Lesson 10 - Normal Distribution!

YOUR NAME

2023-12-05

Contents

Learning Goals

- Visualize properties of the normal distribution.
- Understand the Central Limit Theorem.
- Calculate sampling properties of sample means.
- Decide whether a data set likely comes from a normal distribution

Tutorial

1. The basics: `rnorm()`,`pnorm()`, `qnorm()`, ‘dnorm()“

`rnorm()`: Random samples from a normal distribution

All of these functions take at least some of the arguments below.

The function `rnorm()` will return a vector of numbers, all drawn randomly from a normal distribution. It takes three arguments:

- n: how many random numbers to generate (the length of the output vector)
- mean: the mean of the normal distribution to sample from
- sd: the standard deviation of the normal distribution

For example, the following command will give a vector of 20 random numbers drawn from a normal distribution with mean 13 and standard deviation 4:

```r
#try running this a few times in a row
rnorm(n = 20, mean = 13, sd = 4)
```

```
##  [1] 17.222705 13.470799 12.188386 19.148775 12.789392 17.811998  3.662655
##  [8] 12.041375 11.307126 12.158010 13.970541  7.907115 23.070858  5.282013
## [15] 14.717953 10.305893 14.969731  6.724455 13.840835  9.859157
```

Ex.1.1) Remember you can always set a seed to make your random sampling reproducible:

```r
#set a seed with any number
set.seed(123)
#sample 3 values from a normal with mean 13 and sd 4
rnorm(n = 3, mean = 13, sd = 4)
```

```
## [1] 10.75810 12.07929 19.23483
```

If you set the seed 123 and then run the code again, then you should always get the same result: `[1] 10.75810 12.07929 19.23483`. Try it a few times in the code block above.

`pnorm()`: Probabilities under the normal curve

Often, we are interested in finding probabilities under the normal curve. The command `pnorm(Y)` gives the probability of obtaining a value less than Y under the normal distribution. Set the arguments `mean =` and `sd =`to the mean and standard deviate of the desired normal distribution.

E.g. NASA is obsessed with heights and deems men shorter than 157.5cm tall unfit to be pilots. Assuming NASA only recruits from the U.S. population and that the mean height of U.S. mean follows an approximately normal distribution with mean=177.6cm and standard deviation 9.7, what is the probability that a randomly selected man will be too short to be a pilot?

```r
#probability of observing q value < 157.5 in a normal with mean 1776 and sd 9.7
pnorm(q=157.5, mean = 177.6, sd = 9.7,lower.tail =T )
```

```
## [1] 0.01912503
```

Ex1.2) NASA also doesn't like men who are too tall (similar restrictins are applied to women, by the way), and considers men above 190.5cm in height to be too tall. What is the probability that a randomly sampled U.S. man is too tall to be a pilot?

```r
#your code here
pnorm(q=190.5, mean = 177.6, sd = 9.7,lower.tail =F)
```

```
## [1] 0.09177612
```

`dnorm()`: Finding the probability density

The function dnorm returns the value of the probability density function (pdf) of the normal distribution given a certain random variable x, a population mean μ and population standard deviation σ.

We know that in a standard normal distribution (mean=0, sd=1), the most common/likely value is zero, so let's see what the density is for 0 in a standard normal distribution:

```r
dnorm(x=0, mean=0, sd=1)
```

```
## [1] 0.3989423
```

```r
#Remember: by default, R uses mean=0 and sd=1, so:
dnorm(0)
```

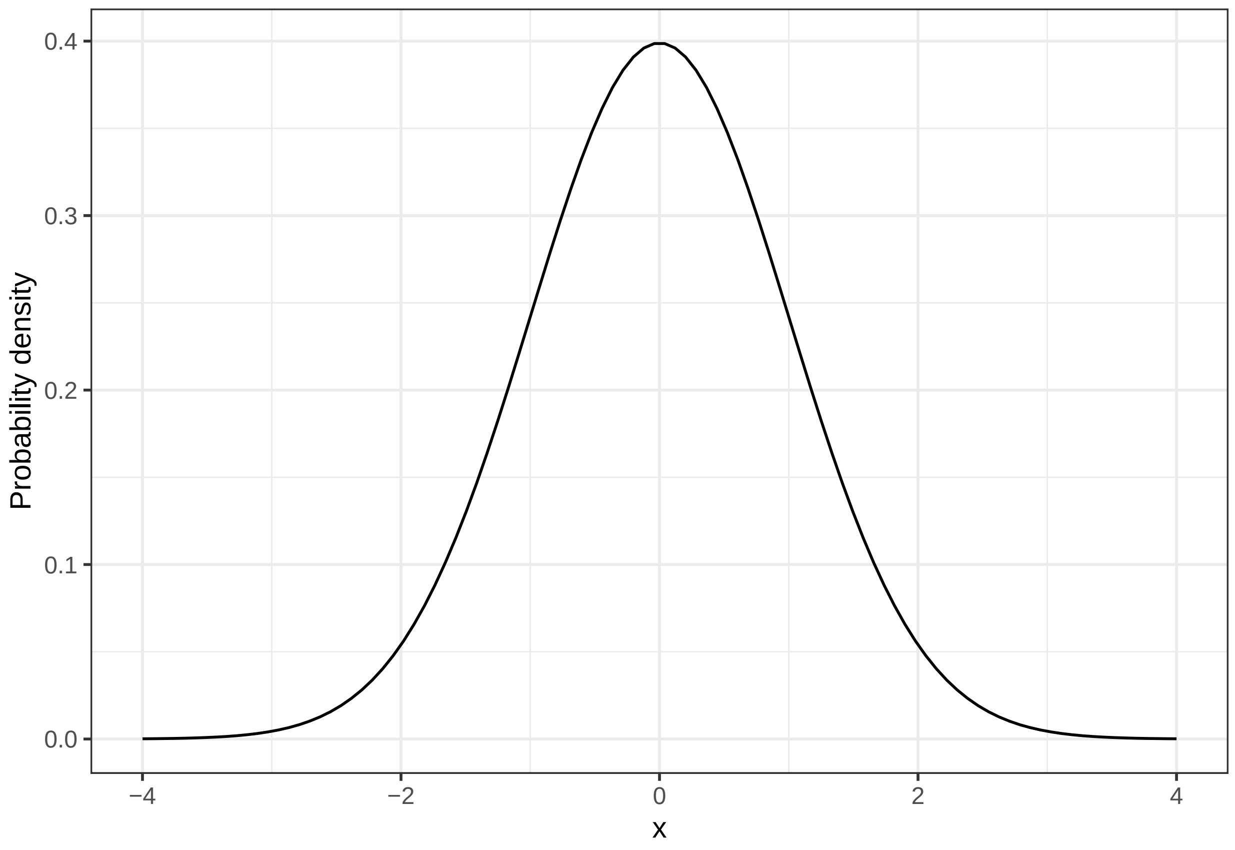
```
## [1] 0.3989423
```

```r
#gives you the same result
```

- `dnorm()` calculates the probability density of an observation from a normal distribution by plugging and chugging through the scary equation:

Typically when you're trying to solve questions about probability using the normal distribution, you'll often use pnorm instead of dnorm. One useful application of dnorm, however, is in creating a normal distribution plot in R. The following code illustrates how to do so:

```r
#Create a sequence of 100 equally spaced numbers between -4 and 4
x <- seq(-4, 4, length=100)

#create a vector of values that shows the height of the probability distribution
#for each value in x
y <- dnorm(x)

#create data.frame

df<-as.data.frame(cbind(x,y))
ggplot(df,aes(x=x, y=y)) +
        geom_line() + #plot line
        theme_bw() + #make it prettier
        ylab("Probability density") #fix y axis label
```



`qnorm()`: Finding the z-score of a quantile

The function qnorm returns the value of the inverse cumulative density function (cdf) of the normal distribution given a certain random variable p, a population mean μ and population standard deviation σ. The syntax for using qnorm is as follows: `qnorm(p, mean, sd)`

Put simply, you can use qnorm to find out what the Z-score is of the pth quantile of the normal distribution.

E.g.

```
#find the Z-score of the 99th quantile of the standard normal distribution
qnorm(.99, mean=0, sd=1)
```

```
## [1] 2.326348
```

```
# [1] 2.326348

#by default, R uses mean=0 and sd=1
qnorm(.99)
```

```
## [1] 2.326348
```

```
# [1] 2.326348

#find the Z-score of the 95th quantile of the standard normal distribution
qnorm(.95)
```

```
## [1] 1.644854
```

```r
# [1] 1.644854

#find the Z-score of the 10th quantile of the standard normal distribution
qnorm(.10)
```

```
## [1] -1.281552
```

```r
# [1] -1.281552
```

Practicing the basics!

Ex.1.3) The gestation period for cats has an approximate mean of 64 days and a standard deviation of 3 days, and the distribution of the gestation period is approximately Normal. What gestation period best corresponds to the 25th percentile?

```r
#your code
qnorm(.25, mean=64, sd=3)
```

```
## [1] 61.97653
```

Ex1.4) The gestation period for cats has an approximate mean of 64 days and a standard deviation of 3 days, and the distribution of the gestation period is approximately Normal. What gestation period corresponds to the top 10% of gestation periods? Round to the nearest tenth of a day.

```r
#your code
qnorm(.90, mean=64, sd=3)
```

```
## [1] 67.84465
```

Ex.1.5) The gestation period for cats has an approximate mean of 64 days and a standard deviation of 3 days, and the distribution of the gestation period is approximately Normal. What proportion of kittens have a gestation period longer than 62 days? Round your answer to two decimal places.

```r
#your code
```

Ex.1.6) The gestation period for cats has an approximate mean of 64 days and a standard deviation of 3 days, and the distribution of the gestation period is approximately Normal. What proportion of kittens have a gestation period between 62 days and 70 days? Round to two decimal places.

```r
#your code
#hint: first find the probability the proportion > 62 days
#then subtract the proportion > 70 days
```

Ex.1.7) An 1868 paper by German physician Carl Wunderlich reported, based on over a million body temperature readings, that healthy-adult body temperatures are approximately Normal with mean $\mu = 98.6$ degrees Fahrenheit (F) and standard deviation $\sigma = 0.6$ F . This is still the most widely quoted result for human temperature.

According to this study, what is the range of body temperatures that can be found in 95% of healthy adults? (We are looking for the middle 95% of the adult population.)

```
#your code
#first the upper range
#Upper range: 1-(alpha/2)=0.975
qnorm(p=0.975,mean=98.7, sd=0.6)
```

```
## [1] 99.87598
```

```
#then the lower range
#Lower range=alpha/2=0.025
qnorm(p=0.025,mean=98.7, sd=0.6)
```

```
## [1] 97.52402
```

```
#ANSWER: [97.52, 99.87]
```

---

2. Looking for normality in data:

This lab mainly focuses on some exercises to better understand the nature of the normal distribution. We will also learn a couple of tools that help us decide whether a particular data set is likely to have come from population with an approximately normal distribution.

Many statistical tests assume that the variable being analyzed has a normal distribution. Fortunately, many of these tests are fairly robust to this assumption—that is, they work reasonably well even when this assumption is not quite true, especially when sample size is large. Therefore it is often sufficient to be able to assess whether the data come from a distribution whose shape is even approximately normal (the bell curve).

Start by drawing a histogram

A good way to start is to simply visualize the frequency distribution of the variable in the data set by drawing a histogram. Let's use the age of passengers on the Titanic for our example.

```
#read in titanic
titanicData <- read.csv("input_files/titanic.csv", stringsAsFactors = TRUE)
#have a look at the data
head(titanicData)
```

```
##   passenger_class                              name     age    embarked
## 1             1st       Allen,MissElisabethWalton 29.0000 Southampton
## 2             1st        Allison,MissHelenLoraine  2.0000 Southampton
## 3             1st           Allison,MrHudsonJoshuaCreighton 30.0000 Southampton
## 4             1st Allison,MrsHudsonJ.C.(BessieWaldoDaniels) 25.0000 Southampton
## 5             1st                Allison,MasterHudsonTrevor  0.9167 Southampton
## 6             1st                          Anderson,MrHarry 47.0000 Southampton
##              home_destination    sex survive
## 1                  StLouis,MO female     yes
## 2 Montreal,PQ/Chesterville,ON female      no
## 3 Montreal,PQ/Chesterville,ON   male      no
## 4 Montreal,PQ/Chesterville,ON female      no
## 5 Montreal,PQ/Chesterville,ON   male     yes
## 6                  NewYork,NY   male     yes
```

```r
str(titanicData)
```

```
## 'data.frame':    1313 obs. of  7 variables:
##  $ passenger_class : Factor w/ 3 levels "1st","2nd","3rd": 1 1 1 1 1 1 1 1 1 1 ...
##  $ name            : Factor w/ 1310 levels "Abbing,MrAnthony",..: 22 25 26 27 24 31 45 46 50 54 ...
##  $ age             : num  29 2 30 25 0.917 ...
##  $ embarked        : Factor w/ 4 levels "","Cherbourg",..: 4 4 4 4 4 4 4 4 4 2 ...
##  $ home_destination: Factor w/ 372 levels "","?Havana,Cuba",..: 314 234 234 234 234 241 165 26 24 23
##  $ sex             : Factor w/ 2 levels "female","male": 1 1 2 1 2 2 1 2 1 2 ...
##  $ survive         : Factor w/ 2 levels "no","yes": 2 1 1 1 2 2 2 1 2 1 ...
```

Let's first check if we have NAs in the age column and, if so, remove them:

```r
#about half of the rows have missing values for age.
table(is.na(titanicData$age))
```

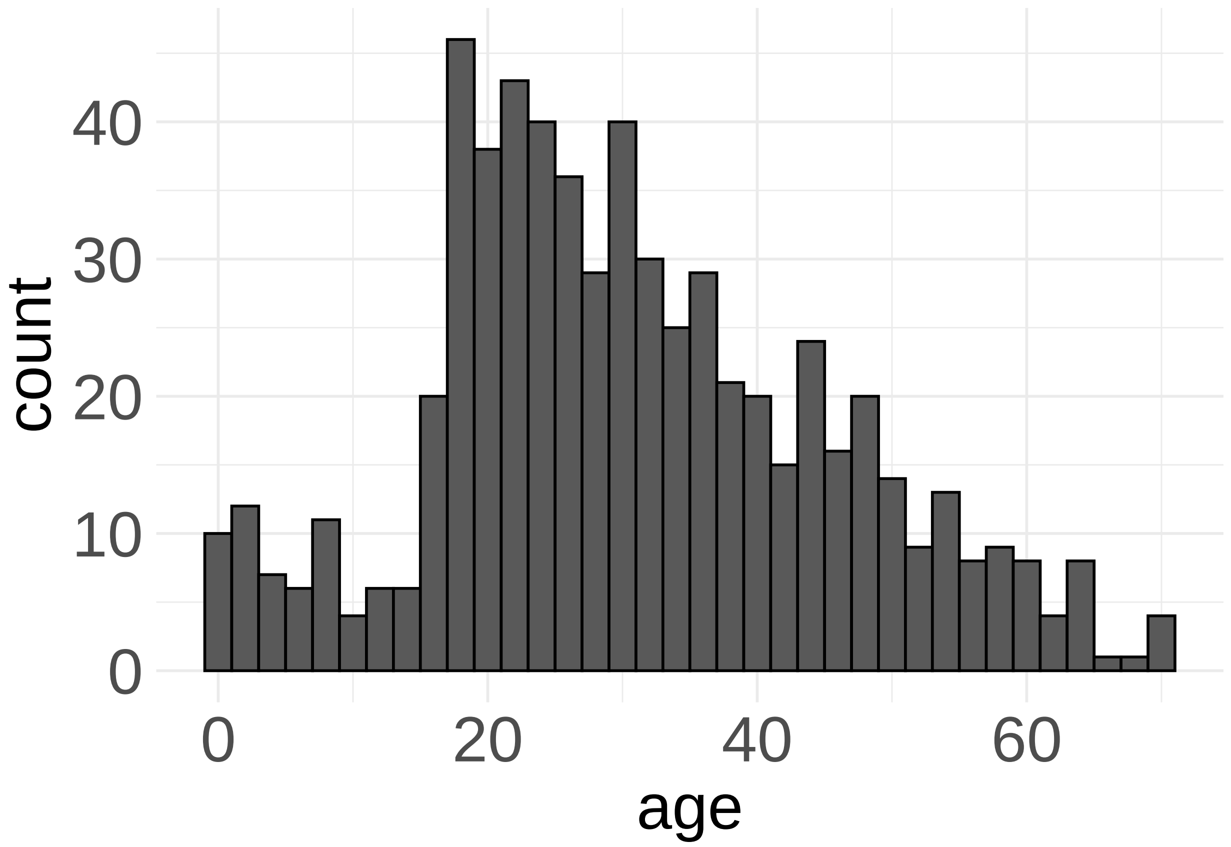
```
##
## FALSE  TRUE
##   633   680
```

About half of the rows have missing values for age. let's get rid of them.

```r
#about half of the rows have missing values for age.
new_titanic_data<-na.omit(titanicData)
```

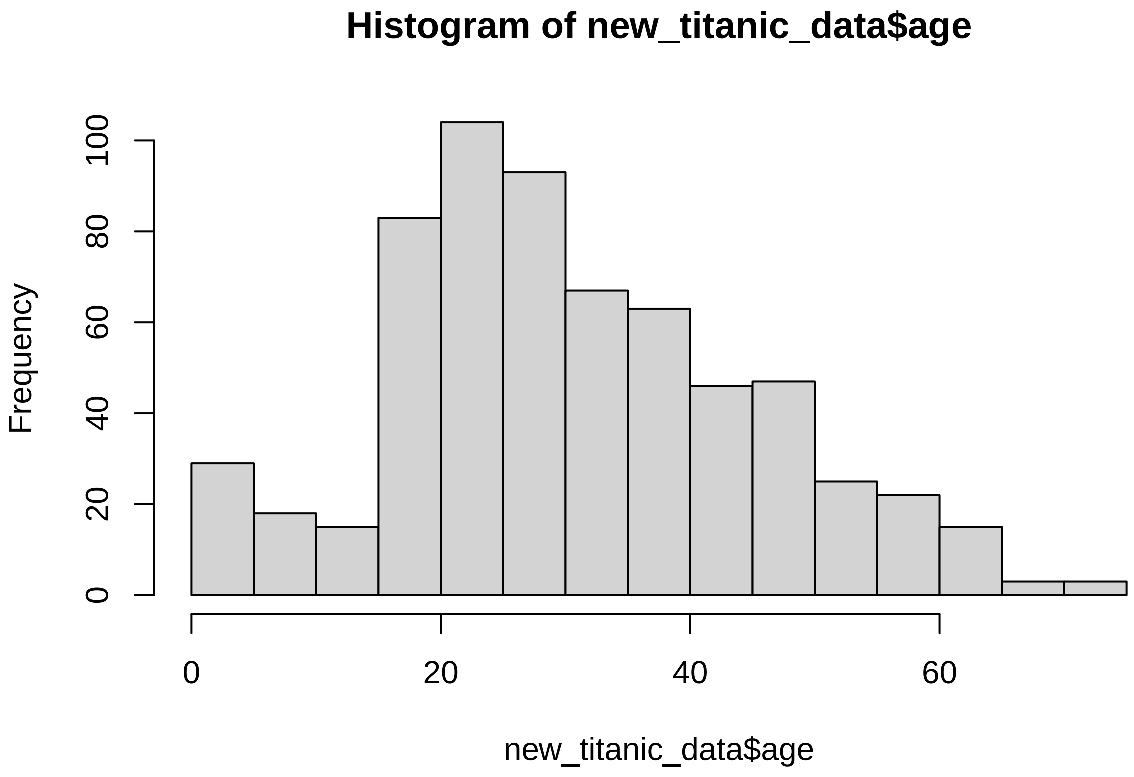
Remember we can use `ggplot()` to draw histograms.

```r
#sneaky trick: at the top of my scripts, I define the object I call bb_theme which has
↪  the ggplot theme options I like to use. That way I don't have to type them all every
↪  time. Feel free to use:

bb_theme <-  theme_minimal()+
        theme(axis.text.x=element_text(size=24),
              axis.text.y=element_text(size=24),
              axis.title.x = element_text(size=24),
              axis.title.y = element_text(size=24),
              legend.title = element_text(size=22),
              legend.text = element_text(size=22),
              plot.title = element_text(size=24),
              strip.text =  element_text(size=22))
#use ggplot to make a histogram of age
ggplot(new_titanic_data, aes(x = age)) +
  geom_histogram(binwidth = 2,color="black") + #bin size=2, add black line around bars
  bb_theme #here I am using my custom theme
```



If we are just drawing a histogram for ourselves to better understand the data, it is even easier to just use a function from base R, `hist()`. Give `hist()` a vector of data as input, and it will print a histogram in the plots window.

```
hist(new_titanic_data$age)
```



Looking at this histogram, we see that the frequency distribution of the variable is not exactly normal; it is slightly asymmetric and there seems to be a second mode near 0.

- Is this a big deal? Is this deviation surprising?
- To find out we need to get a sense of the variability we expect from a normal distribution.
- Note that, like the normal distribution, the frequency distribution has a large mode near the center of the distribution, frequencies mainly fall off to either side, and there are no outliers. This is close enough to normal that most methods would work fine.

QQ plot: a better way

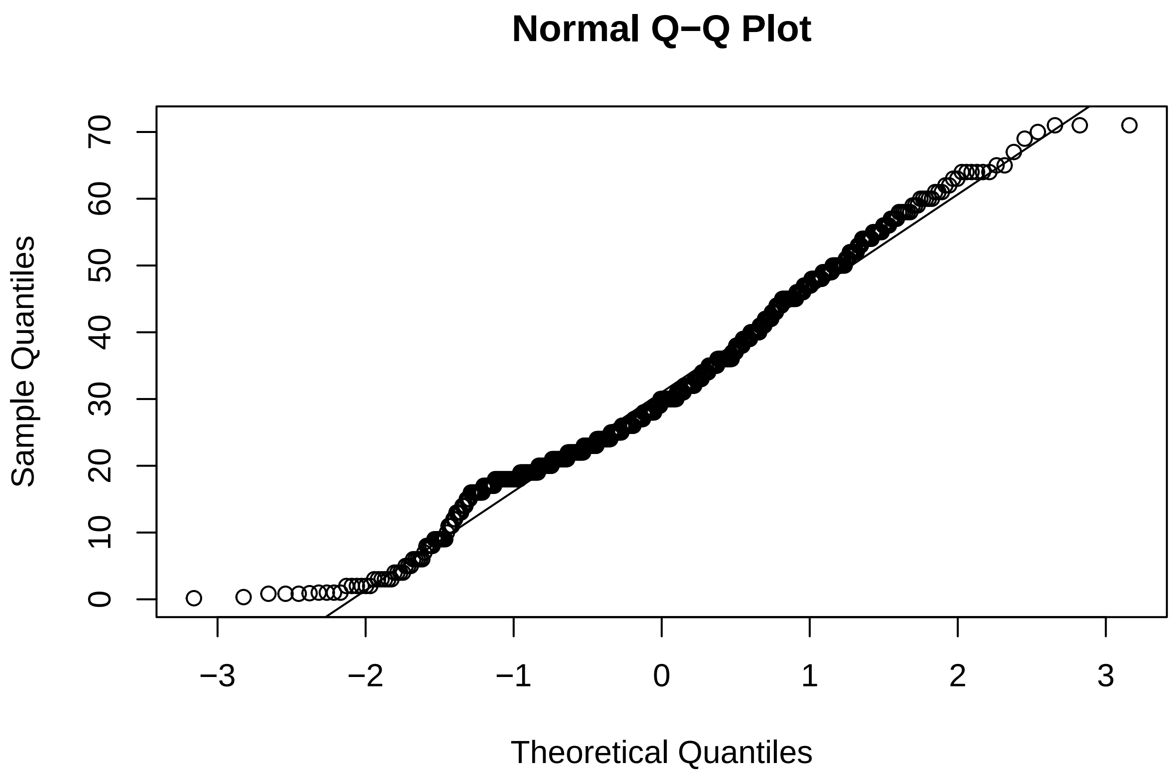
I am always surprised about how easily I can convince myself that a sample does not come from a normal distribution.

- While there are statistical procedures to test the null hypothesis that data come from a normal distribution, we almost never use these because a deviation from a normal distribution can be most important when we have the least power to detect it.
- For that reason,we usually use **our eyes** (believe it or not), rather than null hypothesis significance testing to see if data are approximately normal.
- Apart from a histogram (which we tend to overinterpret as not being "normal enough"), another graphical technique that can help us visualize whether a variable is approximately normal is called a quantile plot (or a **QQ plot**).

The **QQ plot** shows the data on the vertical axis ranked in order from smallest to largest ("sample quantiles" in the figure below). On the horizontal axis, it shows the expected value of an individual with the same quantile if the distribution were normal ("theoretical quantiles" in the same figure). The QQ plot should follow more or less along a straight line if the data come from a normal distribution (with some tolerance for sampling variation).

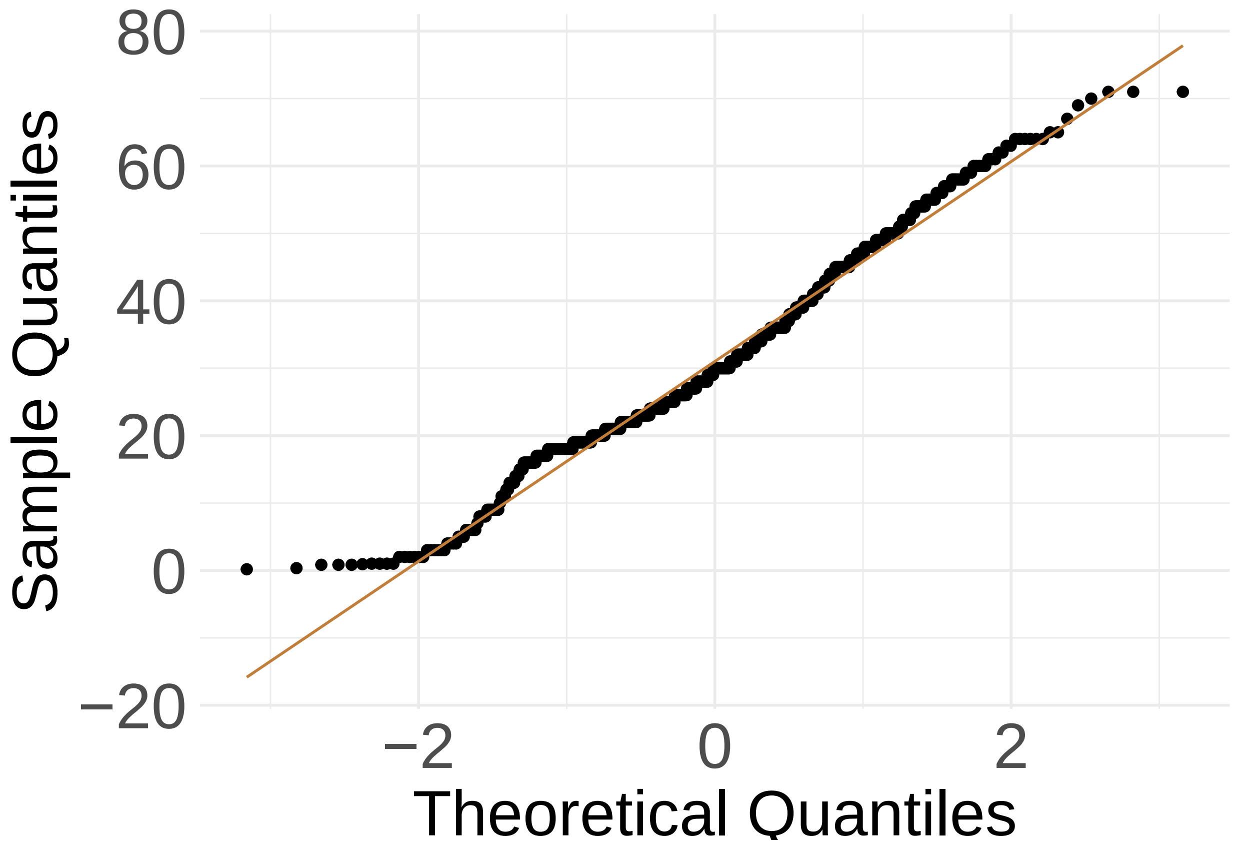
QQ plots can be made in R using a function called `qqnorm()`. Simply give the vector of data as input and it will draw a QQ plot for you. (`qqline()` will draw a line through that Q-Q plot to make the linear relationship easier to see.)

```
#using base R
qqnorm(new_titanic_data$age)
qqline(new_titanic_data$age)
```



Or the `ggplot` way:

```
ggplot(new_titanic_data, aes(sample=age)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```

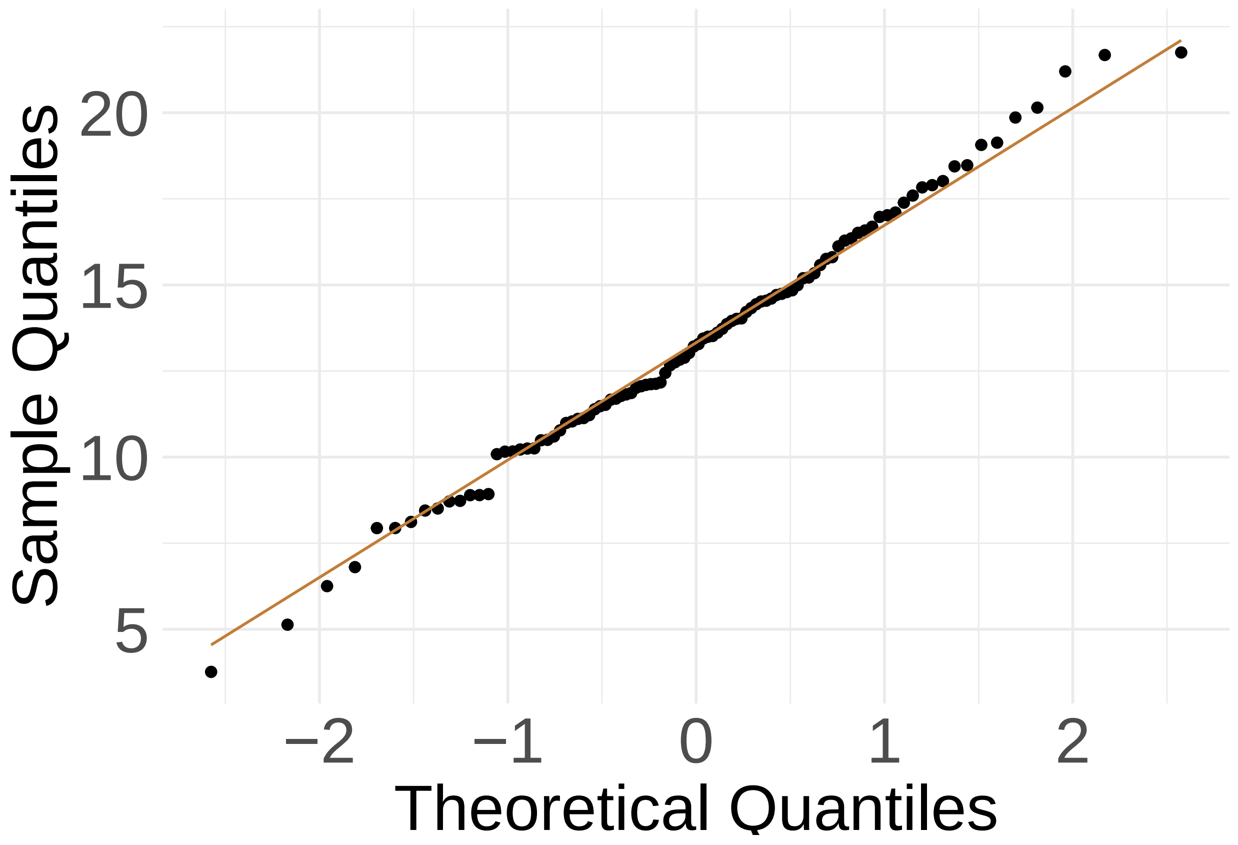


This is what the resulting graph looks like for the Titanic age data. The dots do not land along a perfectly straight line. In particular the graph curves at the upper and lower end. However, this distribution definitely would be close enough to normal to use most standard methods, such as the t-test.

It is difficult to interpret QQ plots without experience. One of the goals of today's exercises will be to develop some visual experience about what these graphs look like when the data is truly normal. To do that, we will take advantage of a function built into R to generate random numbers drawn from a normal distribution. This function is called `rnorm()`.

Let's look at a QQ plot generated from 100 numbers randomly drawn from a normal distribution:

```
#create normal_vector with 100 data points randomly sampled from a normal distribution
↪  with mean 13 and standard deviation 4
normal_vec <- rnorm(n = 100, mean = 13, sd = 4)
#using base R or ggplot, whatever you prefer, plot the qqplot and then the qqline
#I will use ggplot, so let's turn this vector into a data.frame
normal_dataframe<-as.data.frame(x=normal_vec)
colnames(normal_dataframe)<-"x" #change column name
ggplot(normal_dataframe, aes(sample=x)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```



These points fall mainly along a straight line, but there is some wobble around that line even though these points were in fact randomly sampled from a known normal distribution. With a QQ plot, we are looking for an overall pattern that is approximately a straight line, but **we do not expect a perfect line**.

In the exercises, we'll simulate several samples from a normal distribution to try to build intuition about the kinds of results you might get.

When data are not normally distributed, the dots in the quantile plot will not follow a straight line, even approximately. For example, here is a histogram and a QQ plot for the population size of various counties, from the data in countries.csv. These data are very skewed to the right, and do not follow a normal distribution at all.

QQ-practice!

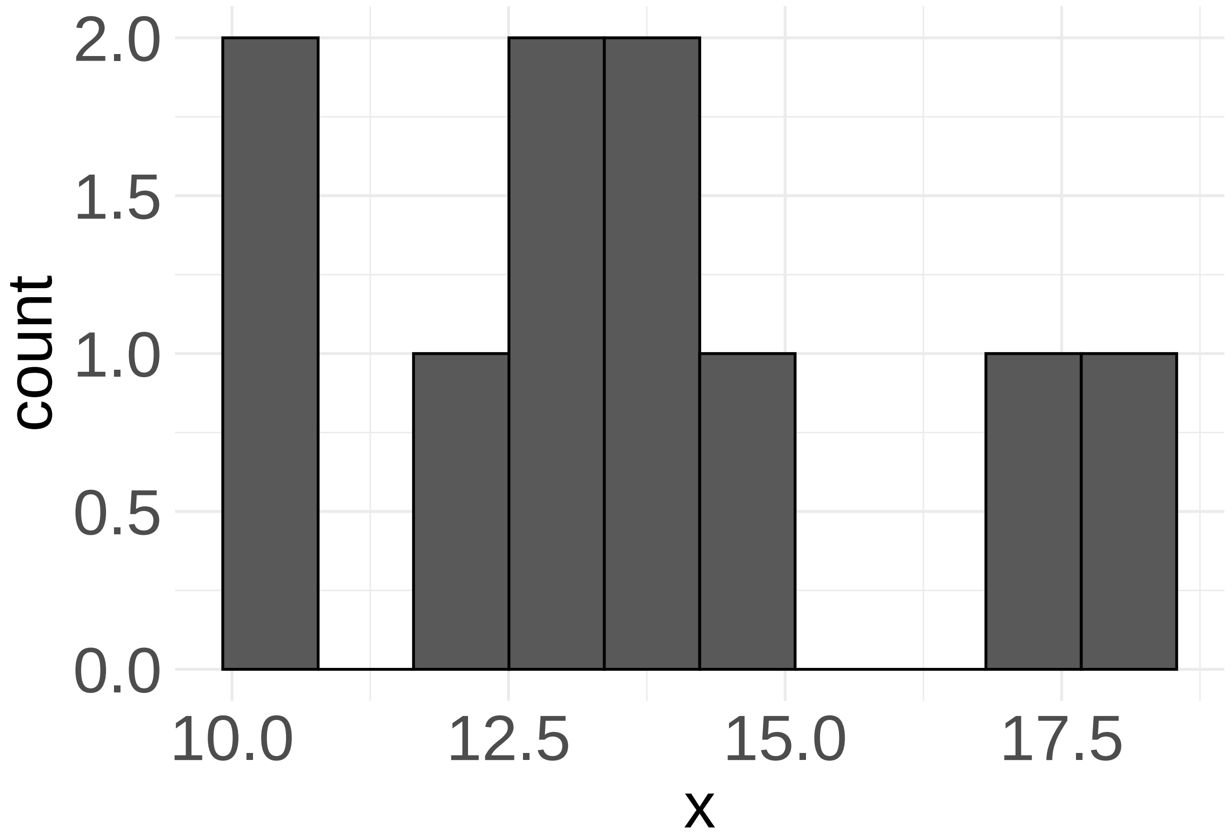
Ex.2.1) Let's use R's random number generator for the normal distribution to build intuition for how to view and interpret histograms and QQ plots. Remember, the lists of values generated by `rnorm()` come from a population that truly has a normal distribution.

A) Generate a list of 10 random numbers from a normal distribution with mean 15 and standard deviation 3 and save the results to `normal_vector`:

```r
#your code
normal_vector<-rnorm(n=10, mean=15, sd=3)
```

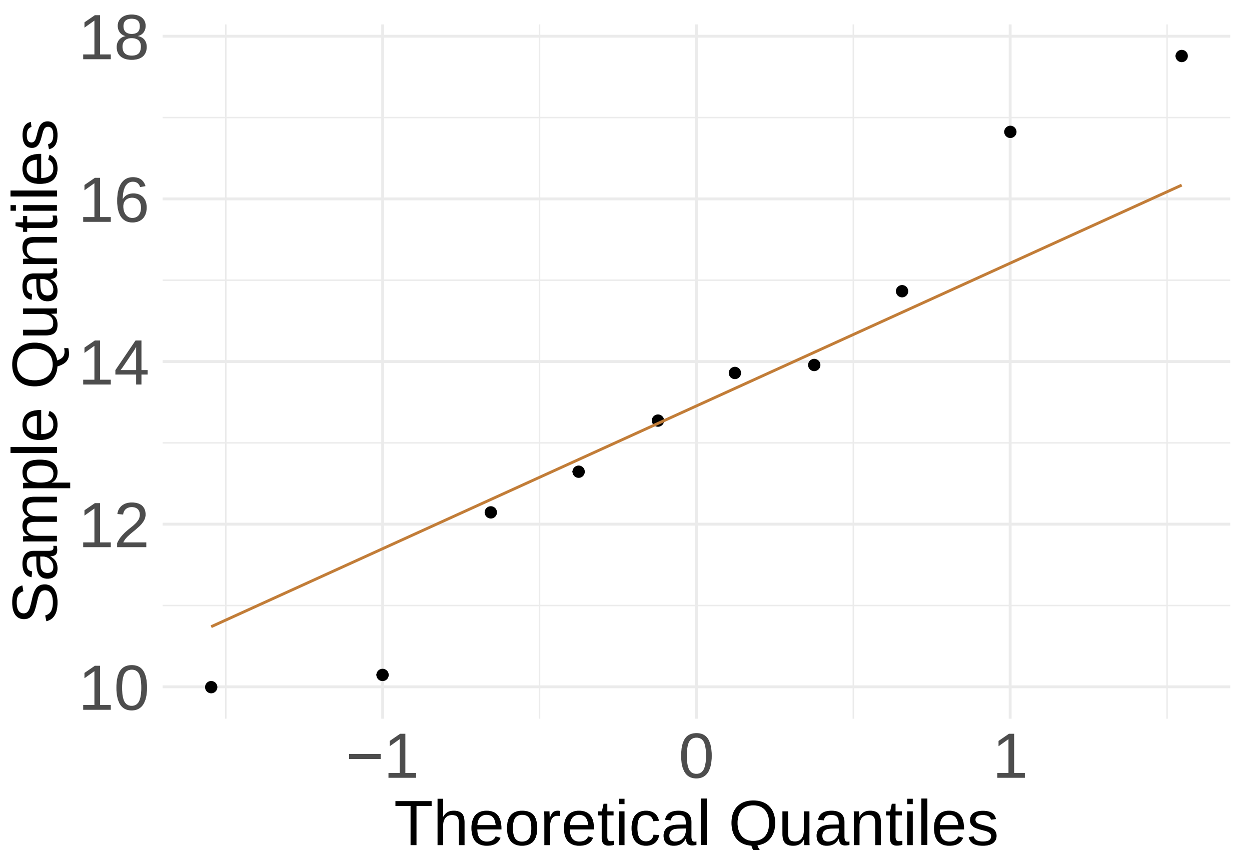
B) Plot a histogram of these numbers from part a.

```r
#your code
#make data.frame for ggplot
normal_df<-as.data.frame(x=normal_vector)
colnames(normal_df)<-"x" #change column name
#plot
ggplot(normal_df, aes(x = x)) +
  geom_histogram(bins=10,color="black") + #add black line around bars
  bb_theme #here I am using my custom theme
```



C) Plot a QQ plot from the numbers in part a.

```r
#your code

ggplot(normal_df, aes(sample=x)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```



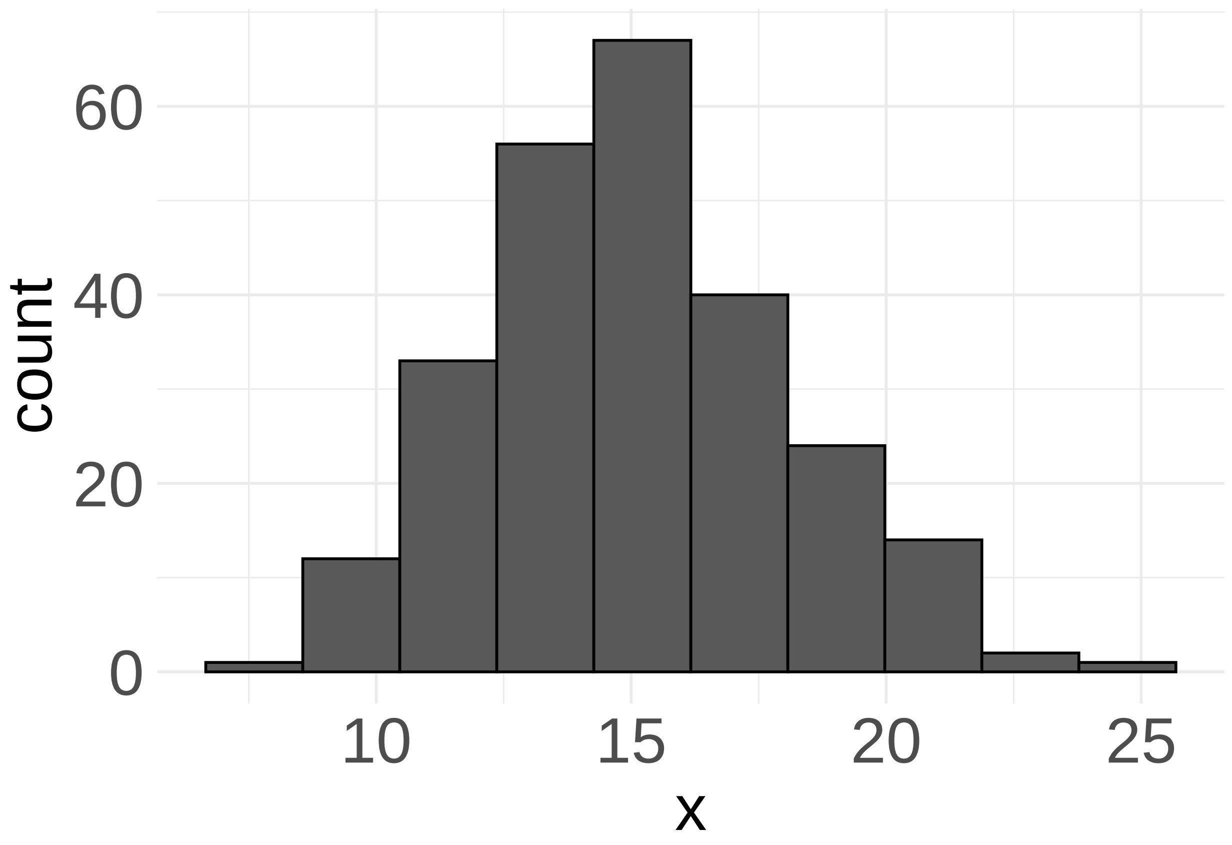
D) Repeat steps A) through C) several times (at least a dozen times). For each, look at the histograms and QQ plots. Think about the ways in which these look different from the expectation of a normal distribution (but remember that each of these samples comes from a truly normal population).

- Challenge: if you're feeling adventurous, you could try writing a for loop to do this!

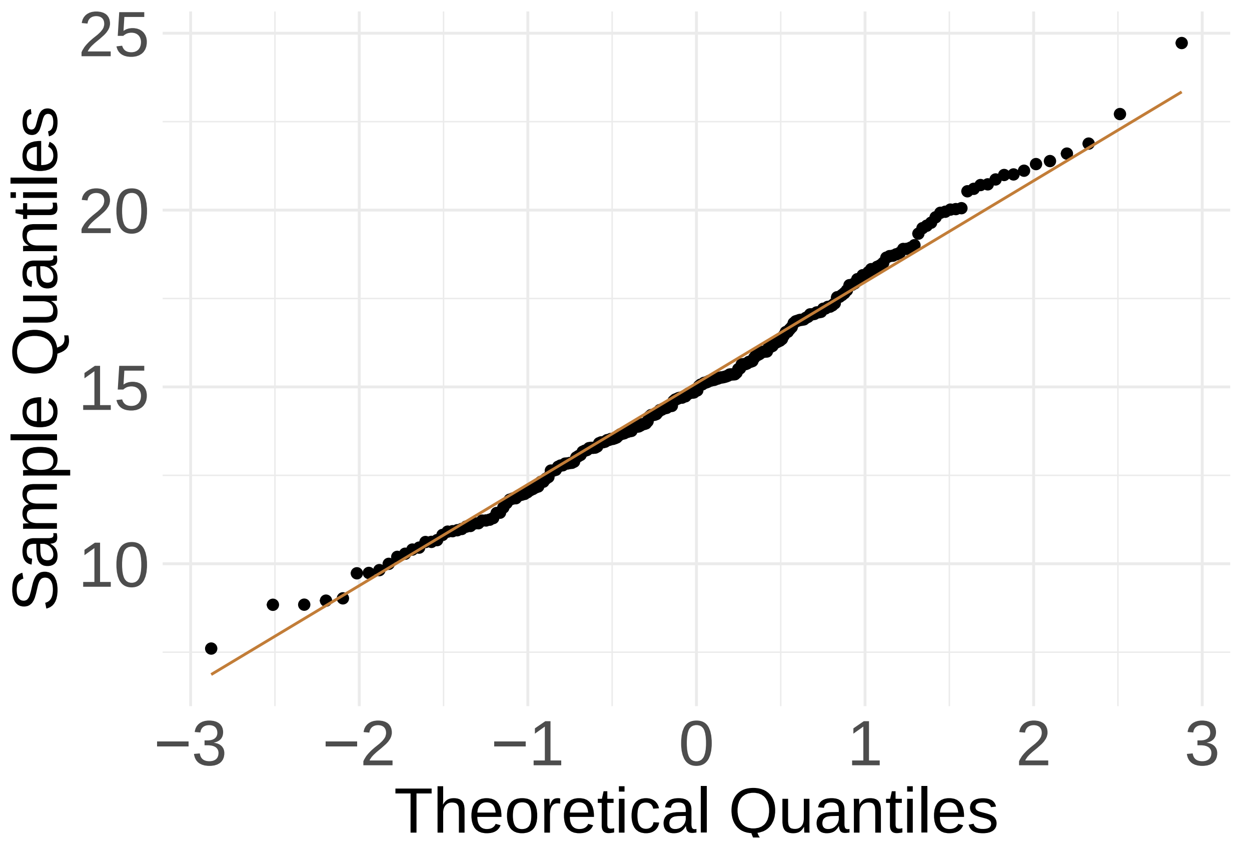
```
#your code (if you write any for this)
#I set it to eval=F because it's a lot of pltos to add to the PDF but feel free to change
↪  that.
#create empty list with 12 spaces
nreps=12
empty_list1<-vector('list', nreps)
empty_list2<-vector('list', nreps)
for(i in 1:nreps){ #for loop
        #step1: sample:
        normal_vector<-rnorm(n=10, mean=15, sd=3)
        normal_df<-as.data.frame(x=normal_vector)
        colnames(normal_df)<-"x" #change column name
        #step2: make histogram
        p1<-ggplot(normal_df, aes(x = x)) +
        geom_histogram(bins=10,color="black") + #add black line around bars
        bb_theme #here I am using my custom theme
        p1<-print(p1)
        #step3: make qqplot
        p2<-ggplot(normal_df, aes(sample=x)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
        p2<-print(p2)
        empty_list1[[i]]<-p1
        empty_list2[[i]]<-p2
}

#check each pair with
empty_list1[[1]]
empty_list2[[2]]
#etc
```

Ex2.2) Repeat the procedures of Question 1A-C, except this time have R sample 250 individuals for each sample. (You can use the same command as in Question 1, but now set n = 250.) Do the graphs and QQ plots from these larger samples look more like the normal expectations than the smaller sample you already did? Why do you think that this is?

```
#your code
#create normal_vector
normal_vector<-rnorm(n=250, mean=15, sd=3)
#plot histogram
normal_df<-as.data.frame(x=normal_vector)
colnames(normal_df)<-"x" #change column name
ggplot(normal_df, aes(x = x)) +
  geom_histogram(bins=10,color="black") + #add black line around bars
  bb_theme #here I am using my custom theme
```



```r
#plot qqplot
ggplot(normal_df, aes(sample=x)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```



```
#now repeat this a few times to get a sense of how much it varies from time to time.
```

Ex.2.3) The air quality data set from the Datasets package contains 4 measures of air quality over a year (1973) in New York: Ozone level, solar radiation, wind speed and temperature. You are going to test the normality of each of these measures. But before you load the data guess which measure you think will look the most and least normal.

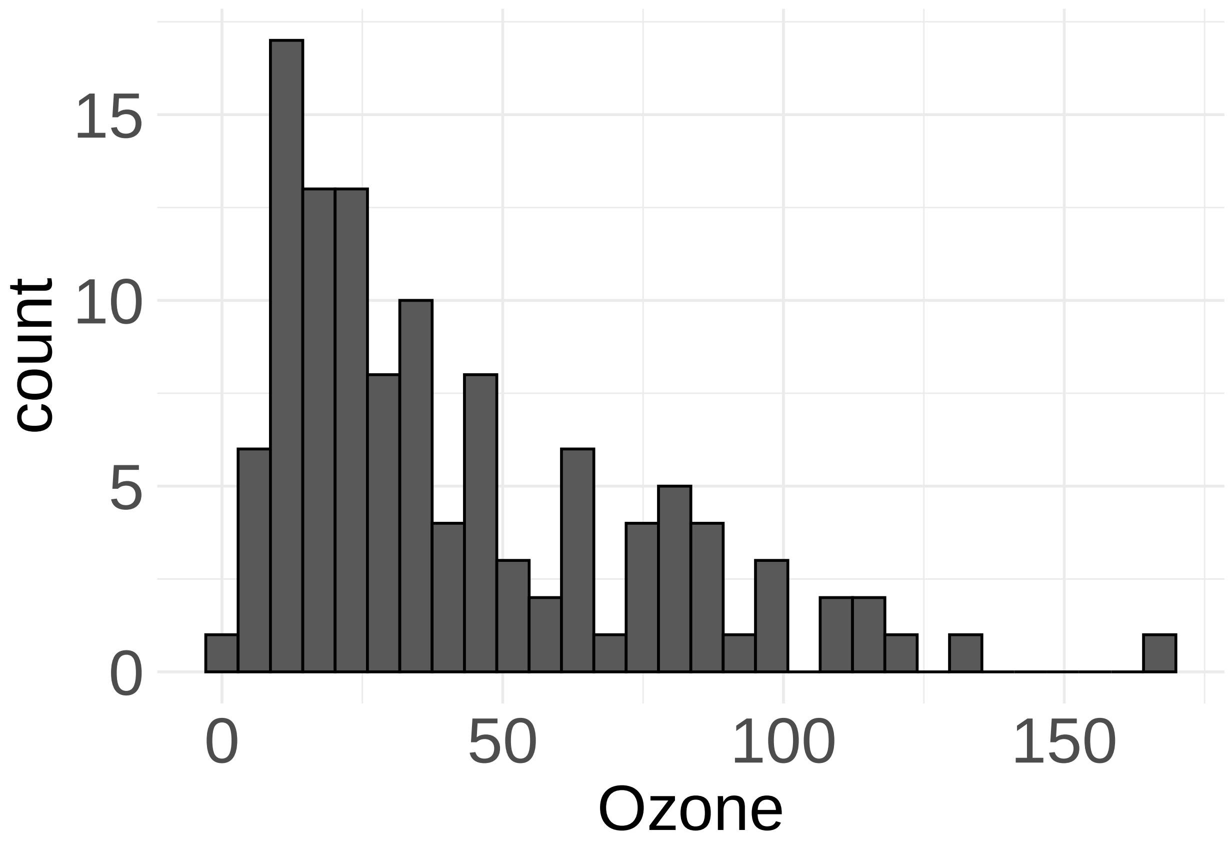
- Most normal:
- Least normal:

Load thn data and then make histograms and QQ plots for each of these variables!

```
#code here
#Load data from the datasets package
airquality<-datasets::airquality
#Ozone histogram and qqplot
ggplot(airquality, aes(x = Ozone)) +
  geom_histogram(color="black") + #add black line around bars
  bb_theme #here I am using my custom theme
```

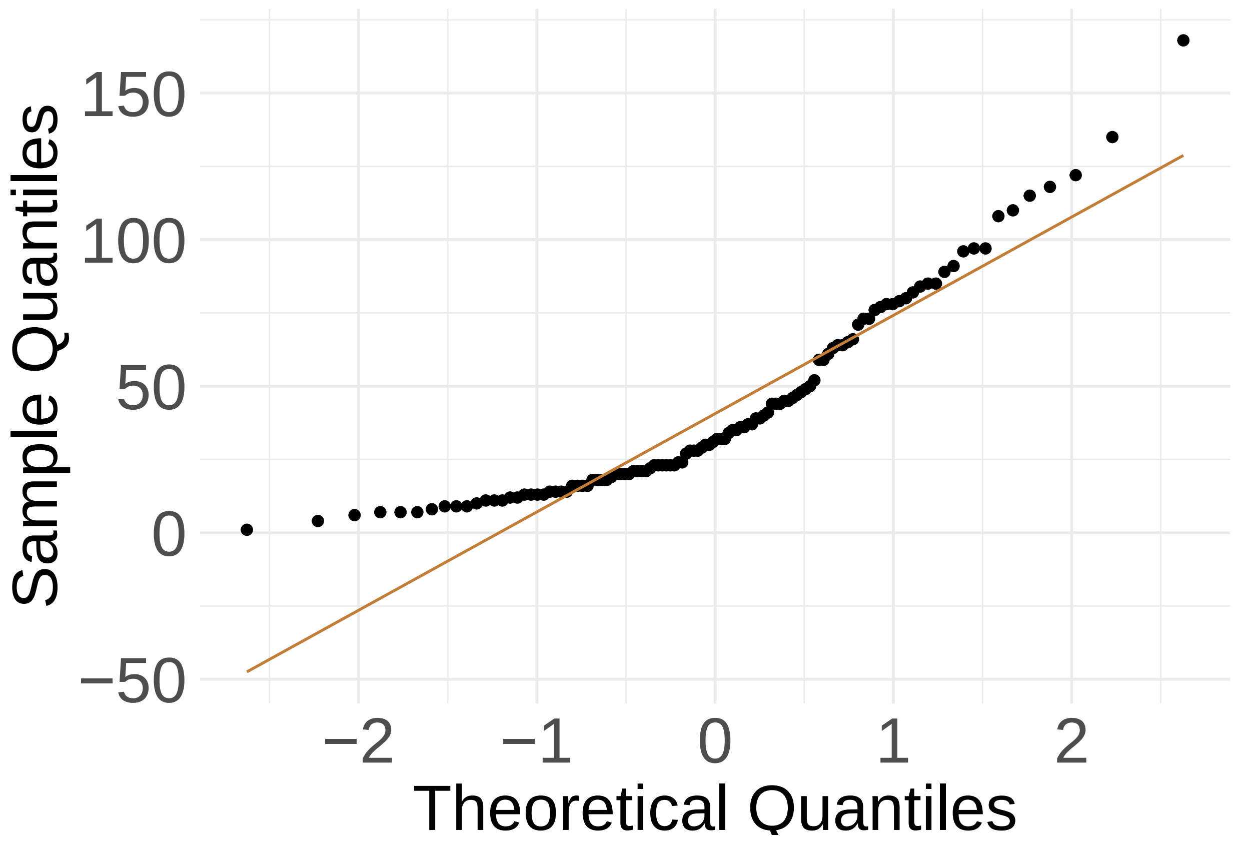
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 37 rows containing non-finite values (`stat_bin()`).
```



```r
#plot qqplot
ggplot(airquality, aes(sample=Ozone)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```

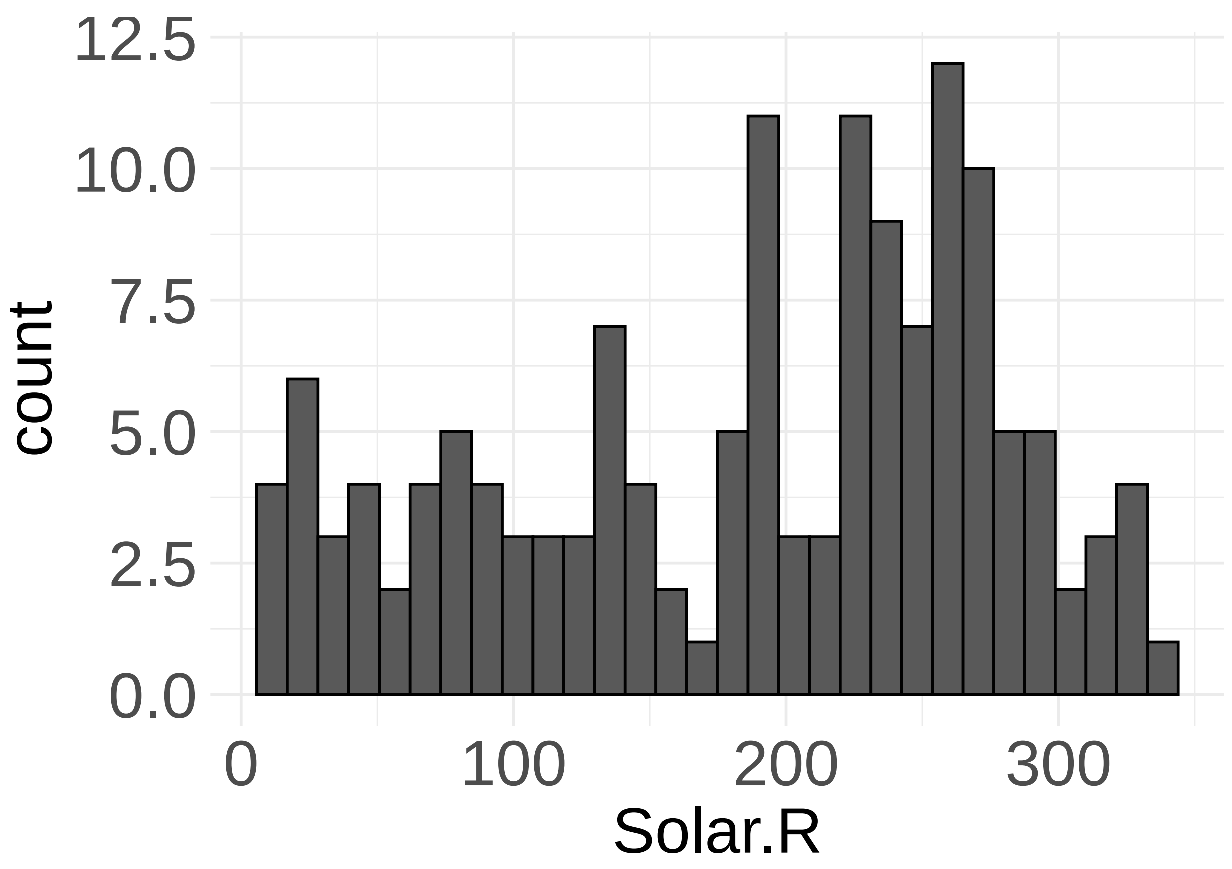
```
## Warning: Removed 37 rows containing non-finite values (`stat_qq()`).

## Warning: Removed 37 rows containing non-finite values (`stat_qq_line()`).
```



```r
#Solar.R histogram and qqplot
ggplot(airquality, aes(x = Solar.R)) +
  geom_histogram(color="black") + #add black line around bars
  bb_theme #here I am using my custom theme
```

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

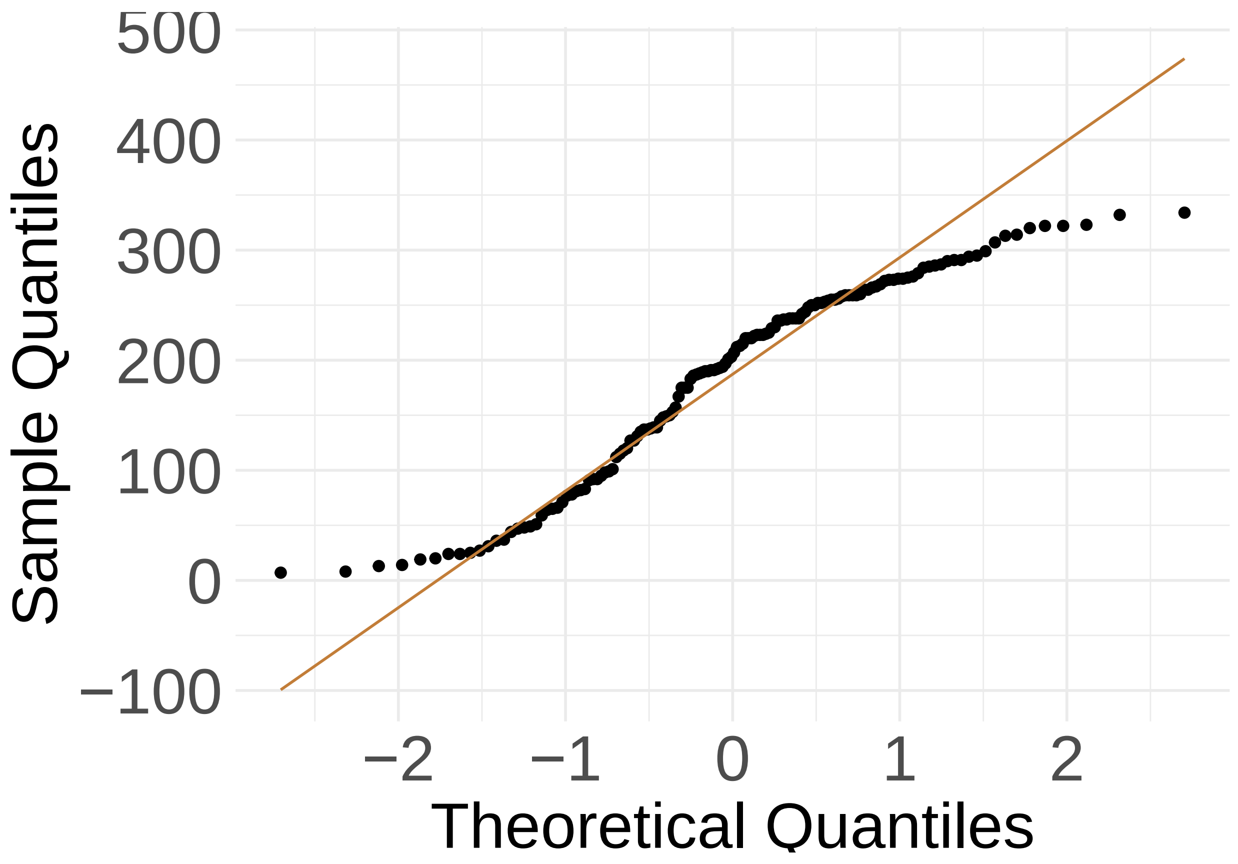
```
## Warning: Removed 7 rows containing non-finite values (`stat_bin()`).
```



```r
ggplot(airquality, aes(sample=Solar.R)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```

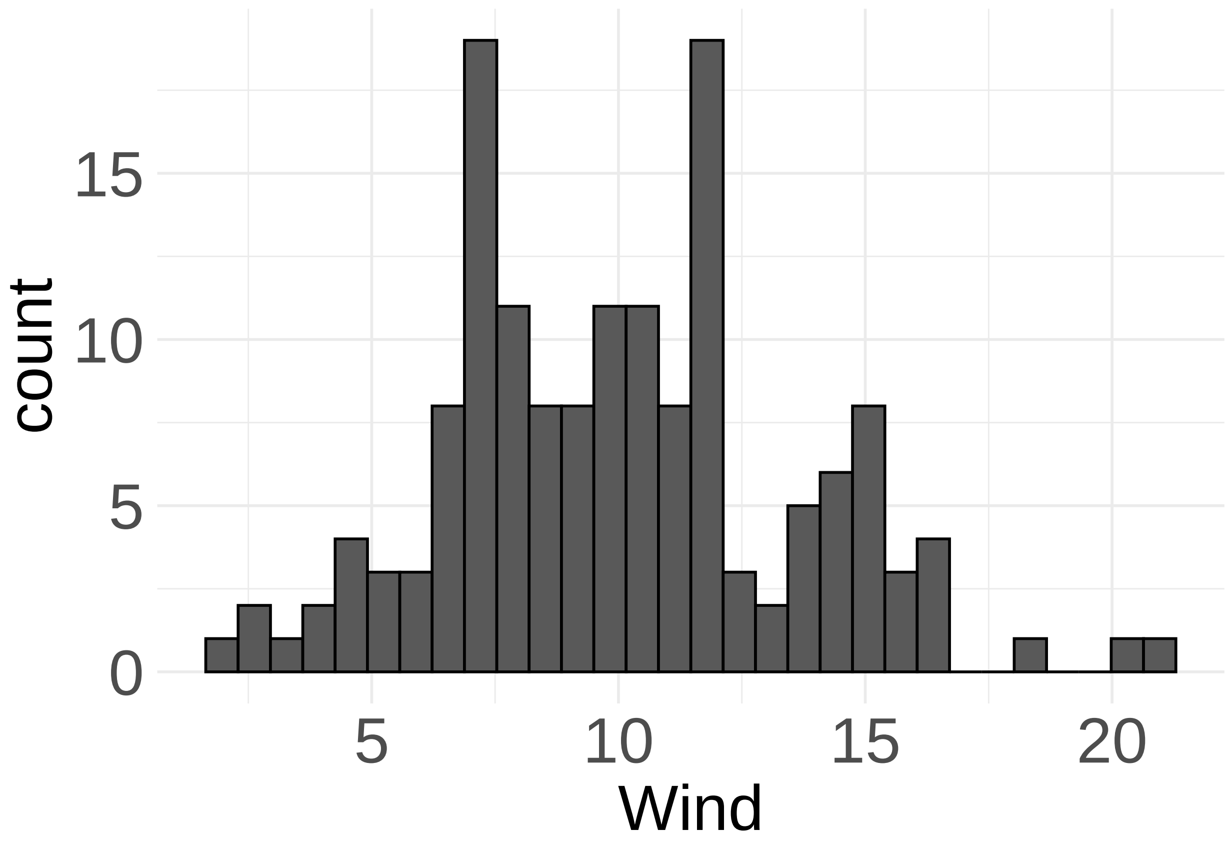
```
## Warning: Removed 7 rows containing non-finite values (`stat_qq()`).
```

```
## Warning: Removed 7 rows containing non-finite values (`stat_qq_line()`).
```

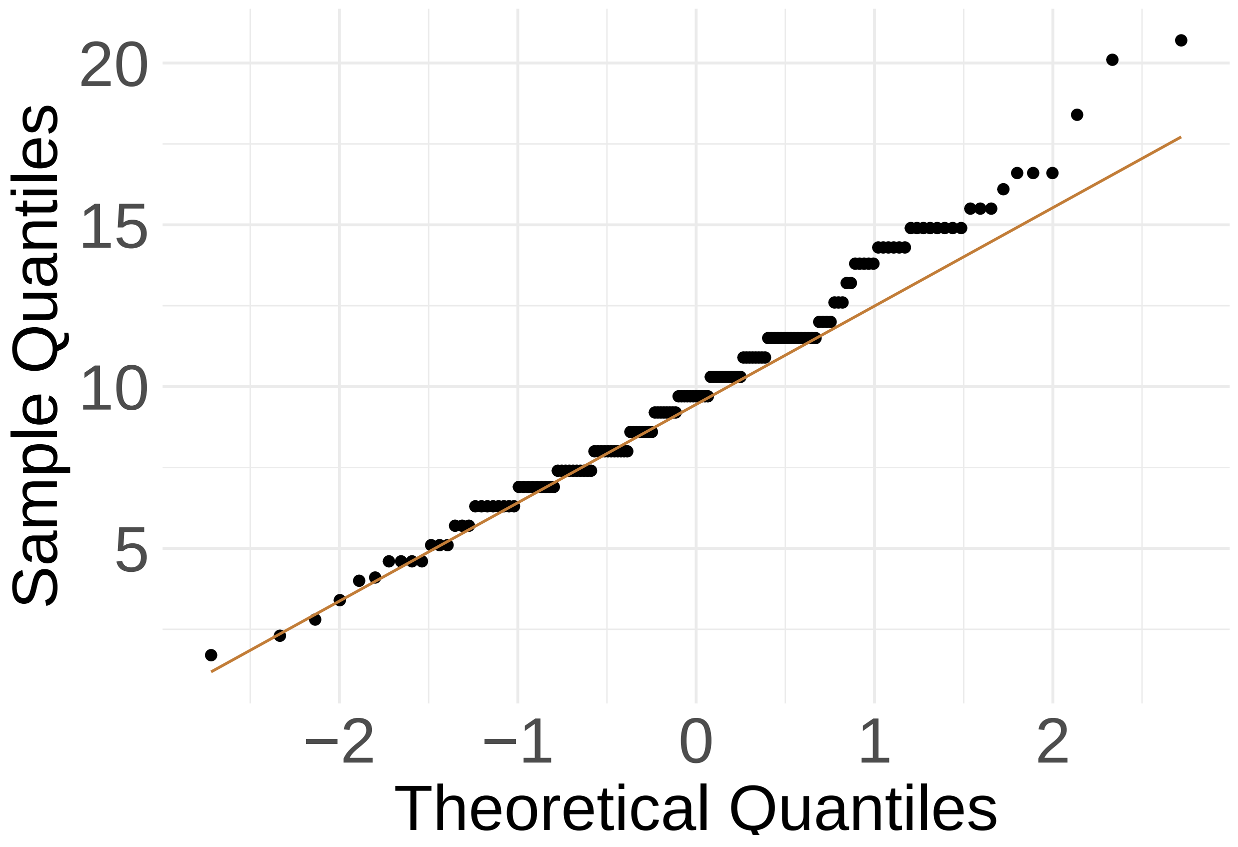


```
#Wind histogram and qqplot
ggplot(airquality, aes(x = Wind)) +
  geom_histogram(color="black") + #add black line around bars
  bb_theme #here I am using my custom theme
```

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

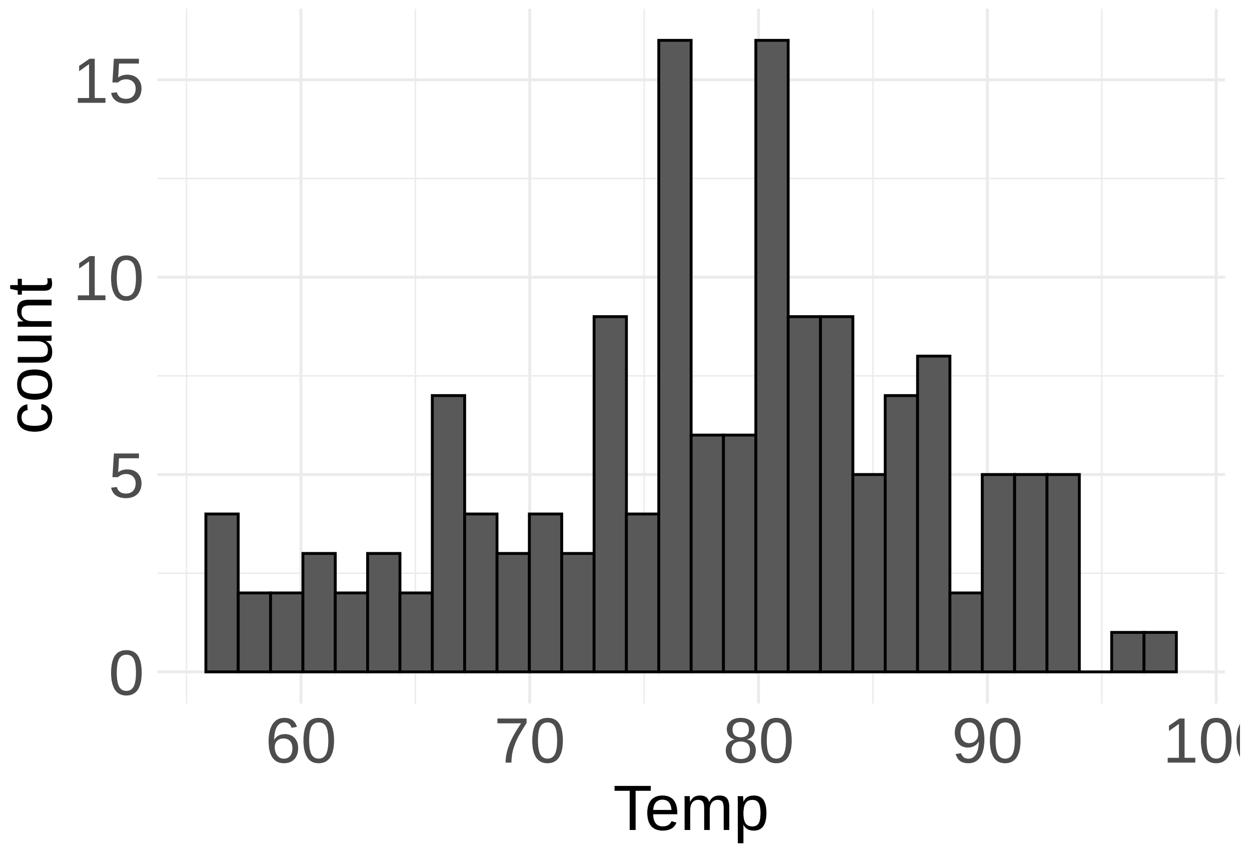


```r
ggplot(airquality, aes(sample=Wind)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```

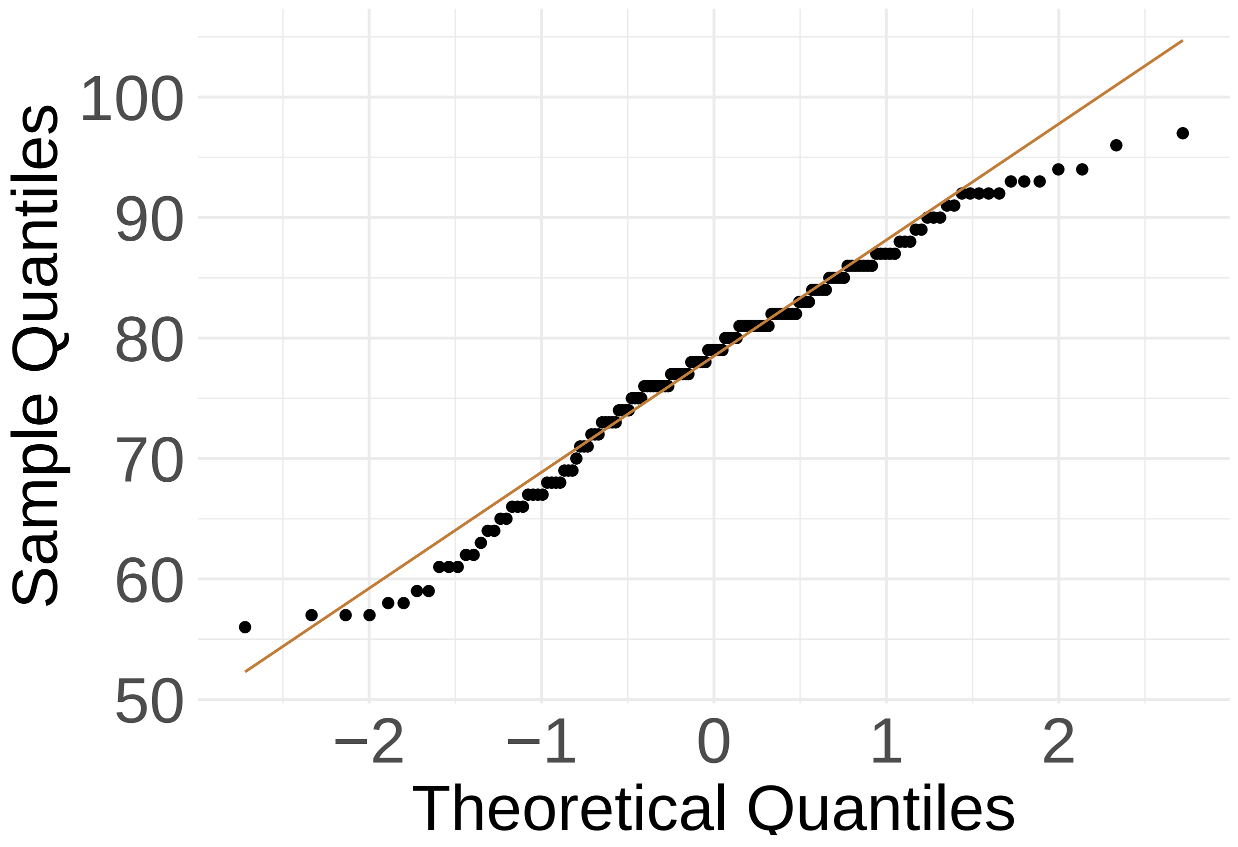


```r
#Temp
ggplot(airquality, aes(x = Temp)) +
  geom_histogram(color="black") + #add black line around bars
  bb_theme #here I am using my custom theme
```

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



```
ggplot(airquality, aes(sample=Temp)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```



Case Study 1: House sparrows

Ex.3.1) In 1898, Hermon Bumpus collected house sparrows (*Passer domesticus*) that had been caught in a severe winter storm in Chicago. He made several measurements on these sparrows, and his data are in the file `input_files/bumpus.csv`.

Bumpus used these data to observe differences between the birds that survived and those that died from the storm. This became one of the first direct and quantitative observations of natural selection on morphological traits. Here, let's use these data to practice looking for fit of the normal distribution.

A) Plot the distribution of total length (this is the length of the bird from beak to tail). Does the data look as though it comes from distribution that is approximately normal?

```
#read in input_files/bumpus.csv. Call it bumpus
bumpus<-readr::read_csv("input_files/bumpus.csv")
```

```
## Rows: 136 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): sex, age, survival
## dbl (10): bumpus_number, total_length_mm, alar_extent_mm, weight_g, length_b...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#have a look at the data
head(bumpus)
```

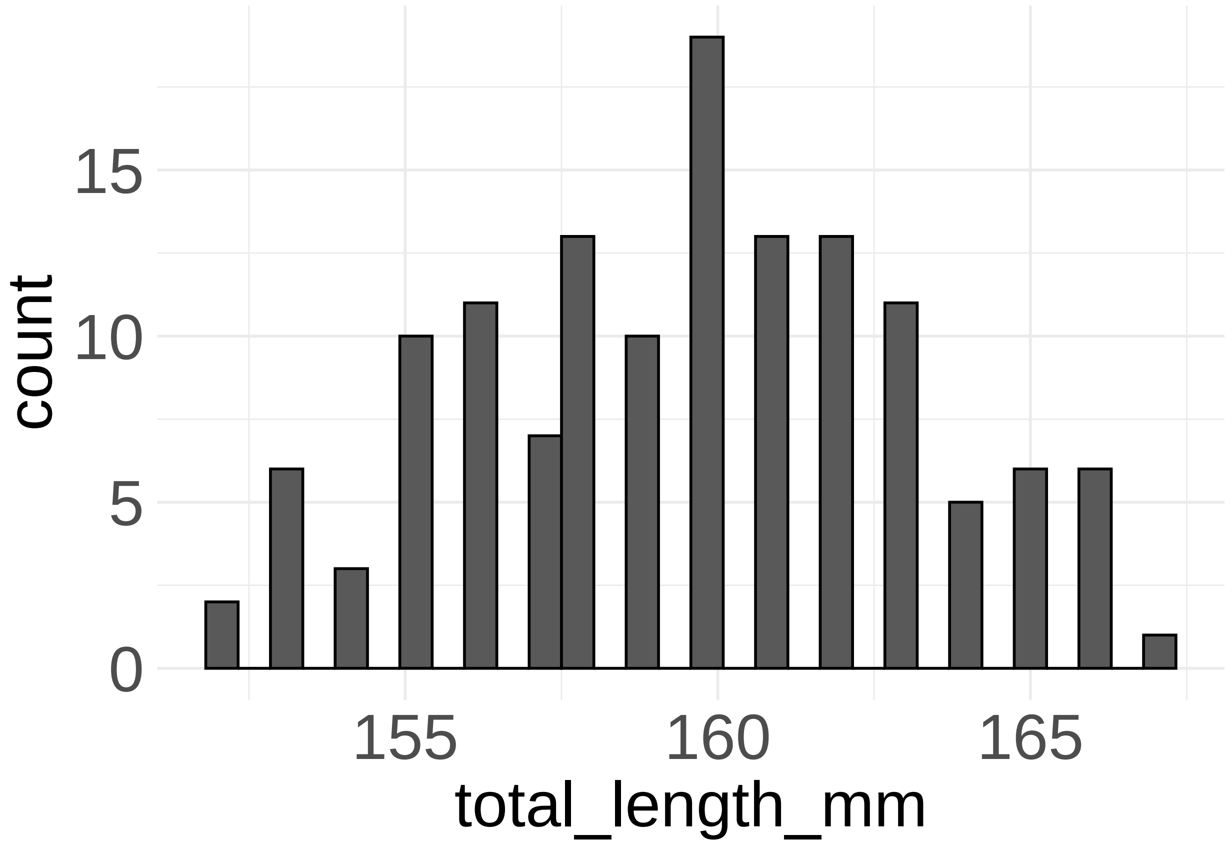
```
## # A tibble: 6 x 13
##   bumpus_number sex   age   survival total_length_mm alar_extent_mm weight_g
##           <dbl> <chr> <chr> <chr>              <dbl>          <dbl>    <dbl>
## 1             1 m     a     survived             154            241     24.5
## 2             1 m     a     died                 165            240     26.5
## 3             2 m     a     died                 160            245     26.1
## 4             2 m     a     survived             160            252     26.9
## 5             3 m     a     survived             155            243     26.9
## 6             3 m     a     died                 161            249     25.6
## # i 6 more variables: length_beak_head_mm <dbl>, length_humerus_in <dbl>,
## #   length_femur_in <dbl>, length_tibiotarsus_in <dbl>, skull_width_in_ <dbl>,
## #   keel_length_in <dbl>
```

```
#check the structure. Make sure total_length_mm is numeric.
str(bumpus)
```

```
## spc_tbl_ [136 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ bumpus_number        : num [1:136] 1 1 2 2 3 3 4 4 5 5 ...
##  $ sex                  : chr [1:136] "m" "m" "m" "m" ...
##  $ age                  : chr [1:136] "a" "a" "a" "a" ...
##  $ survival             : chr [1:136] "survived" "died" "died" "survived" ...
##  $ total_length_mm      : num [1:136] 154 165 160 160 155 161 154 162 156 163 ...
##  $ alar_extent_mm       : num [1:136] 241 240 245 252 243 249 245 246 247 250 ...
##  $ weight_g             : num [1:136] 24.5 26.5 26.1 26.9 26.9 25.6 24.3 25.9 24.1 25.5 ...
##  $ length_beak_head_mm  : num [1:136] 31.2 31 32 30.8 30.6 32.3 31.7 32.3 31.5 32.5 ...
##  $ length_humerus_in    : num [1:136] 0.687 0.738 0.736 0.736 0.733 0.743 0.741 0.738 0.715 0.752 ..
##  $ length_femur_in      : num [1:136] 0.668 0.704 0.709 0.709 0.704 0.718 0.688 0.709 0.706 0.731 ..
##  $ length_tibiotarsus_in: num [1:136] 1.02 1.09 1.11 1.18 1.15 ...
##  $ skull_width_in_      : num [1:136] 0.587 0.606 0.611 0.602 0.602 0.602 0.584 0.607 0.575 0.623 ..
##  $ keel_length_in       : num [1:136] 0.83 0.847 0.842 0.841 0.846 0.828 0.839 0.869 0.821 0.888 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   bumpus_number = col_double(),
##   ..   sex = col_character(),
##   ..   age = col_character(),
##   ..   survival = col_character(),
##   ..   total_length_mm = col_double(),
##   ..   alar_extent_mm = col_double(),
##   ..   weight_g = col_double(),
##   ..   length_beak_head_mm = col_double(),
##   ..   length_humerus_in = col_double(),
##   ..   length_femur_in = col_double(),
##   ..   length_tibiotarsus_in = col_double(),
##   ..   skull_width_in_ = col_double(),
##   ..   keel_length_in = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

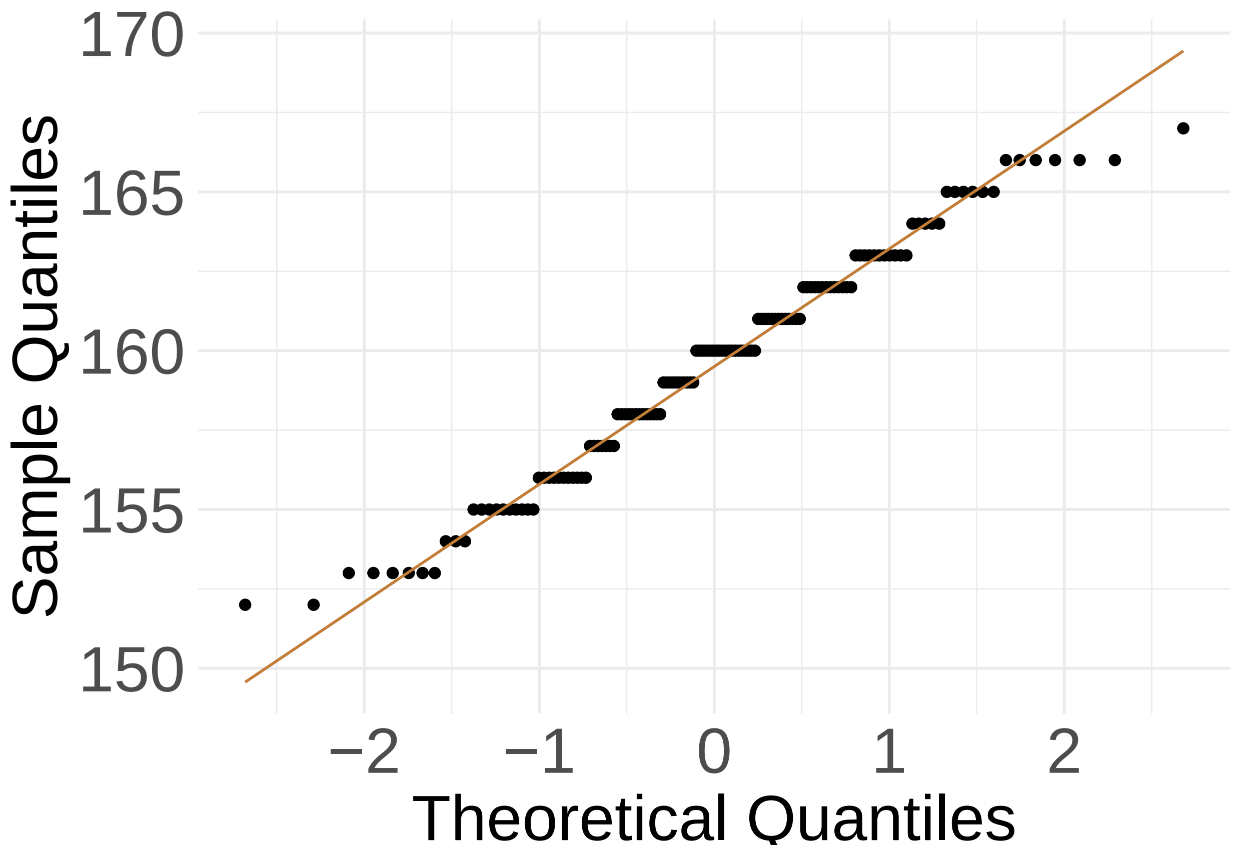
```r
#make histogram

ggplot(bumpus, aes(x =total_length_mm )) +
  geom_histogram(color="black") + #add black line around bars

  bb_theme #here I am using my custom theme
```

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



B) Plot a QQ plot for total length. Does the data fall approximately along a straight line in the QQ plot? If so, what does this imply about the fit of these data to a normal distribution?

```r
#qqplot
ggplot(bumpus, aes(sample=total_length_mm)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```



---

3. Data Transformations

Remember:

- The Normal distribution is *very common* and the CLT (Central Limit Theorem) is very useful
- There are a bunch of statistical approaches made for data with some form of normality assumption.

- But sometimes data are *too far* from normal to be modeled as if they are normal
- Or, details of a statistical distribution lead to breaking other assumptions of statistical tests. When this happens, we have a few options:

1. We can transform the data to meet our assumptions

2. We can permute and bootstrap! (sampling/simulation approaches)
3. We can use/develop tools to model the data as they are actually distributed

Let's look at option 1. In this case, we can try to use a simple mathematical transformation on each data point to create a list of numbers that still convey the information about the original question but that may be better matched to the assumptions of our statistical tests.

With a transformation, we apply the same mathematical function to each value of a given numerical variable for individual in the data set. With a log-transformation, we take the logarithm of each individual's value for a numerical variable.

Common transformations There are numerous common transformations that will make data normal, depending on their initial shape.

Name	Formula	What type of data?
Log	$Y' = \log_x(Y + \epsilon)$	Right skewed
Square-root	$Y' = \sqrt{Y + 1/2}$	Right skewed
Reciprocal	$Y' = 1/Y$	Right skewed
Square	$Y' = Y^2$	Left skewed
Exponential	$Y' = e^Y$	Left skewed

Chapter 13 of Whitlock and Schluter has more info on this, but for now let's learn how to do one of the most common data transformations, the **log-transformation**.

- Be careful when log-transforming!! All data with a value of zero or less will disappear.

- Try calculating `log(0)` or `log10(0)`.
- For this reason, we often use a `log1p` transform, which adds one to each number before logging them.

- Also, it will only improve the fit of the normal distribution to the data in cases when the **frequency distribution of the data is right-skewed.**

To take the log transformation for a variable in R is very simple. We simply use the function `log()`, and apply it to the vector of the numerical variable in question. For example, to calculate the log of age for all passengers on the Titanic, we use the command:

```
#eval=F so we don't print three pages of values in the pdf, but you should run it
log(titanicData$age)
```

This will return a vector of values, each of which is the log of age of a passenger.

Case study 2:Body mass in mammals

A) The file `MammalSizes.csv`"in theinput_files/' folder contains information on the body mass of various mammal species.

```
#code here
#read it in. Call it mammal_sizes
mammal_sizes<-readr::read_csv("input_files/MammalSizes.csv")
```

```
## New names:
## Rows: 651 Columns: 11
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (3): Order, Family, Binomial dbl (8): ...1, BrainMass, BodyMass, EQ,
## Defense_Type, Defense, Insectivory, ...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```r
#have a look
head(mammal_sizes)
```

```
## # A tibble: 6 x 11
##    ...1 Order      Family Binomial BrainMass BodyMass    EQ Defense_Type Defense
##   <dbl> <chr>      <chr>  <chr>        <dbl>    <dbl> <dbl>        <dbl>   <dbl>
## 1     1 Afrosoric~ CHRYS~ Chrysoc~      0.7        49 0.702            0       0
## 2     2 Afrosoric~ CHRYS~ Chrysoc~      1.06       50 1.05             0       0
## 3     3 Afrosoric~ TENRE~ Echinop~      0.62       88 0.403            1      18
## 4     4 Afrosoric~ TENRE~ Hemicen~      0.83      110 0.455            1      21
## 5     5 Afrosoric~ TENRE~ Limnoga~      1.15       92 0.721            0       0
## 6     6 Afrosoric~ TENRE~ Microga~      0.42       15 1.01             0       0
## # i 2 more variables: Insectivory <dbl>, Habitat_Openness <dbl>
```

```r
#check the structure of the dataset. Make sure the BodyMass (given in kg) column is
↪  numeric.
str(mammal_sizes)
```

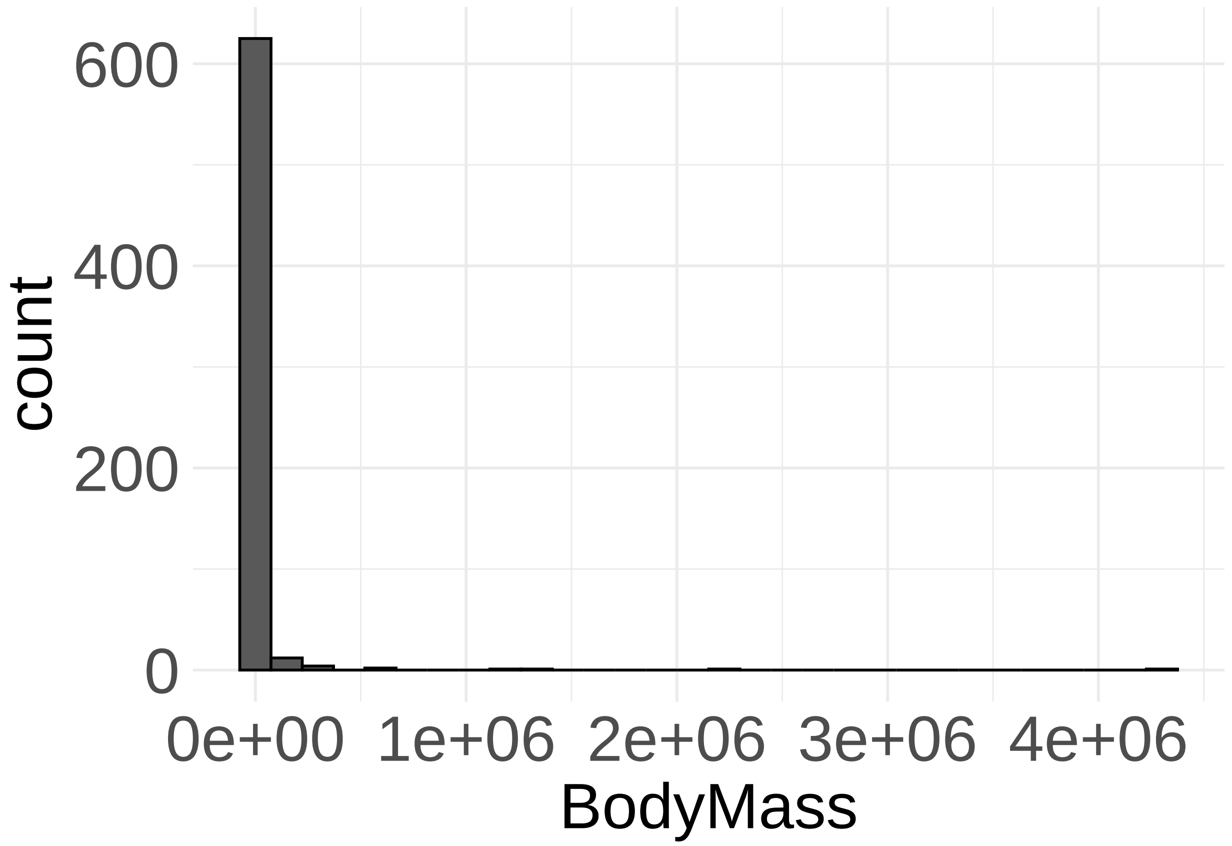
```
## spc_tbl_ [651 x 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1            : num [1:651] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Order           : chr [1:651] "Afrosoricida" "Afrosoricida" "Afrosoricida" "Afrosoricida" ...
##  $ Family          : chr [1:651] "CHRYSOCHLORIDAE" "CHRYSOCHLORIDAE" "TENRECIDAE" "TENRECIDAE" ...
##  $ Binomial        : chr [1:651] "Chrysochloris_asiatica" "Chrysochloris_stuhlmanni" "Echinops_telfa
##  $ BrainMass       : num [1:651] 0.7 1.06 0.62 0.83 1.15 0.42 0.56 0.79 0.58 4.16 ...
##  $ BodyMass        : num [1:651] 49 50 88 110 92 15 33 50 44 660 ...
##  $ EQ              : num [1:651] 0.702 1.046 0.403 0.455 0.721 ...
##  $ Defense_Type    : num [1:651] 0 0 1 1 0 0 0 0 0 0 ...
##  $ Defense         : num [1:651] 0 0 18 21 0 0 0 0 0 0 ...
##  $ Insectivory     : num [1:651] 1 1 0 1 0 1 1 1 1 0 ...
##  $ Habitat_Openness: num [1:651] 0.5 0.55 0.53 0.55 0.2 0.45 0.45 0.1 0.4 0.3 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_double(),
##   ..   Order = col_character(),
##   ..   Family = col_character(),
##   ..   Binomial = col_character(),
##   ..   BrainMass = col_double(),
##   ..   BodyMass = col_double(),
##   ..   EQ = col_double(),
##   ..   Defense_Type = col_double(),
##   ..   Defense = col_double(),
##   ..   Insectivory = col_double(),
##   ..   Habitat_Openness = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

B) Plot the distribution of body mass, and describe its shape. Does this look like it has a normal distribution?

```r
#code here
ggplot(mammal_sizes, aes(x =BodyMass)) +
  geom_histogram(color="black") + #add black line around bars
  bb_theme #here I am using my custom theme
```

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

```
## Warning: Removed 4 rows containing non-finite values (`stat_bin()`).
```



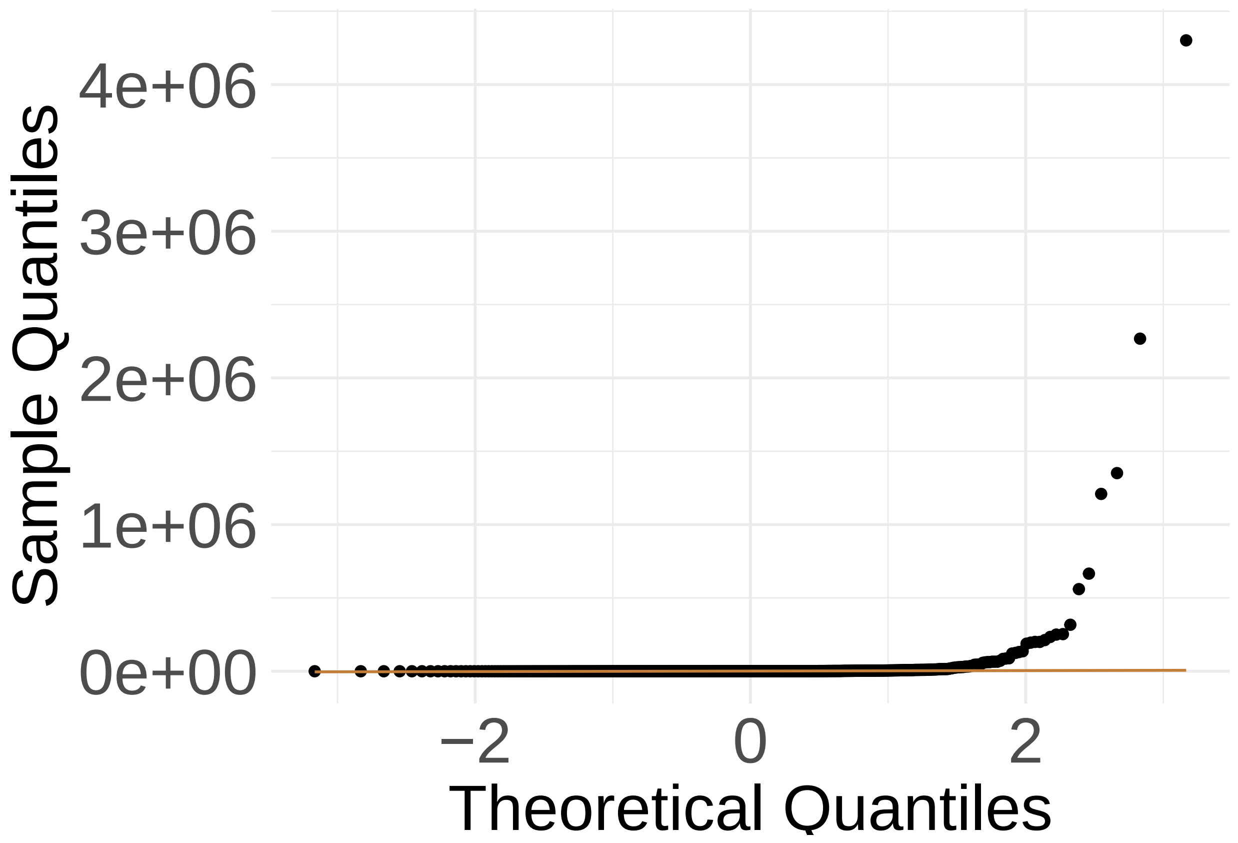
What a terrible plot! Does this seem normally distributed?

C) Make a qqplot for the body mass.

```
#code here
ggplot(mammal_sizes, aes(sample=BodyMass)) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```

```
## Warning: Removed 4 rows containing non-finite values (`stat_qq()`).
```

```
## Warning: Removed 4 rows containing non-finite values (`stat_qq_line()`).
```



Not very informative! But look at those points in the far right - not normal at all!

D) If you think the distribution in C) is skewed, what is the direction of the skew? Consult the table above and decide whether a log-transformation is appropriate.

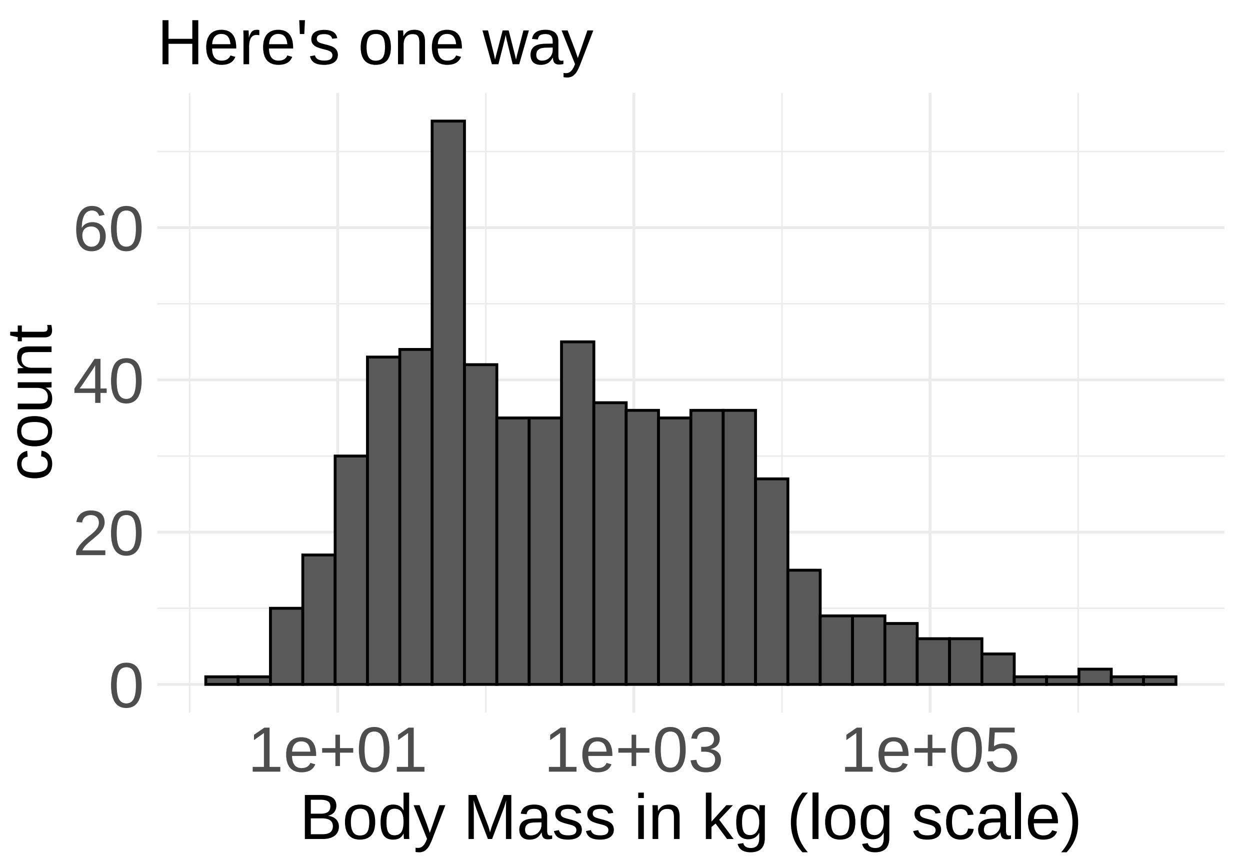
If so, transform the body mass data with a log-transformation. Plot the distribution of log body mass (remember the difference between log and log10, but also know that either one is fine to use). Make sure to label the axes properly.

Describe the new distribution. Does it seem normal?

```r
#code here
#option 1
ggplot(mammal_sizes, aes(x =BodyMass)) +
  geom_histogram(color="black") + #add black line around bars
         scale_x_continuous(trans='log10')+ #log10 scale
        ggtitle("Here's one way")+
        xlab("Body Mass in kg (log scale)")+
  bb_theme #here I am using my custom theme
```

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

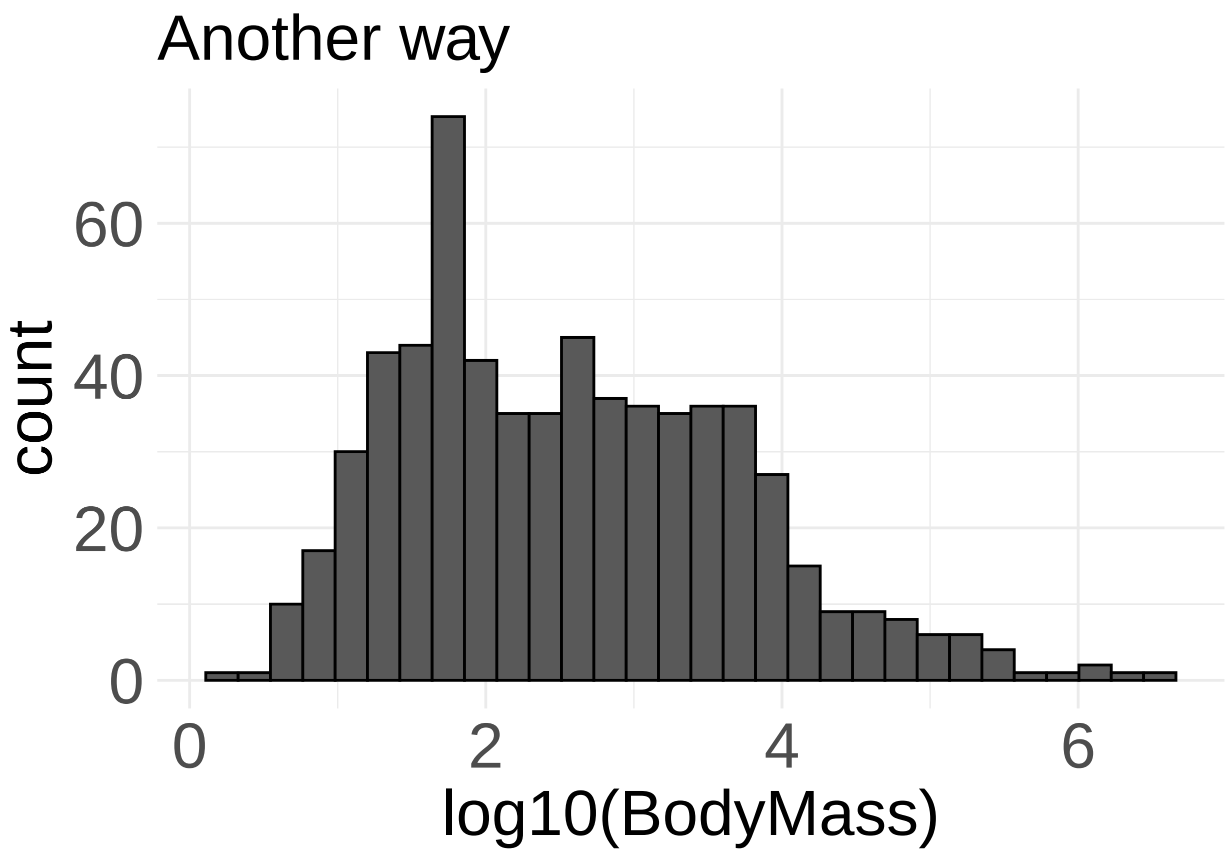
```
## Warning: Removed 4 rows containing non-finite values (`stat_bin()`).
```



```
#option 2
ggplot(mammal_sizes, aes(x =log10(BodyMass))) +
  geom_histogram(color="black") + #add black line around bars
        ggtitle("Another way")+
  bb_theme #here I am using my custom theme
```

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

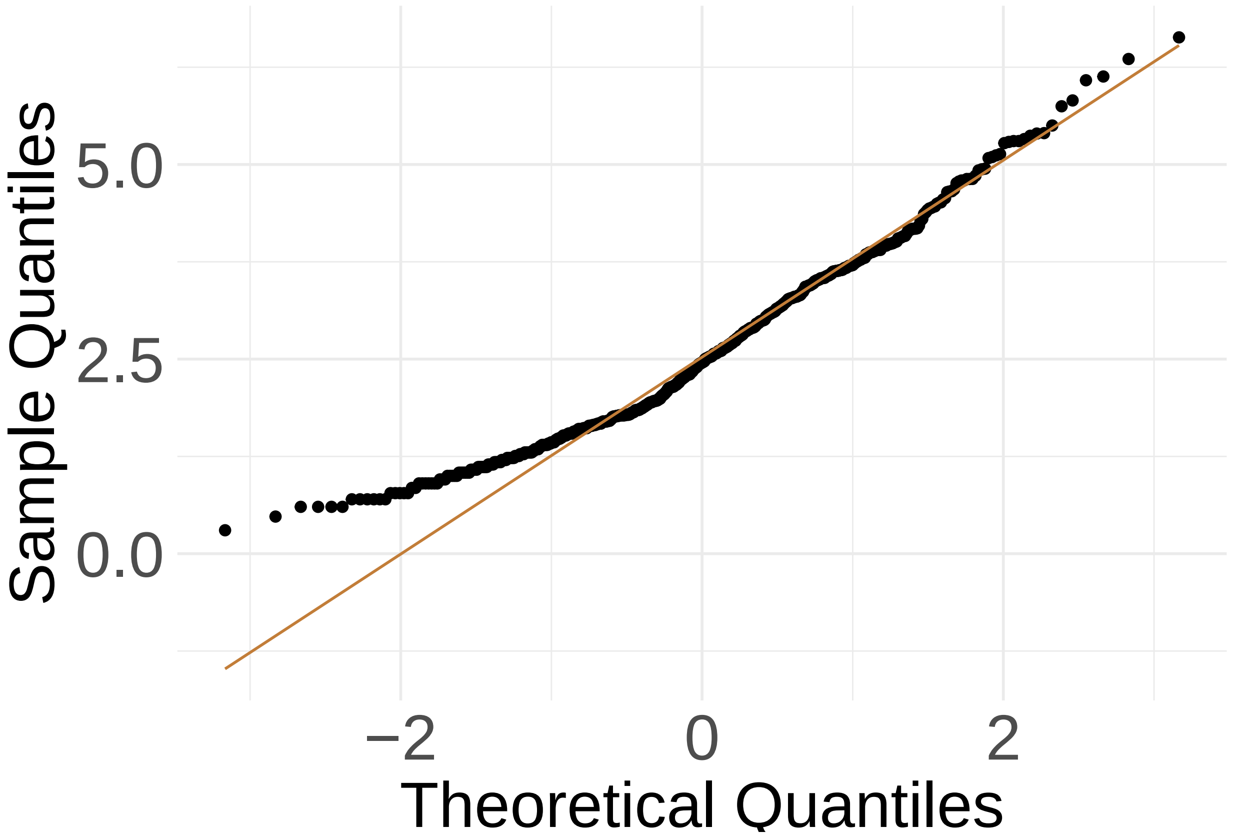
```
## Warning: Removed 4 rows containing non-finite values (`stat_bin()`).
```



E) Make a qqplot for the body mass rescaled using log (or log10).

```
#code here
ggplot(mammal_sizes, aes(sample=log10(BodyMass))) + # x is z transformed data, y is data
        geom_qq() +
        geom_qq_line(col="#C27D38") +
        xlab("Theoretical Quantiles")+
        ylab("Sample Quantiles")+
        bb_theme #my custom ggplot theme
```

```
## Warning: Removed 4 rows containing non-finite values (`stat_qq()`).

## Warning: Removed 4 rows containing non-finite values (`stat_qq_line()`).
```



And voilà!

References

A guide to dnorm, pnorm, qknorm, and rnorm in R. (Accessed December 5 2023) https://www.statology.org/dnorm-pnorm-rnorm-qnorm-in-r/