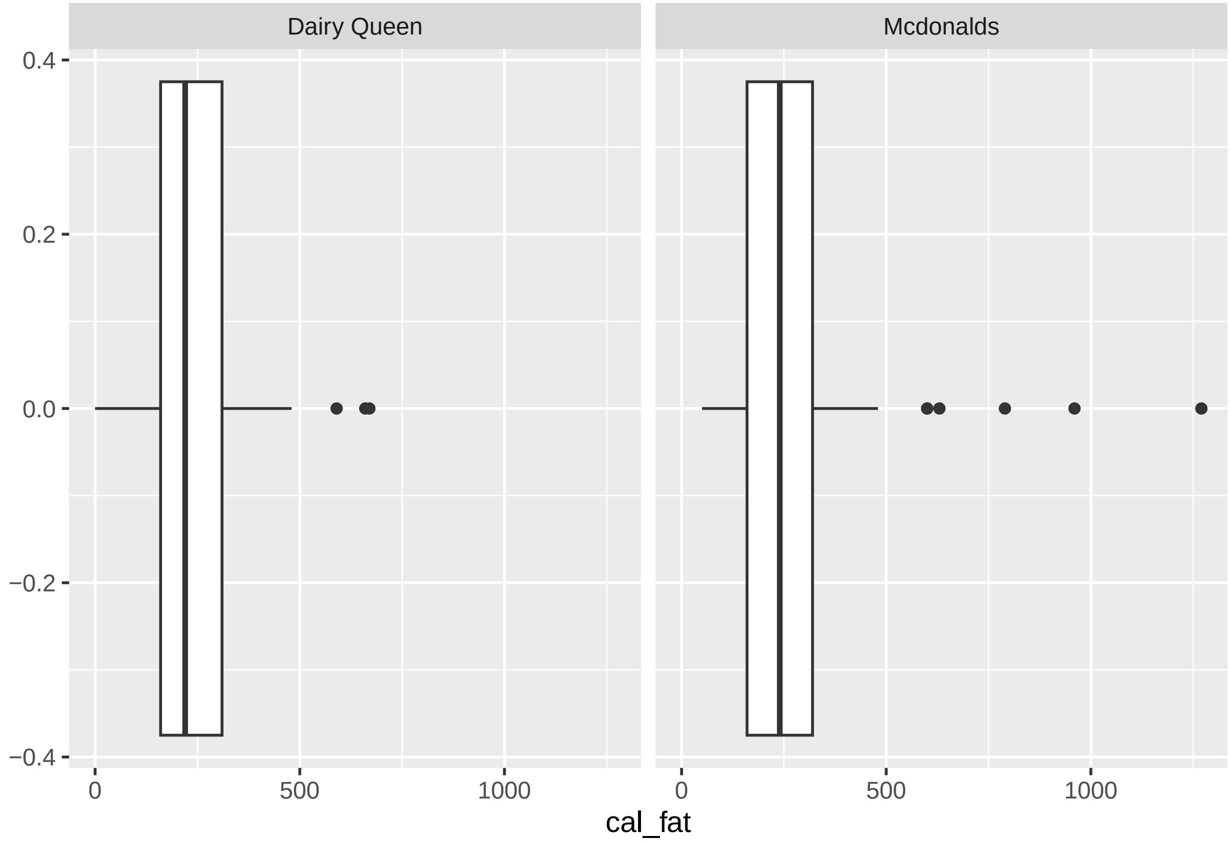
Let's first focus on just products from McDonalds and Dairy Queen.

```r
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?

Insert your answer here

```r
mcdonalds_and_dq <- fastfood |>
  filter(
    restaurant == "Mcdonalds" | restaurant == "Dairy Queen"
  )

ggplot(data = mcdonalds_and_dq, aes(x = cal_fat)) +
  geom_boxplot() +
  facet_wrap(~restaurant)
```



Above, we've created a box and whisker plot. Judging from the plot, we can see that the center/median is fairly similar although Dairy Queen's median is a little bit lower. The shape is also fairly similar with a pretty tight IQR, meaning that most of the items have pretty similar calories from fat.

The outliers are where we can observe differences where Mcdonalds has items which contain much more calories from fat.

```
mcdonalds_and_dq |>
  group_by(restaurant) |>
  summarise(
    median = median(cal_fat),
    variance = var(cal_fat)
  )
```

```
## # A tibble: 2 x 3
##   restaurant  median variance
##   <chr>        <dbl>    <dbl>
## 1 Dairy Queen    220   24488.
## 2 Mcdonalds      240   48796.
```

We can see that verified with this table. The variance of fat from calories for mcdonalds is much greater but the medians are the same.

End of Answer

The normal distribution

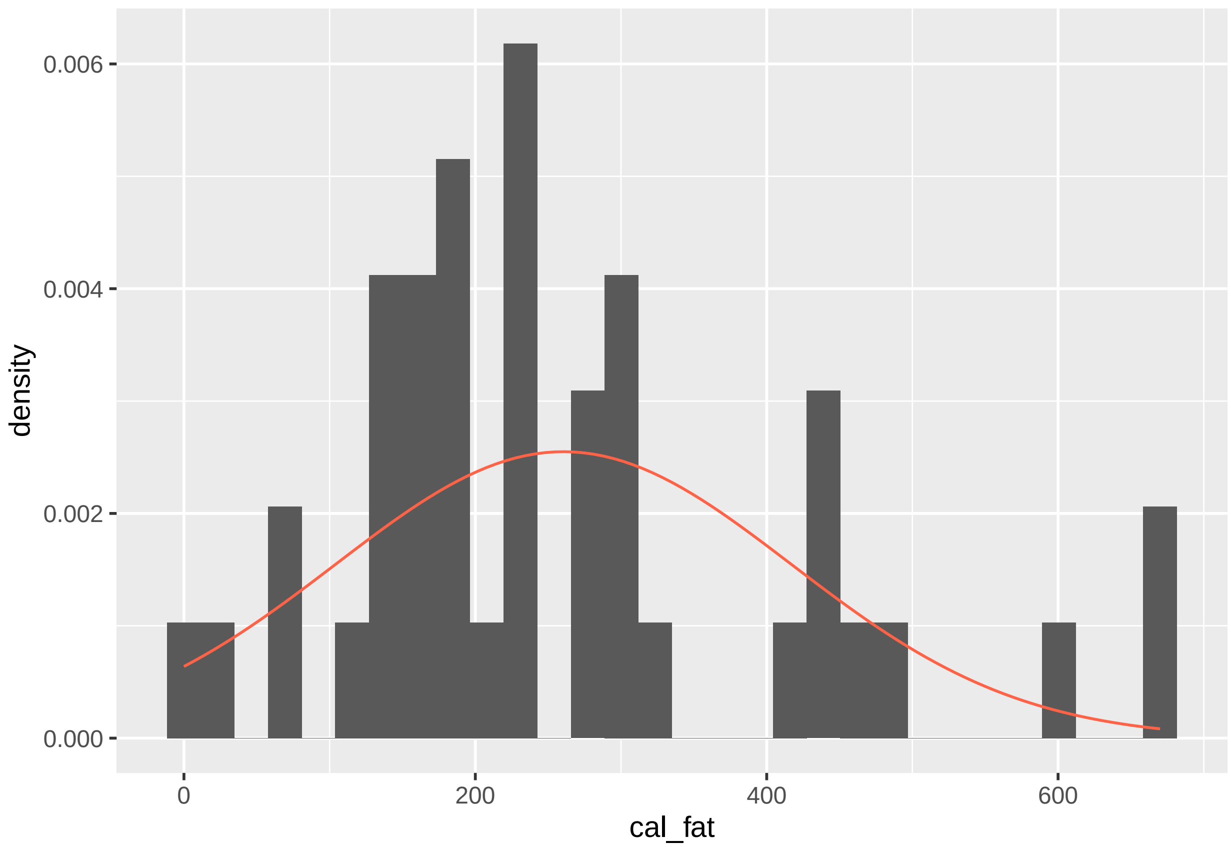
In your description of the distributions, did you use words like *bell-shaped*or *normal*? It's tempting to say so when faced with a unimodal symmetric distribution.

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?

Insert your answer here

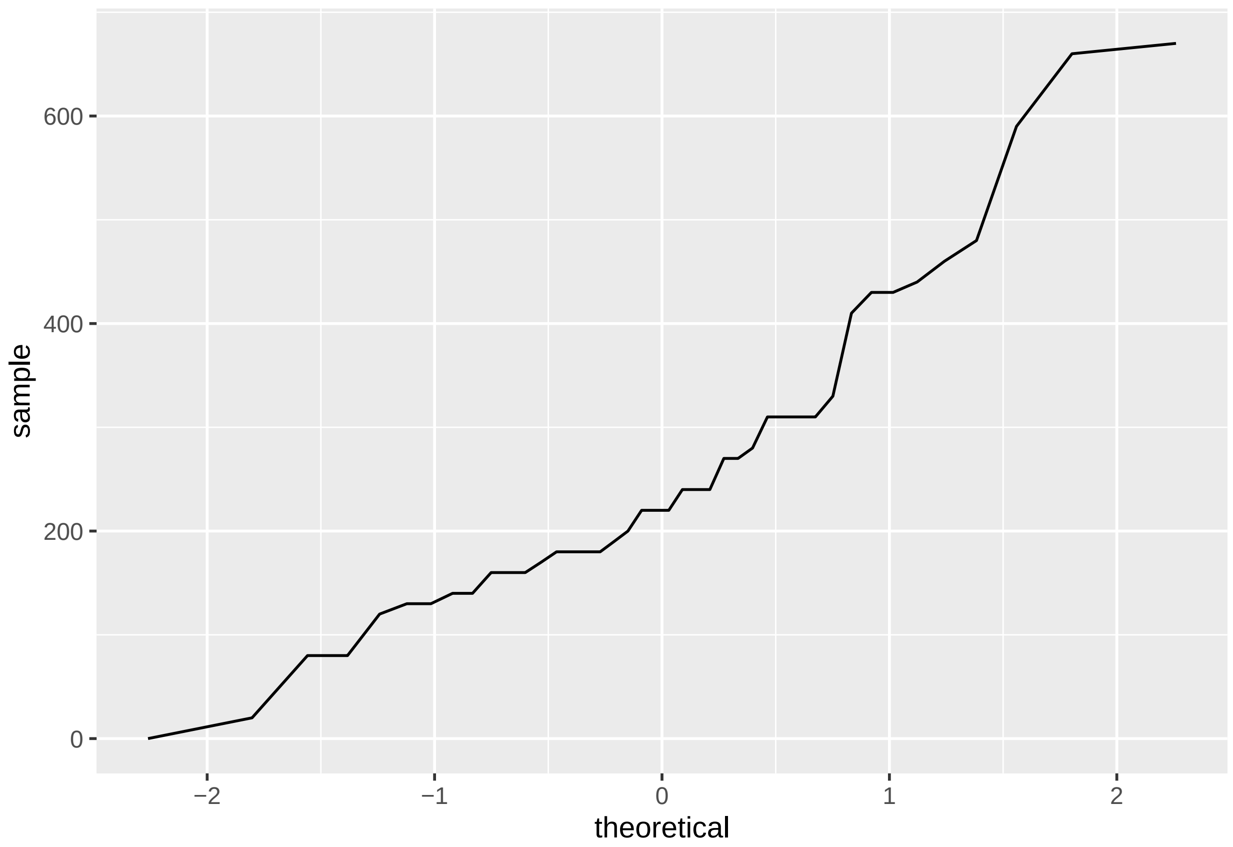
Visually, this plot doesn't seem to follow a normal distribution at all. This can be seen as the tails seem to have a fair amount of values and much of the data is pretty erratically distributed between the minimum and maximum values.

End of Answer

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`.

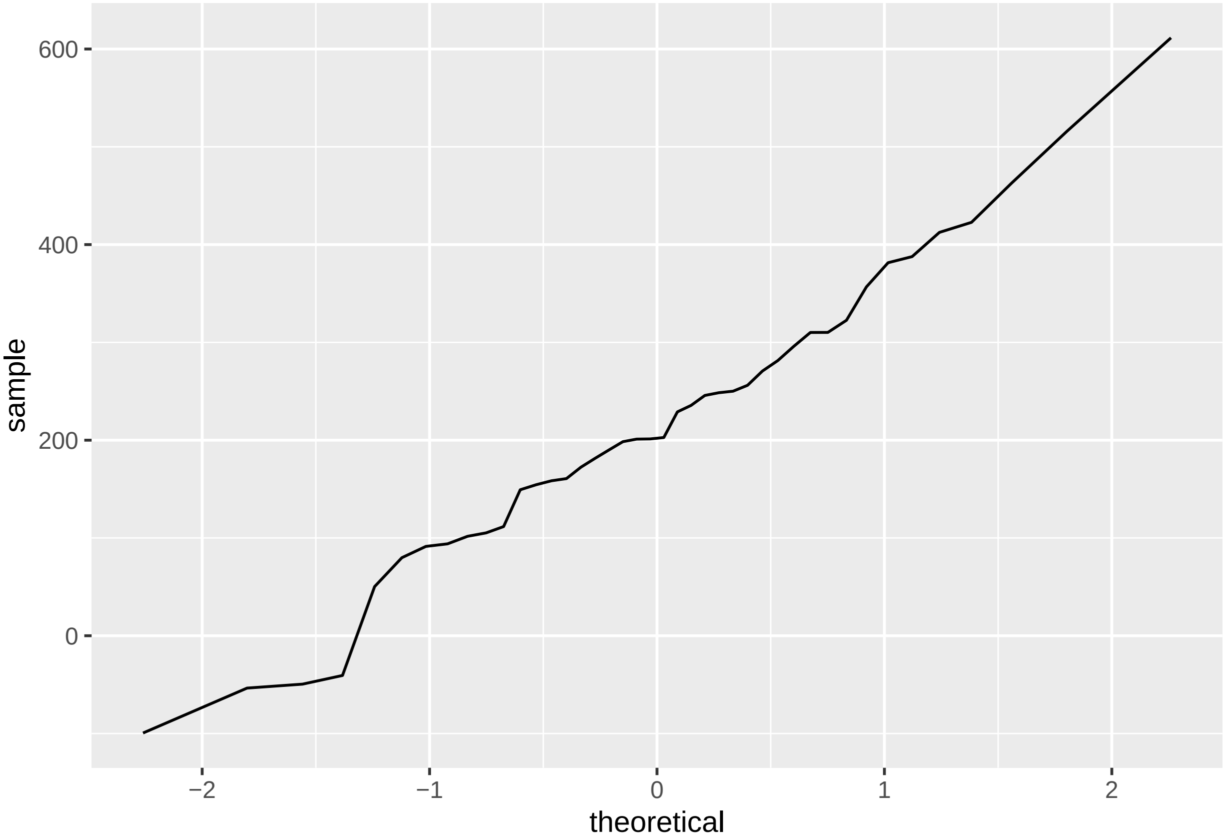
```r
set.seed(20240224)
sim_norm <- rnorm(n = nrow(dairy_queen), mean = dqmean, sd = dqsd)
```

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.)

Insert your answer here

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

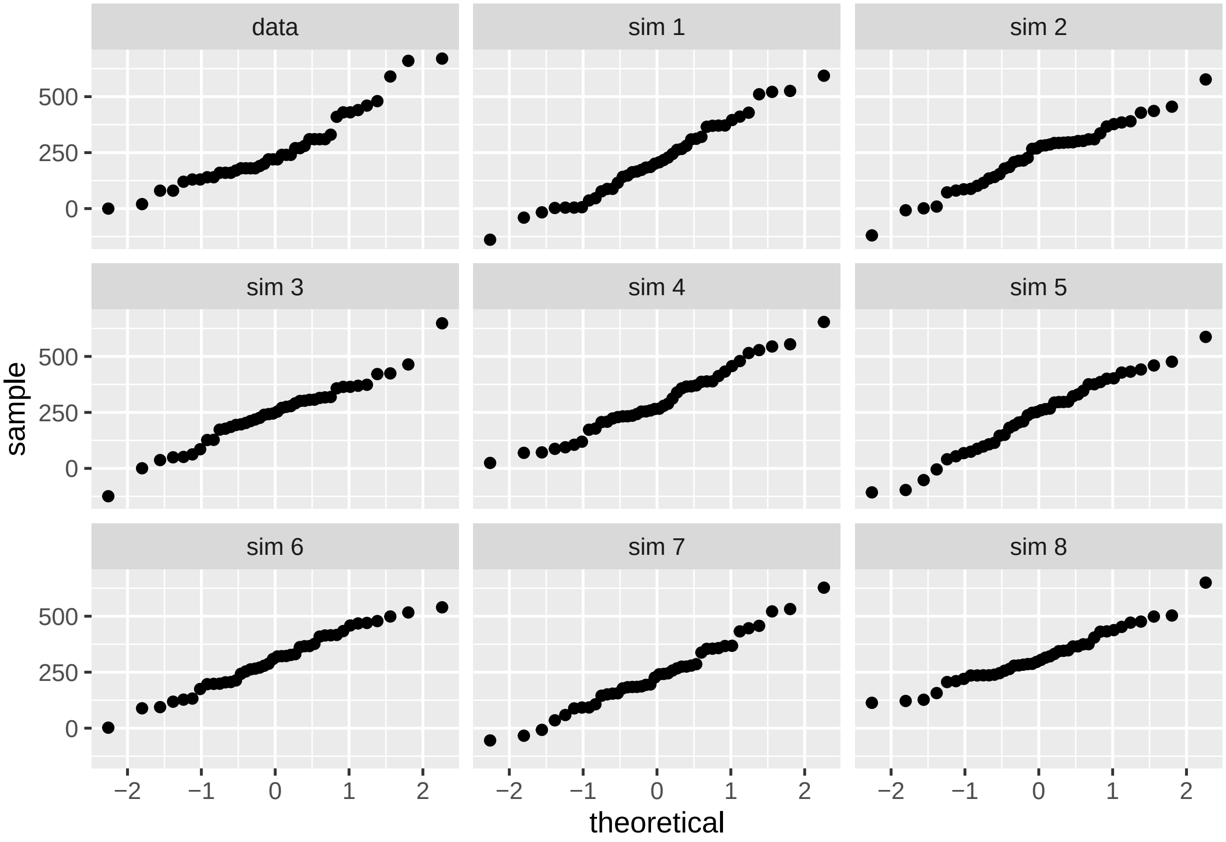


In this plot, all points don't seem to follow a diagonal line. This plot has a negative value for when theoretical is -2, which doesn't make sense as there aren't any negative calories. To me, since we know that there is a practical limit of x=0. Aside from that the data seems to visually be visually similar to the plot created on the real data.

End of Answer

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?

Insert your answer here

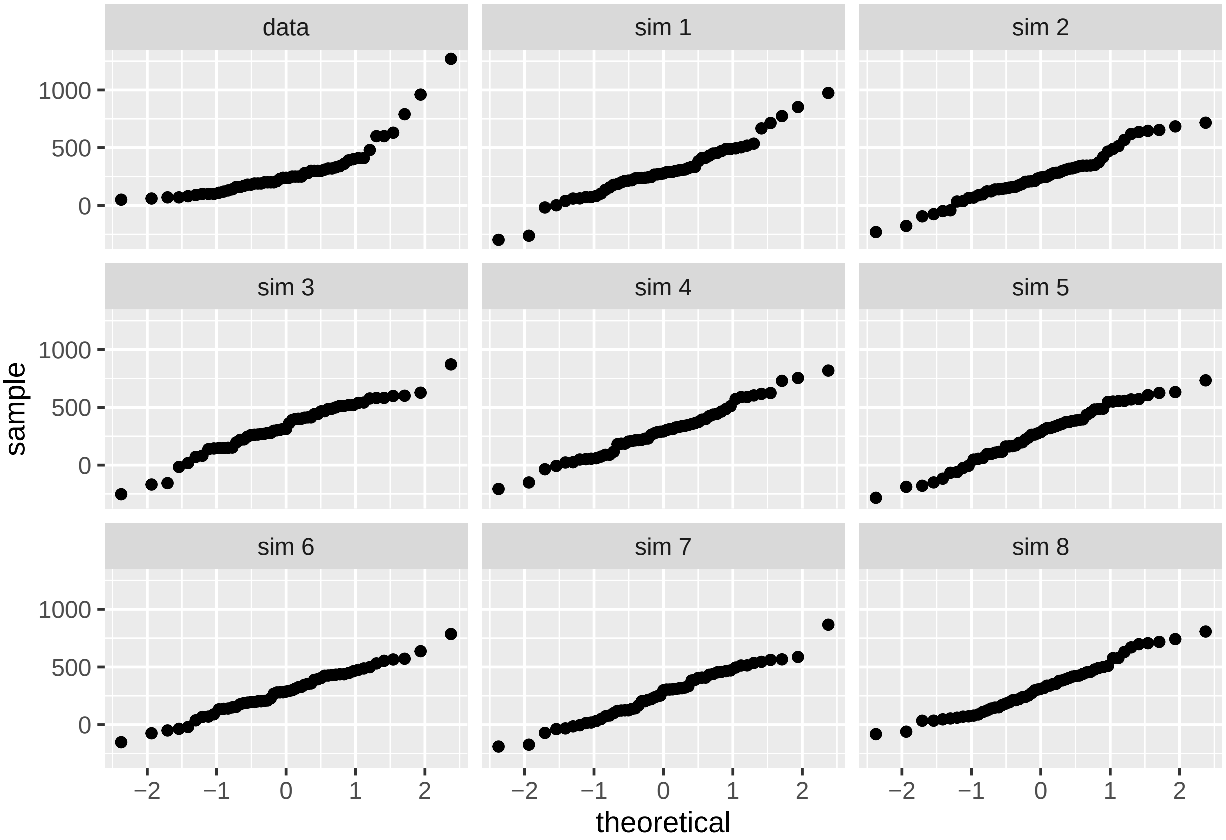
Compared to all of the other plots, they do look somewhat similar. From what was discussed in this lab, this would suggest that the data is nearly normal.

End of Answer

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

Insert your answer here

```r
# Create the simulations using qqnormsim
qqnormsim(sample = cal_fat, data = mcdonalds)
```



Looking at the graph generated above, we can see that the mcondalds dataset has the same issue where each simulation has entries with negative calories. Aside from that, we can also observe that the actual mcdonalds dataset resembles an exponential graph. When comparing that to the simulations, it seems that the mcdonalds is less normal than the Dairy Queen dataset

End of Answer

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?

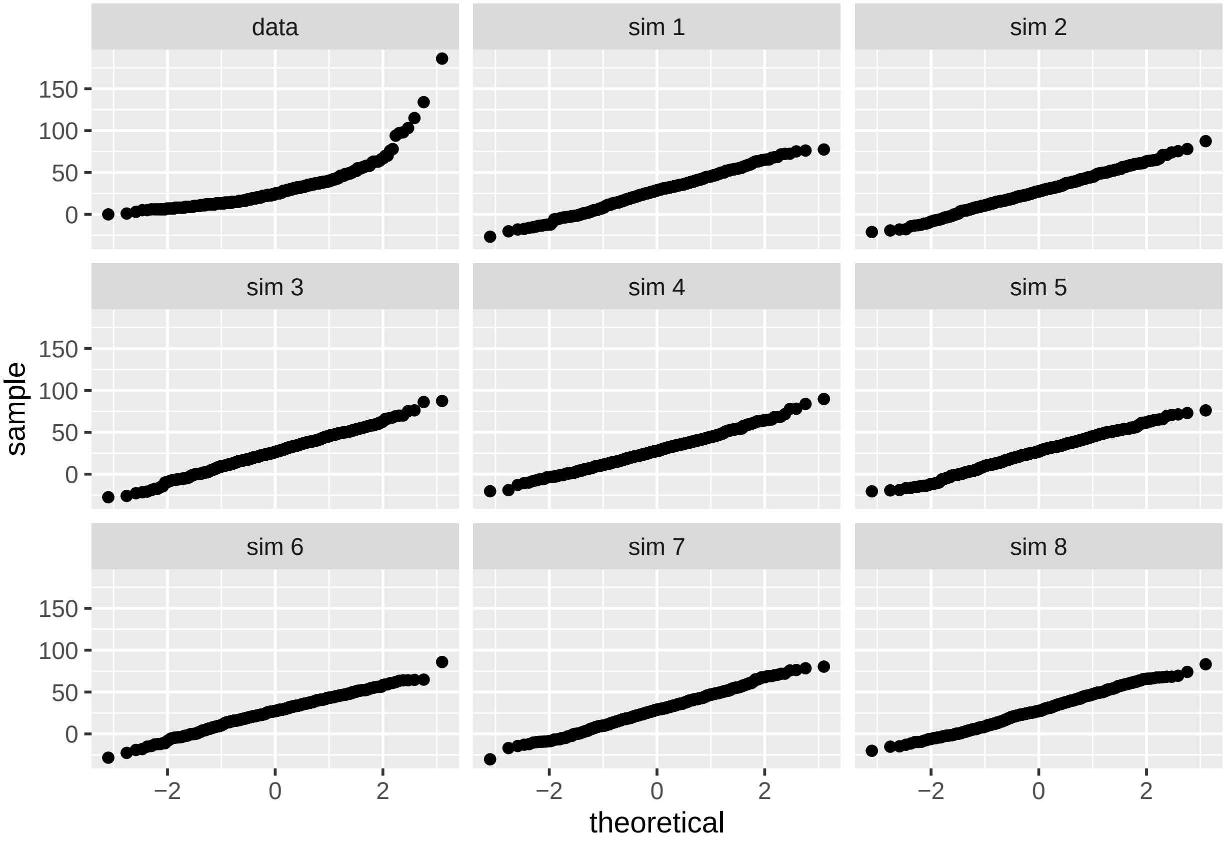
Insert your answer here

Question 1 - What is the probability that any item from a fast food restaurant has at least 50g of protein?

Question 2 - What is the probability of any fast food item having more sugar than the 25g that is recommended by the American Heart Association?

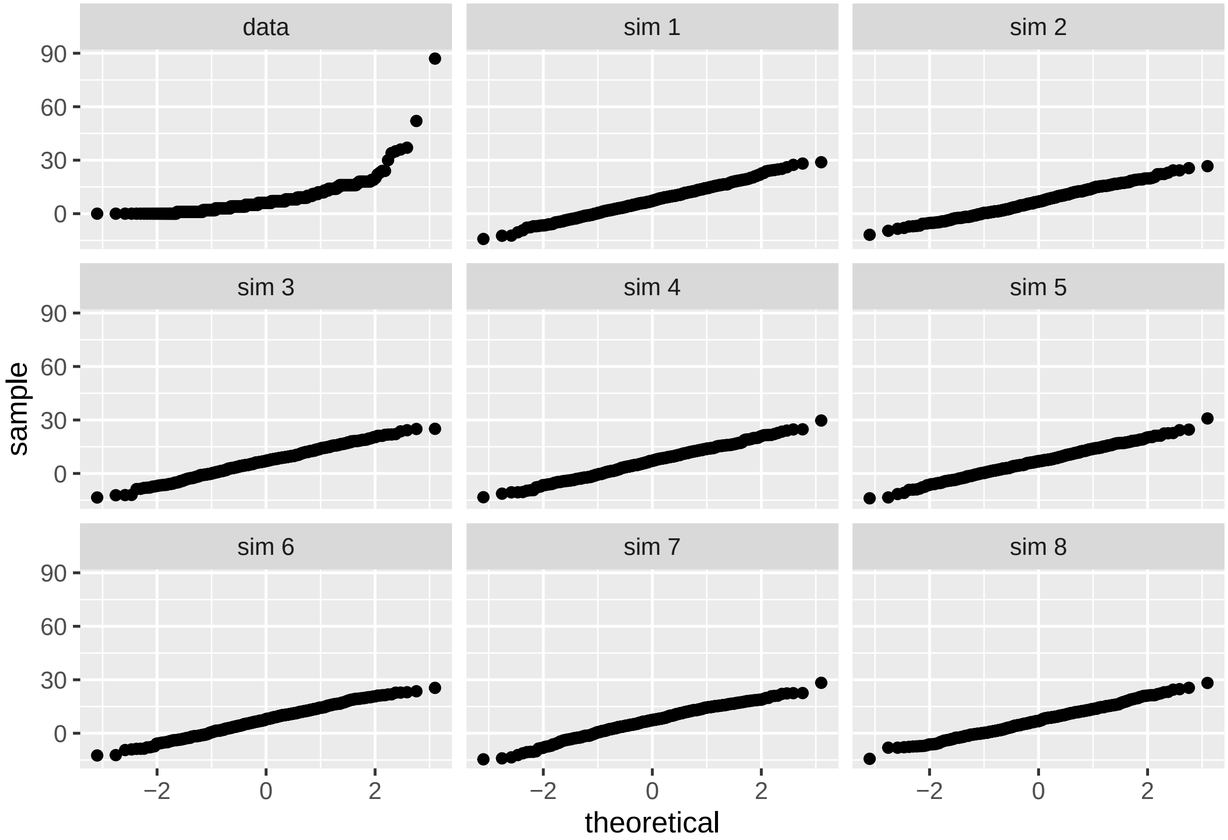
We will check to make sure that these distributions are normal first, but will assume it anyway afterwards

```
# First fill any NAs with 0 in the protein field
q1_ff <- fastfood
q1_ff$protein <- replace_na(fastfood$protein, 0)

qqnormsim(sample = protein, data = q1_ff)
```



The protein distribution looks similar to the mcondalds distribution of cal_fat.

```
q2_ff <- fastfood
q2_ff$sugar <- replace_na(fastfood$sugar, 0)

qqnormsim(sample = sugar, data = q1_ff)
```



The sugar distribution looks even less normal than the protein one. But continuing anyway:

```
protein_threshold <- 50

protein_mean <- mean(q1_ff$protein)
protein_sd   <- sd(q1_ff$protein)

protein_theo <- 1 - pnorm(
  q = protein_threshold,
  mean = protein_mean,
  sd = protein_sd
)

print(
  paste(
    "There is a",
    round(100 * protein_theo, 2),
    "% chance than a random fast food item will have at least",
    protein_threshold,
    "g of protein."
  )
)
```

```
## [1] "There is a 10.54 % chance than a random fast food item will have at least 50 g of protein."
```

```r
q1_ff |>
  filter(protein > protein_threshold) |>
  summarise(percent = n() / nrow(q1_ff))
```

```
## # A tibble: 1 x 1
##   percent
##     <dbl>
## 1  0.0757
```

For the first question, we can see that there is a theoretical 10.54% chance that a random observation will have at least 50 grams of protein and empirically there is a 7.57% chance. There is a difference of 2.97% chance between these two values.

Moving onto the sugar question:

```r
sugar_threshold <- 25

sugar_mean <- mean(q2_ff$sugar)
sugar_sd   <- sd(q2_ff$sugar)

sugar_theo <- 1 - pnorm(
  q = sugar_threshold,
  mean = sugar_mean,
  sd = sugar_sd
)

print(
  paste(
    "There is a",
    round(100 * sugar_theo, 2),
    "% chance than a random fast food item will have at least",
    sugar_threshold,
    "g of sugar."
  )
)
```

```
## [1] "There is a 0.44 % chance than a random fast food item will have at least 25 g of sugar."
```

```r
q2_ff |>
  filter(sugar > sugar_threshold) |>
  summarise(percent = n() / nrow(q2_ff))
```

```
## # A tibble: 1 x 1
##   percent
##     <dbl>
## 1  0.0136
```

For the second question, we can see that there is theoretically a 0.44% chance that a random observation will have at least 25g of sugar and empirically there is an 1.36% chance. This results in a difference of 0.92%.

Although the magnitude of differences are pretty great (2.97% and 0.92%), I believe the protein theoretical and empirical values are closer. I believe so because the percent difference for protein is 40% while it was 67.6% for sugar. A quick definition on my calculation of percent difference:

$$\frac{P_{theoretical} - P_{empirical}}{P_{empirical}}$$

End of Answer

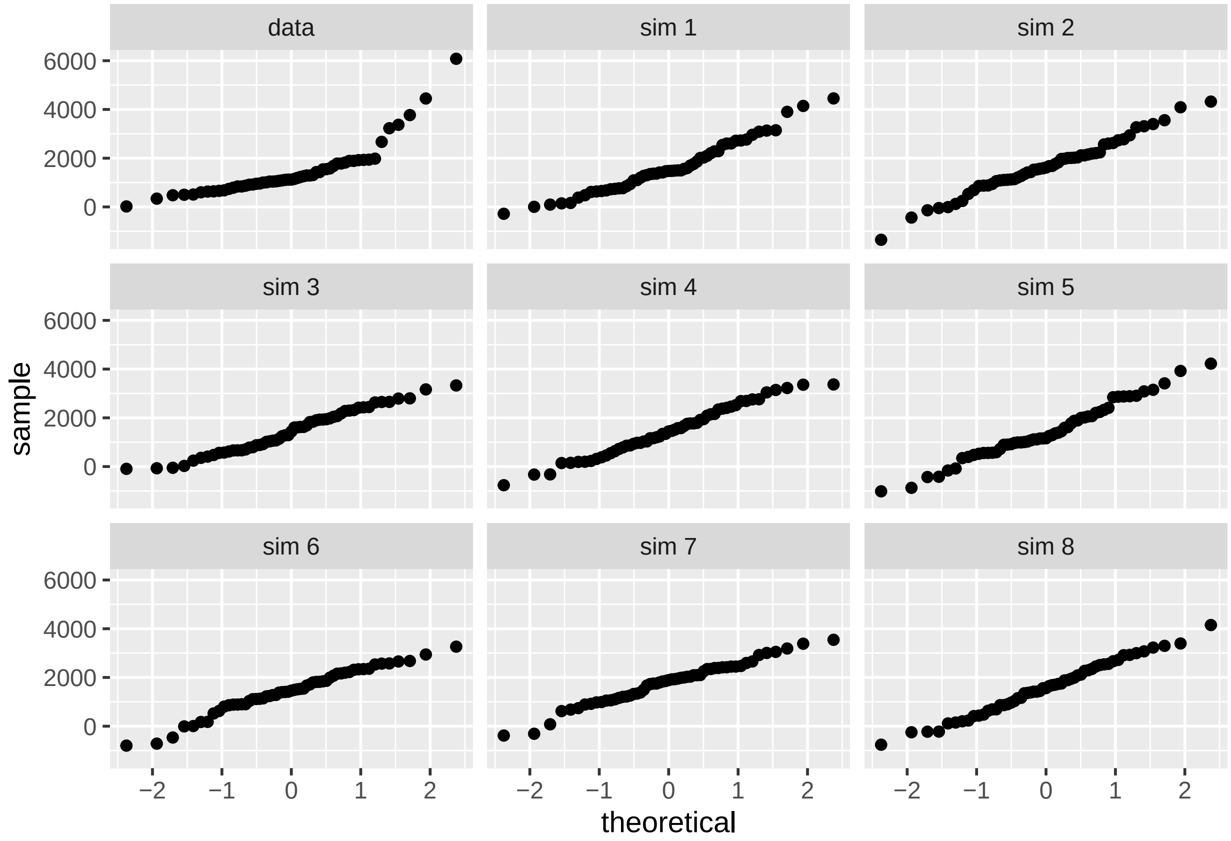
More Practice

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?

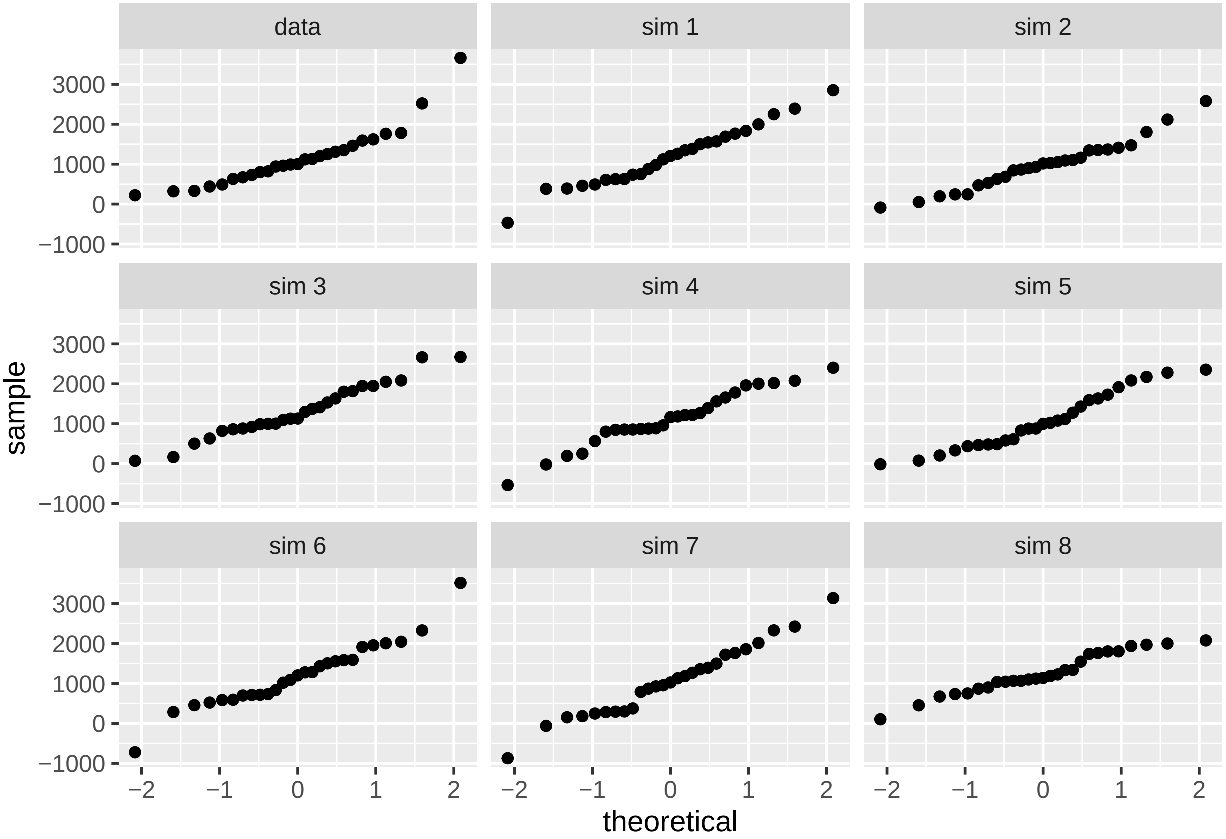
Insert your answer here

We'll iterate through the restaurants and generate a QQ plot for each's sodium distribution:

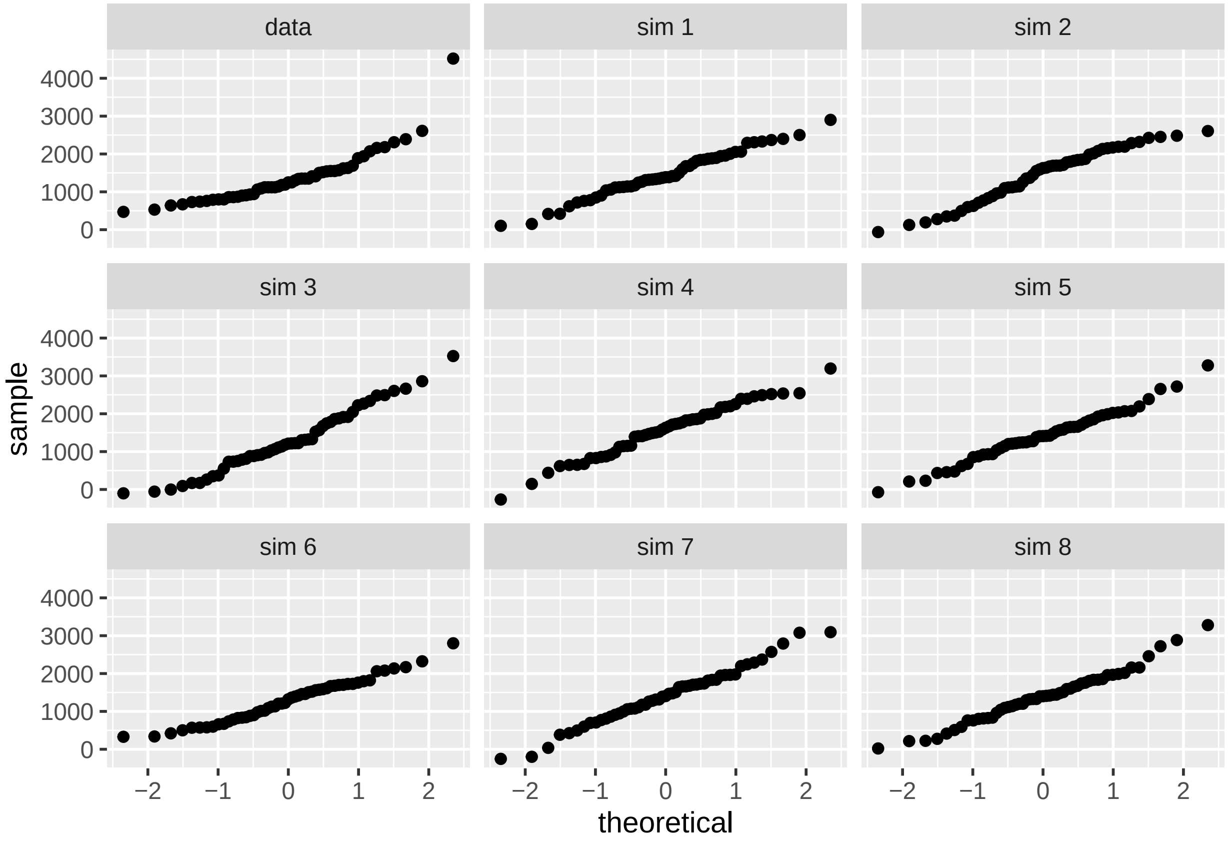
```r
qqnormsim(
  sample = sodium,
  data = fastfood |>
    filter(restaurant == "Mcdonalds")
)
```



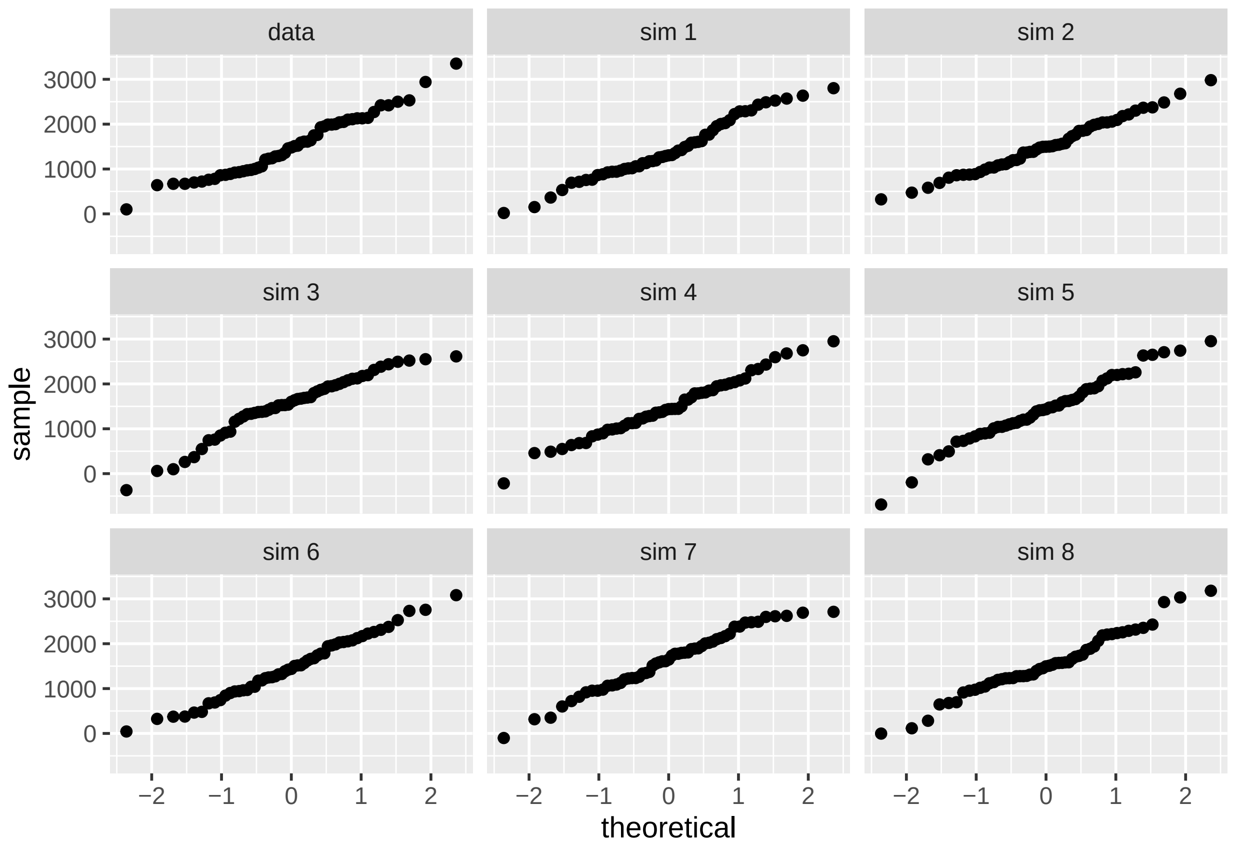
```r
qqnormsim(
  sample = sodium,
  data = fastfood |>
    filter(restaurant == "Chick Fil-A")
)
```



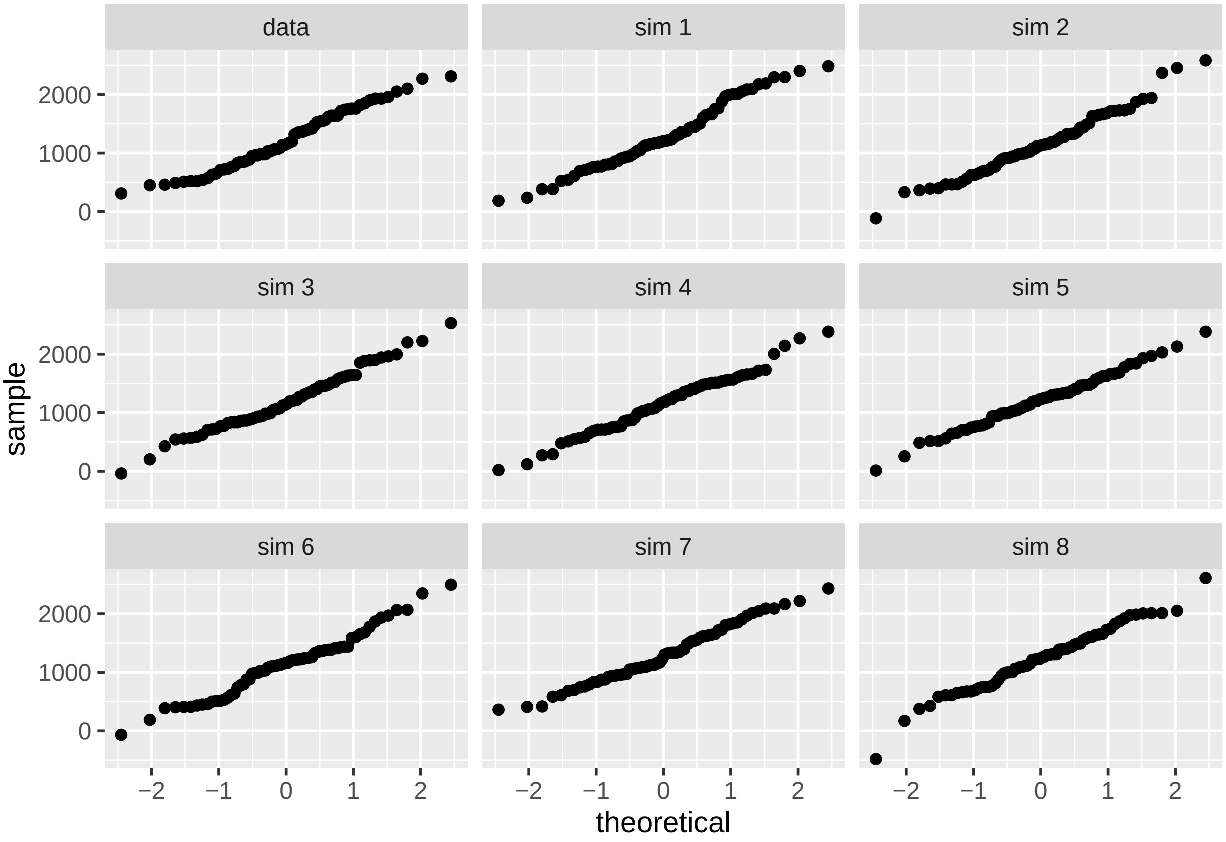
```r
qqnormsim(
  sample = sodium,
  data = fastfood |>
    filter(restaurant == "Sonic")
)
```



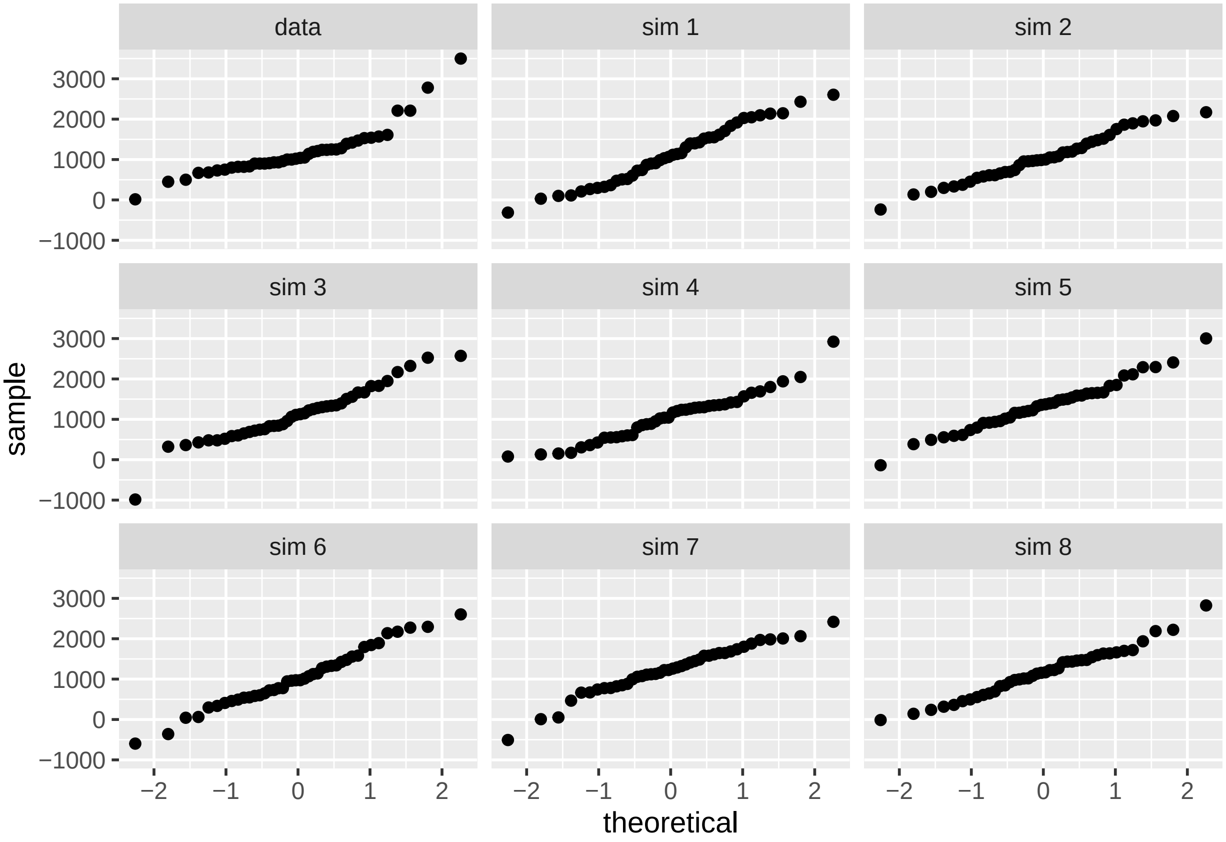
```
qqnormsim(
  sample = sodium,
  data = fastfood |>
    filter(restaurant == "Arbys")
)
```



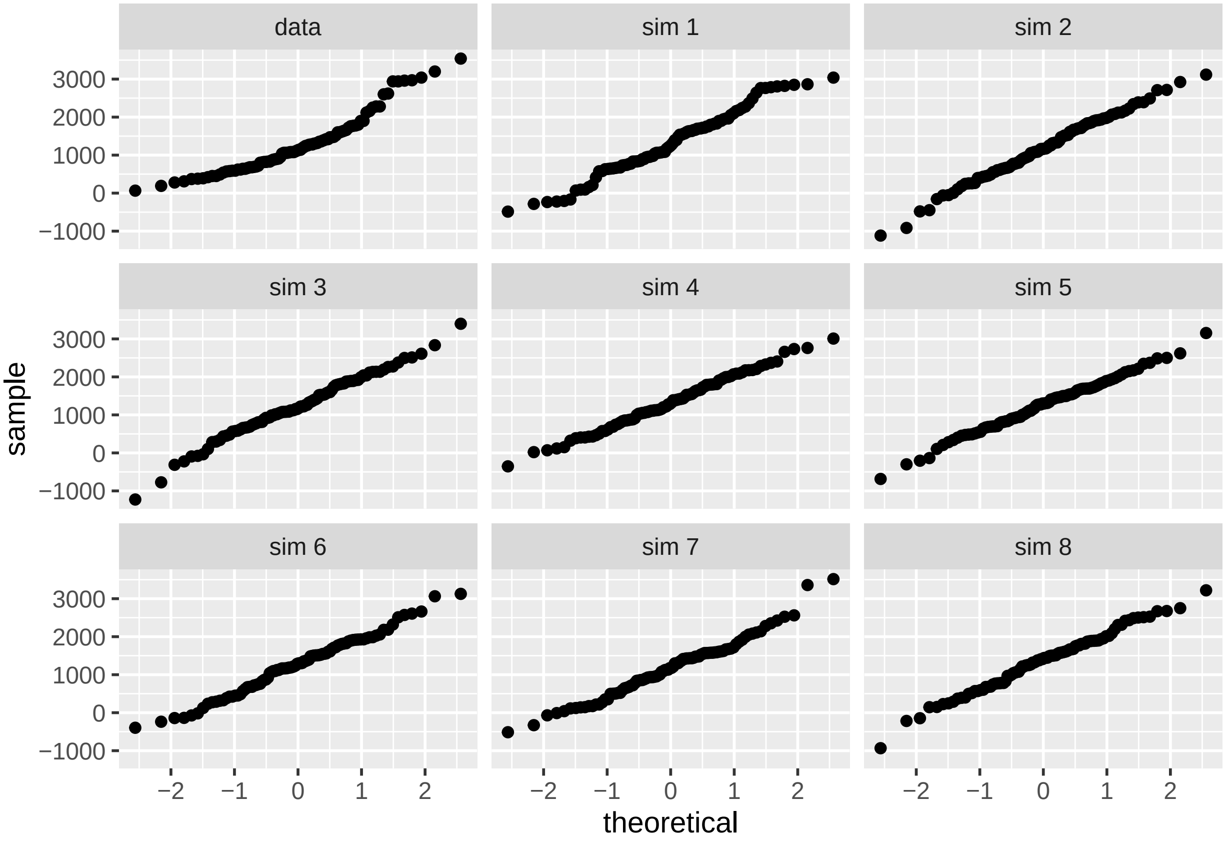
```
qqnormsim(
  sample = sodium,
  data = fastfood |>
    filter(restaurant == "Burger King")
)
```



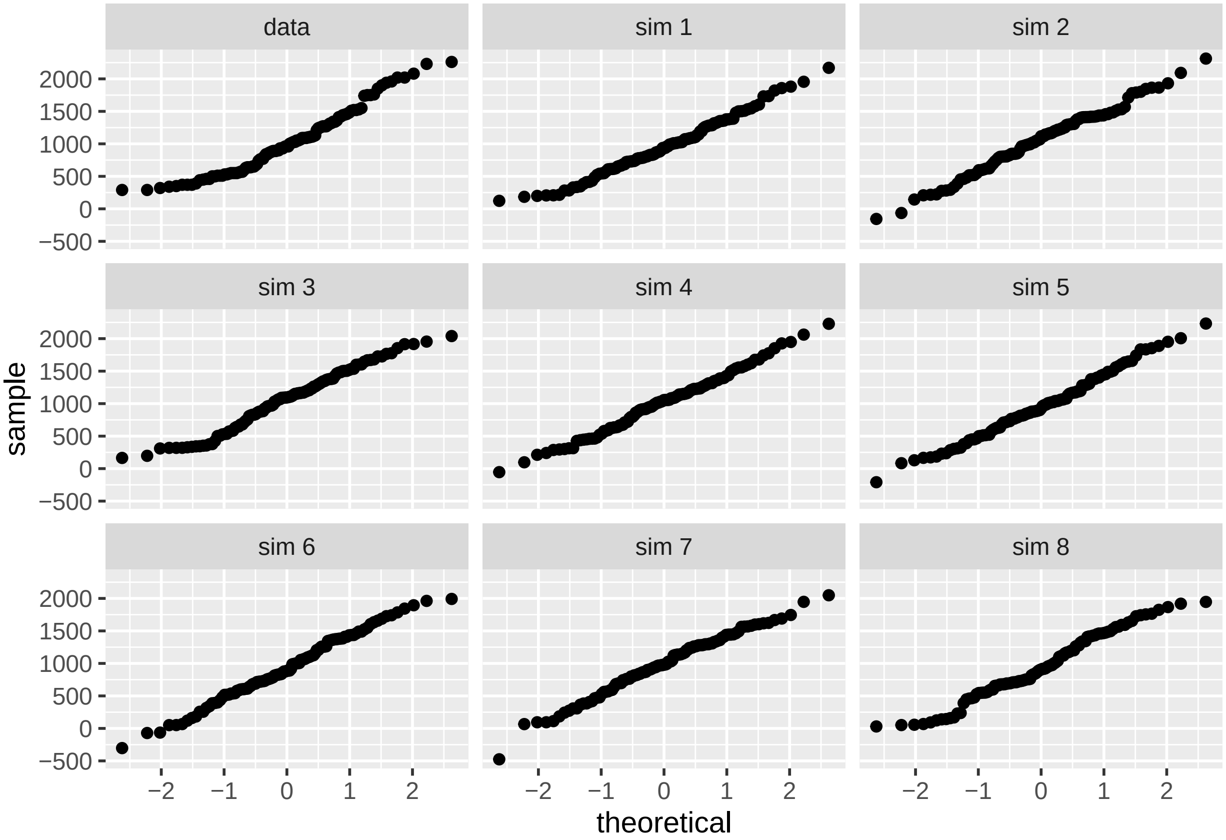
```
qqnormsim(
  sample = sodium,
  data = fastfood |>
    filter(restaurant == "Dairy Queen")
)
```



```
qqnormsim(
  sample = sodium,
  data = fastfood |>
    filter(restaurant == "Subway")
)
```



```
qqnormsim(
  sample = sodium,
  data = fastfood |>
    filter(restaurant == "Taco Bell")
)
```



Visually, it seems that Arbys is the most normal as all the simulations and the actual data appears to be the most diagonal. In order to better define which is most normal, it would be better if we can have a numeric value that we can compare across each of these.

End of Answer

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?

Insert your answer here

This could be due to the nature of the menus where clusters of small items like sides and other lower sodium menu items will create a step that will remain in a similar sodium level until the next class of food items is introduced. As a concrete example, I fully expect a side salad or a junior burger to have significantly less sodium than an entree cheeseburger.

```
fastfood |>
  summarise(mean_all_sodium = mean(sodium))
```

```
## # A tibble: 1 x 1
##   mean_all_sodium
##             <dbl>
## 1           1247.
```

```
fastfood |>
  filter(grepl("Salad", item)) |>
  summarise(mean_salad_sodium = mean(sodium))
```

```
## # A tibble: 1 x 1
##   mean_salad_sodium
##               <dbl>
## 1              940.
```

Here's an example with the salad population vs the total dataset population. We can see that the mean sodium content across the entire menu is significantly higher than that of the salads.

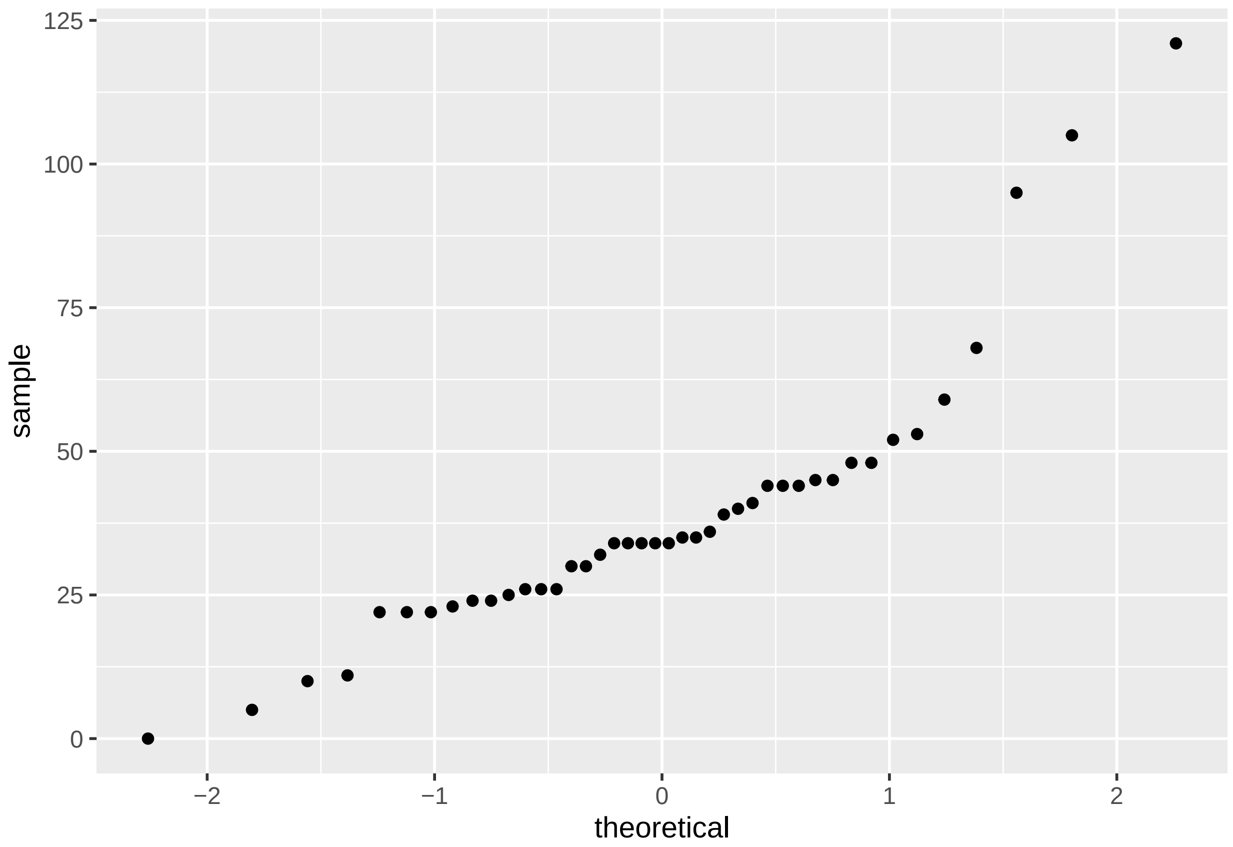
End of Answer

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.

Insert your answer here

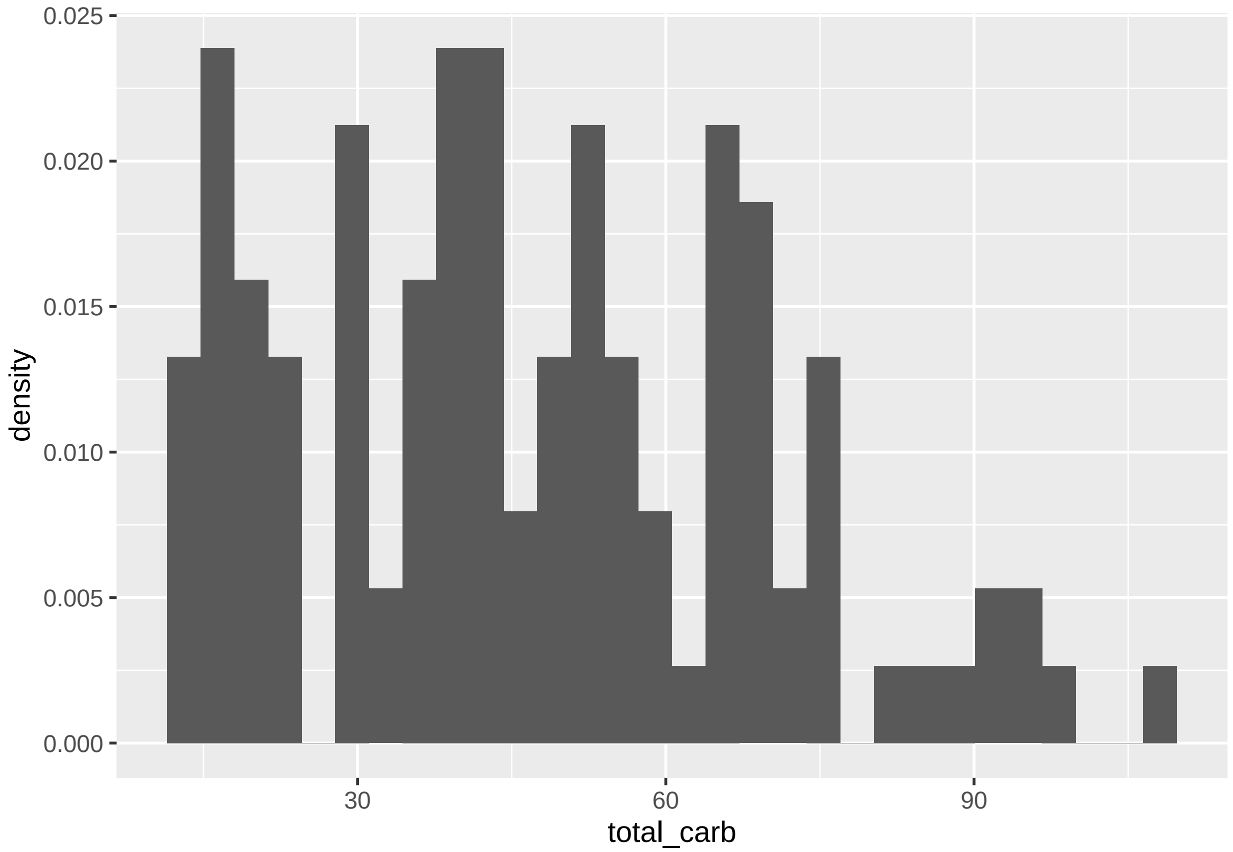
I'll be looking at the carbohydrates level of Taco Bell:

```
taco_bell <- fastfood |>
  filter(restaurant == "Taco Bell")

ggplot(data = dairy_queen, aes(sample = total_carb)) +
  geom_point(stat = "qq")
```



Judging from this QQ plot, I would assume that the dataset is left skewed because of the general caved-downward shape of the Q-Q plot.

```
ggplot(data = taco_bell, aes(x = total_carb)) +
  geom_histogram(aes(y = ..density..))
```



Looking at the histogram, it seems that the data is a bit left skewed where there is a much higher density of observations on the left side of the plot.

End of Answer