Statistics with Sparrows - 02

Julia Schroeder

July 2020

Learning aims

Understand how we quantify distributions

Encounter z-scores and z-distribution

Describing distributions

Let's get down to the hard stuff: statistics! But first, watch the video SwS.v.02 Afterwards, we'll get our hands dirty and do the things we've seen in the video - and, of course stumble across all the hurdles that come our way when using R and heterogenous ecological data!

We will use the sparrow data as example. But before we begin, we clear our workspace. Never forget!

```
rm(list=ls())

setwd("/cloud/project/SwS02")

d<-read.table("SparrowSize.txt", header=TRUE)
str(d)

## what you see below might differ from what you will see in your r console –
## it differs by R version. Don't be distracted by these.

## 'data.frame':    1770 obs. of  8 variables:
##  $ BirdID: int  1 2 2 2 2 2 2 2 2 ...
##  $ Year  : int  2002 2001 2002 2003 2004 2004 2004 2004 2004 2005 ...
##  $ Tarsus: num  16.9 16.8 17.2 17.5 17.8 ...
##  $ Bill  : num  NA NA NA 13.5 13.4 ...
##  $ Wing  : num  76 76 76 76 77 78 77 77 77 77 ...
##  $ Mass  : num  23.6 27.5 28.1 27.8 26.5 ...
##  $ Sex   : int  0 1 1 1 1 1 1 1 1 ...
##  $ Sex.1 : chr  "female" "male" "male" "male" ...

names(d)

## [1] "BirdID" "Year"   "Tarsus" "Bill"   "Wing"   "Mass"   "Sex"    "Sex.1"

head(d)

##   BirdID Year Tarsus Bill Wing  Mass Sex  Sex.1
## 1      1 2002   16.9   NA   76 23.60   0 female
```

```
## 2      2 2001   16.8   NA   76 27.50   1   male
## 3      2 2002   17.2   NA   76 28.10   1   male
## 4      2 2003   17.5 13.5   76 27.75   1   male
## 5      2 2004   17.8 13.4   77 26.50   1   male
## 6      2 2004   17.7 13.1   78 26.00   1   male
```

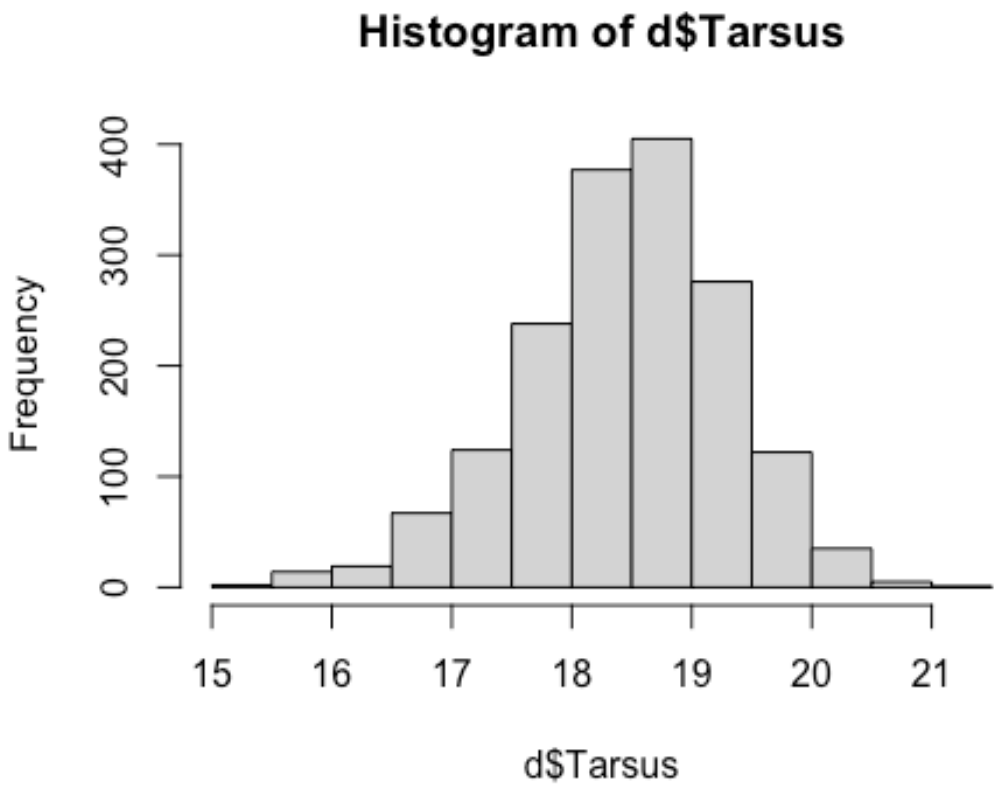
There are a few NA's - and we will deal with them soon. We already know the data structure. Now let's check the distribution of the data. We usually do this with a histogram. For now, we will work with data on the length of a bird's tarsus, that's the leg of a bird. We have a lot of data to deal with here!

```
length(d$Tarsus)

## [1] 1770
```

That's a good sample size! So let's get cracking!

Histograms, mean, median and mode

```
hist(d$Tarsus)
```



This looks like a normal distribution, doesn't it? It might lean a bit to the right, though. What is a normal distribution? Many data that we collect will be expected to approximately follow a normal distribution. Therefore, many statistics, - more explicitely, parametric

statistics, rely on the data being normally distributed. So it's a good idea to google them that they do actually look somewhat normal. However, the actual expectation of many tests is that not the data themselves, but the residuals are normally distributed. Confused much? No need to be - it's less important as it sounds, and there's a few ways around it. However it's a good idea to think ahead of designing your experiment about this - using some 5-point scale (ice cold, cold, meh, warm, hot) might be less of a good idea than acutally measuring it in a unit on a continous scale (like, degree Celsius).

Now, how can we describe this distribution of values best? Usually, we use a description of centrality, and one of the spread. What is centrality?

Centrality, mean, median and mode in normally distributed data

```
mean(d$Tarsus)

## [1] NA
```

Uuups. We have missing values in our dataset (NAs). This is reflected by the error message. How to deal with this? A quick look up in help reveals that we can use an argument na.rm that is by default set to "FALSE". When set to TRUE, it means that NAs are stripped before computation. That's what we want. Let's give it a try:

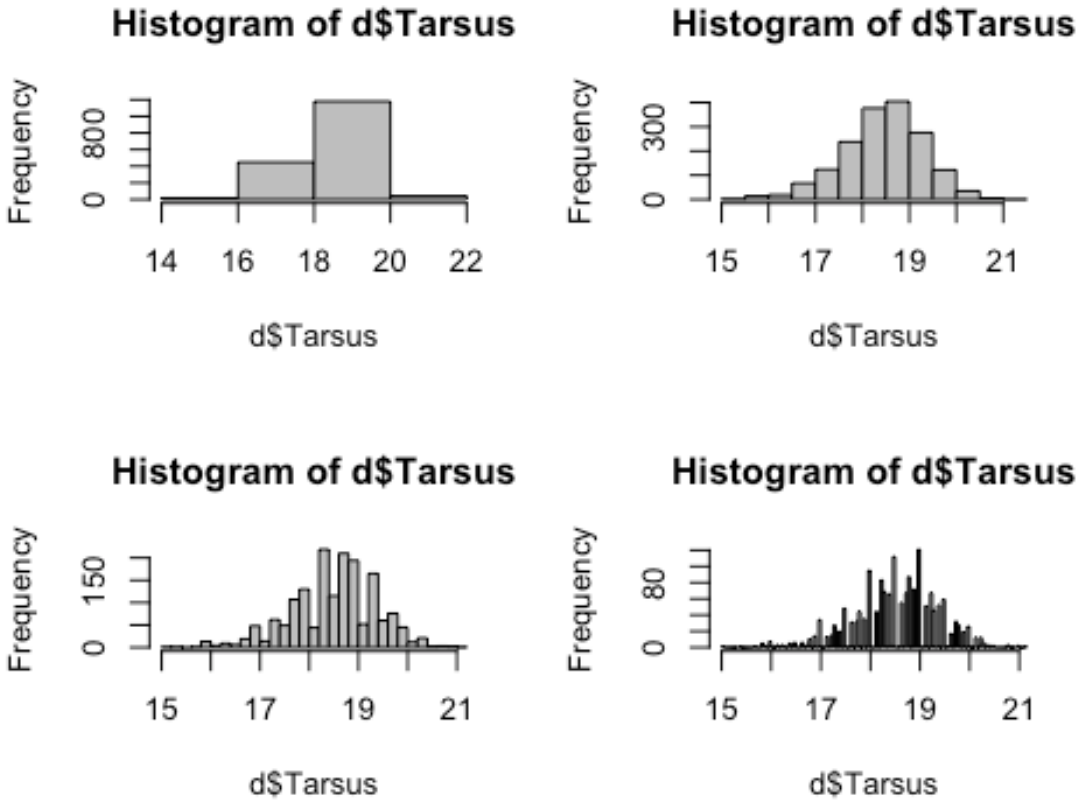
```
help(mean)
mean(d$Tarsus, na.rm = TRUE)

## [1] 18.52335

median(d$Tarsus, na.rm = TRUE)

## [1] 18.6

mode(d$Tarsus)

## [1] "numeric"
```

Now, this worked at least two of three times. What's with the odd result for mode? The mode function returns a description of the type of object. It tells us that d$Tarsus is a numerical vector. What does this mean? If we have continous data, we have a hard time estimating the mode. That is because the mode is the most frequently occuring value. Why do you think it is hard to estimate it in a continous dataset?

Because most values occur only once.

Let's play with the data a bit more and look how the distribution changes, and the (supposed) mode. Also, can you remember what par(mfrow=c(2,2)) does?

```
par(mfrow = c(2, 2))
hist(d$Tarsus, breaks = 3, col="grey")
hist(d$Tarsus, breaks = 10, col="grey")
hist(d$Tarsus, breaks = 30, col="grey")
hist(d$Tarsus, breaks = 100, col="grey")
```



Have a think:

Think about this. Why do you think there are these odd gaps?

The number of breaks determines the number of bins that is used to draw the histogram. And at some point, the resolution is larger than the resolution of the measuring device! Clearly the mode is somewhere between 18 and 19. It would be nice to get this more precisely. However, the is currently no package that calculates the mode. So we have to do it ourselves. The mode is the value that occurs most frequently in the data. Let's see - first, we need to count how often each value occurs. Seems familiar from the data strucutre session? It is:

```
head(table(d$Tarsus))
```

```
##
##          15 15.39999962 15.69999981 15.80000019 15.89999962          16
##           1           1           1           5           1           7
```

That really long table! See what I did there with head()?

Some values have been measured 7 times, others only once. Ok. Maybe it would be reasonable to round the values first, because there's clearly some odd issue at play - nobody can measure a tarsus that precisely. We'll round to one decimal.

```
?round
```

That's what we need. Here we can see how to code it so we get 1 decimal:

```
d$Tarsus.rounded<-round(d$Tarsus, digits=1)
head(d$Tarsus.rounded)
```

```
## [1] 16.9 16.8 17.2 17.5 17.8 17.7
```

Better. But now we need to find out which one is the highest. Didn't we sort earlier with dplyr?

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
TarsusTally <- d %>% count(Tarsus.rounded, sort=TRUE)
TarsusTally
```

```
## # A tibble: 55 x 2
##    Tarsus.rounded     n
##             <dbl> <int>
##  1           19     121
##  2           18.5   112
##  3           18      95
##  4           18.8    87
##  5           NA      85
##  6           18.2    83
##  7           18.9    73
##  8           18.3    70
##  9           18.4    68
## 10           18.7    68
## # … with 45 more rows
```

This works- ish. The top number (19) is the mode. But we can also see that in the 5th row, there's a "NA" - that's a rather frequent value. So in some datasets that might show up as the most frequent value. Better to remove the NAs first.

```
d2<-subset(d, d$Tarsus!="NA")
length(d$Tarsus)-length(d2$Tarsus)
```

```
## [1] 85
```

Fits - the difference in length between both datasets is 85- the tally for NA is the same, so this is correct.

```
TarsusTally <- d2 %>% count(Tarsus.rounded, sort=TRUE)
TarsusTally
```

```
## # A tibble: 54 x 2
##    Tarsus.rounded     n
##             <dbl> <int>
##  1           19     121
##  2           18.5   112
##  3           18      95
##  4           18.8    87
##  5           18.2    83
##  6           18.9    73
##  7           18.3    70
##  8           18.4    68
##  9           18.7    68
## 10           19.2    67
## # … with 44 more rows
```

Now all we need is to extract the first value of the first column. The result of anything dplyr is usually a tibble, and that's accessed with square brackets [] and numbers. I don't know exactly how (maybe Josh knows by heart) so I will have to play around a bit until I find the right solution:

```
TarsusTally[1]
```

```
## # A tibble: 54 x 1
##    Tarsus.rounded
##             <dbl>
##  1           19
##  2           18.5
##  3           18
##  4           18.8
##  5           18.2
##  6           18.9
##  7           18.3
##  8           18.4
##  9           18.7
## 10           19.2
## # … with 44 more rows
```

This extracts the first column

```
TarsusTally[2]
```

```
## # A tibble: 54 x 1
##        n
##    <int>
##  1   121
```

```
##  2   112
##  3    95
##  4    87
##  5    83
##  6    73
##  7    70
##  8    68
##  9    68
## 10    67
## # … with 44 more rows
```

And this the second (not so surprisingly).

```
TarsusTally[[1]]
```

```
##  [1] 19.0 18.5 18.0 18.8 18.2 18.9 18.3 18.4 18.7 19.2 19.5 18.6 19.1 19.4
17.5
## [16] 19.3 17.8 18.1 17.9 17.0 17.6 17.7 19.7 17.3 19.8 20.0 17.4 19.9 19.6
17.2
## [31] 16.9 17.1 20.1 20.2 16.8 16.0 16.5 16.6 20.3 15.8 16.4 16.2 16.1 16.7
20.4
## [46] 20.8 20.5 21.0 15.0 15.4 15.7 15.9 16.3 21.1
```

This gives us a normal vector (takes away the wrapping and naming of the column). Now I can simply access the first element by adding [1]:

```
TarsusTally[[1]][1]
```

```
## [1] 19
```

So, let's try this again:

```
mean(d$Tarsus, na.rm = TRUE)
```

```
## [1] 18.52335
```

```
median(d$Tarsus, na.rm = TRUE)
```

```
## [1] 18.6
```

```
TarsusTally[[1]][1]
```

```
## [1] 19
```

In normally distributed data, mean, median and mode should be fairly similar. If the distribution is perfectly normal, they should even be identical. As the skew of the distribution increases, these three measures diverge.

Range, variance and standard deviation

R is really good because you can guess the name of certain functions. We can just guess that "range" might give us the range of values in a vector. Either type it into the console (?range) or google it.

```
range(d$Tarsus, na.rm = TRUE)

## [1] 15.0 21.1

range(d2$Tarsus, na.rm = TRUE)

## [1] 15.0 21.1

var(d$Tarsus, na.rm = TRUE)

## [1] 0.7404059

var(d2$Tarsus, na.rm = TRUE)

## [1] 0.7404059
```

Now, range is easy - that's the minimum and maximum values. Now let's look again more closely at the variance. It's often ignored, yet it plays a very central role, and you should embrace it!

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}$$

So, the variance is described as σ^2. It should be apparent, from the video lecture, why it's a square. Let's have a closer look at the formula. It includes the mean, but also other stuff. The numerator is called the sum of squares, and is sometimes denoted by SS. This is super important. Why is it a sum of squares? Let's a have a look at how to get this. It's the sum of the squares of the *deviations* from the mean. I'll write out how to calculate this in R. We use the tarsus bit without NAs as then the length is correct.

```
sum((d2$Tarsus - mean(d2$Tarsus))^2)/(length(d2$Tarsus) - 1)

## [1] 0.7404059
```

Quick check:

```
var(d2$Tarsus)

## [1] 0.7404059
```

Now, why is it denoted as σ^2? If we square-root the variance, we get σ, and that is the standard deviation:

```
sqrt(var(d2$Tarsus))

## [1] 0.8604684
```

```
sqrt(0.74)
```

```
## [1] 0.8602325
```

```
sd(d2$Tarsus)
```

```
## [1] 0.8604684
```

Cool. Now we understand how to describe data that is about normally distributed - with measures of centrality (mean, median, mode) and measures that describe the spread (range, variance, standard deviation).

Z-scores and quantiles

Z-values, and z transformation (z-score, or standardized scores) is a super important topic that you will encounter very often and use very often. Z-scores come from a standardized normal distribution, with a mean of 0 and a standard deviation of 1.

Have a think: *What variance does this distribution have?*

This is not a trick question. It's an easy question for someone with an intuitive understanding of maths. People who don't have that might struggle. The answer is - the variance of z-standardized data with a sd of 1 is also 1. That's because the squareroot of 1 is 1. So, in this special case, the variance equals the standard deviation.

When we do stats it is often useful to transform our data so it follows exactly these rule - z-transforming data. You can do that by dividing the deviation from the mean by the standard deviation:

$$z = \frac{y - \overline{y}}{\sigma_y}$$

Here, σ_y means standard deviation of y. Ok, let's do this, and check is all went according to plan:

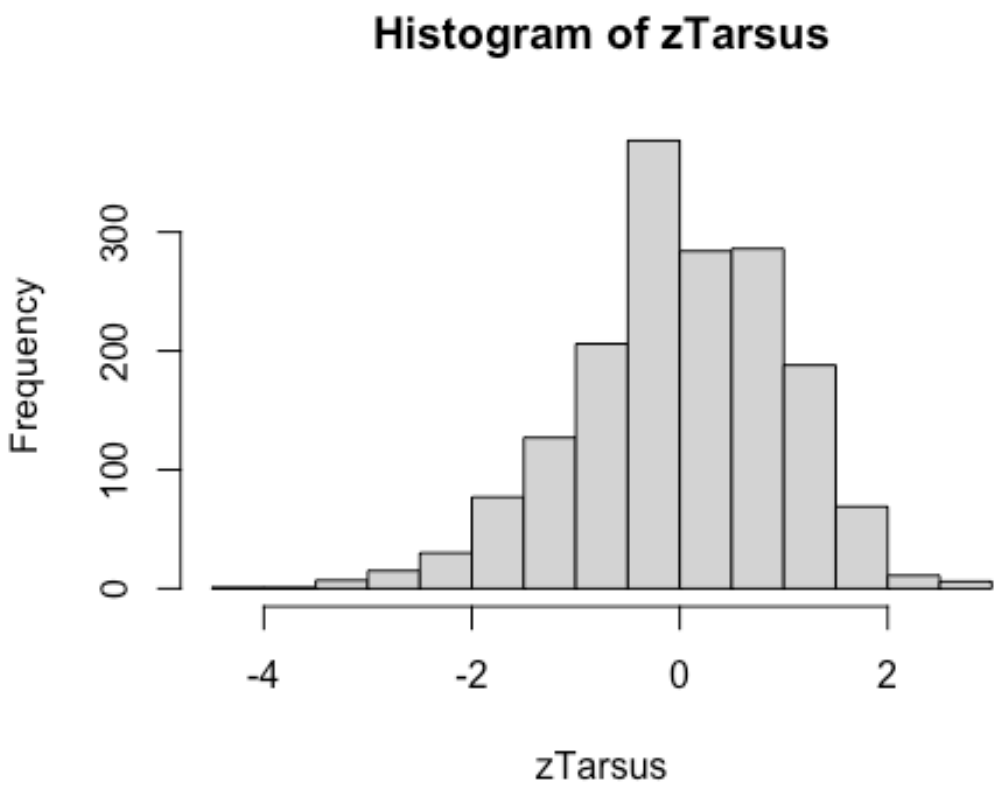
```
zTarsus <- (d2$Tarsus - mean(d2$Tarsus))/sd(d2$Tarsus)
var(zTarsus)
```

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

```
sd(zTarsus)
```

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

```
hist(zTarsus)
```

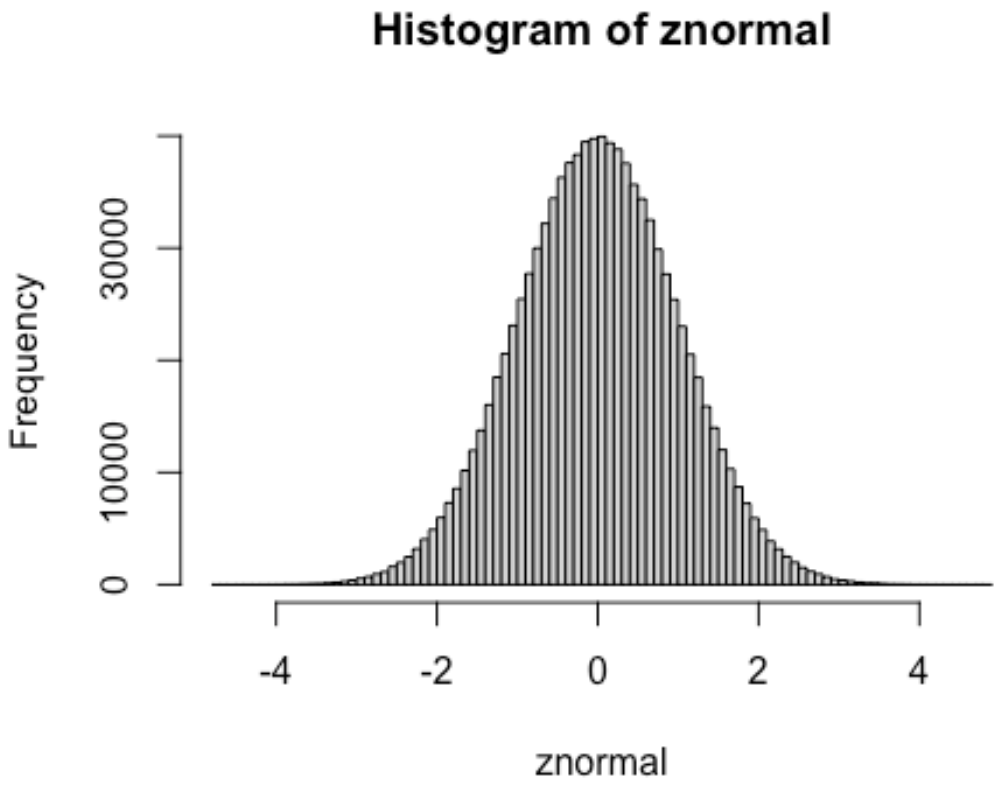


But, of course, there is a function for this in R.

Have a think: *Use Google to find this function, remember it, and use it often! Share the function you found in your group to see if you all found the same one!*

BTW, the reason we call them z-scores is because the normal distribution is also called the z-distribution. The neat thing about R is that it allows you to make us datasets from scratch that follow this distribution:

```
znormal <- rnorm(1e+06)
hist(znormal, breaks = 100)
```



That's a beautiful normal distribution. Let's examine it a bit closer:

```
summary(znormal)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -4.799191 -0.674387 -0.000260 -0.000521  0.673333  4.850767
```

Very helpful. We even get the central tendencies in one go. The quantiles refer to the respective cut-offs of the data distribution, think back to the beginning of this lecture! The median is the second quantile where 50% of data points are included. R can also give us other information on the normal distribution such as the value at a given quantile (qnorm) or the probability at a given value (not data randomly sampled from it, which we did above):

```
qnorm(c(0.025, 0.975))
pnorm(.Last.value)
```

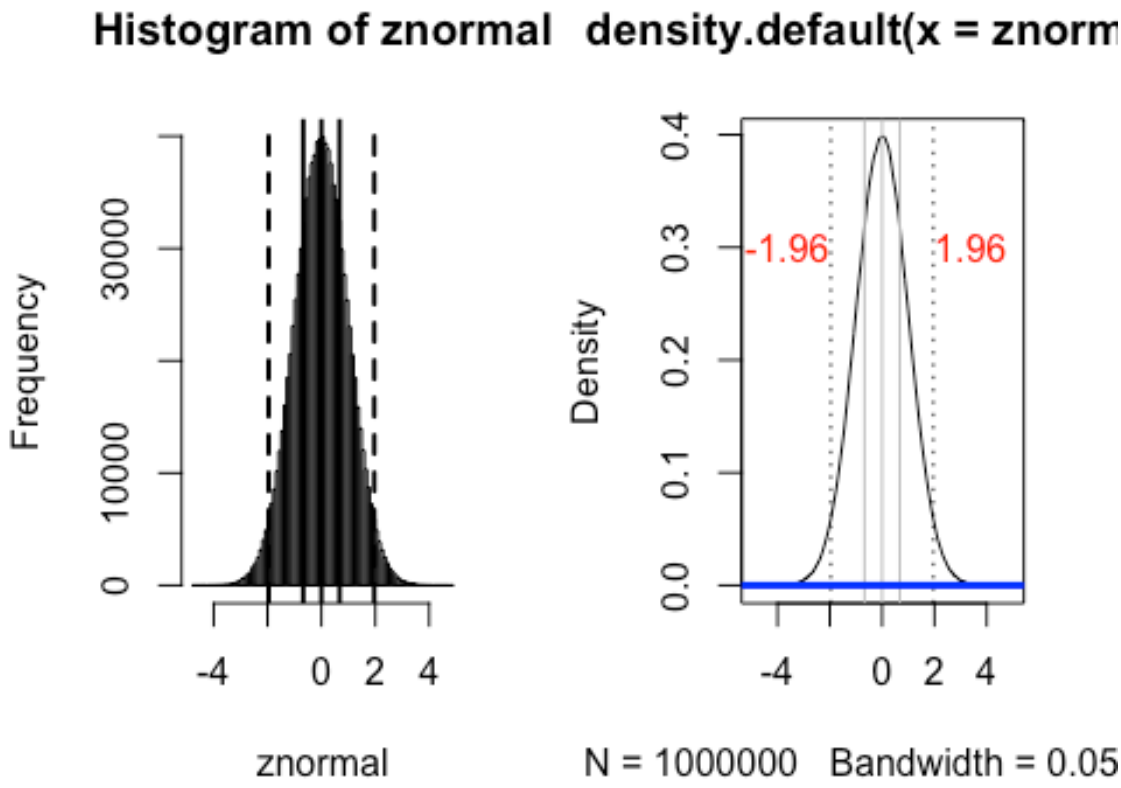
```
## [1] -1.959964  1.959964
```

```
## [1] 0.025 0.975
```

`qnorm(c(0.025,0.975))` gives us the 2.5% and 97.5% quantiles from the corresponding probability distribution. Both bracket 95% of all values in the distribution. And pnorm gets us the corresponding probabilities.

These last quantiles are important when we get to hypothesis testing, so remember them!

```
par(mfrow = c(1, 2))
hist(znormal, breaks = 100)
abline(v = qnorm(c(0.25, 0.5, 0.75)), lwd = 2)
abline(v = qnorm(c(0.025, 0.975)), lwd = 2, lty = "dashed")
plot(density(znormal))
abline(v = qnorm(c(0.25, 0.5, 0.75)), col = "gray")
abline(v = qnorm(c(0.025, 0.975)), lty = "dotted", col = "black")
abline(h = 0, lwd = 3, col = "blue")
text(2, 0.3, "1.96", col = "red", adj = 0)
text(-2, 0.3, "-1.96", col = "red", adj = 1)
```




```
## null device
##           1
```

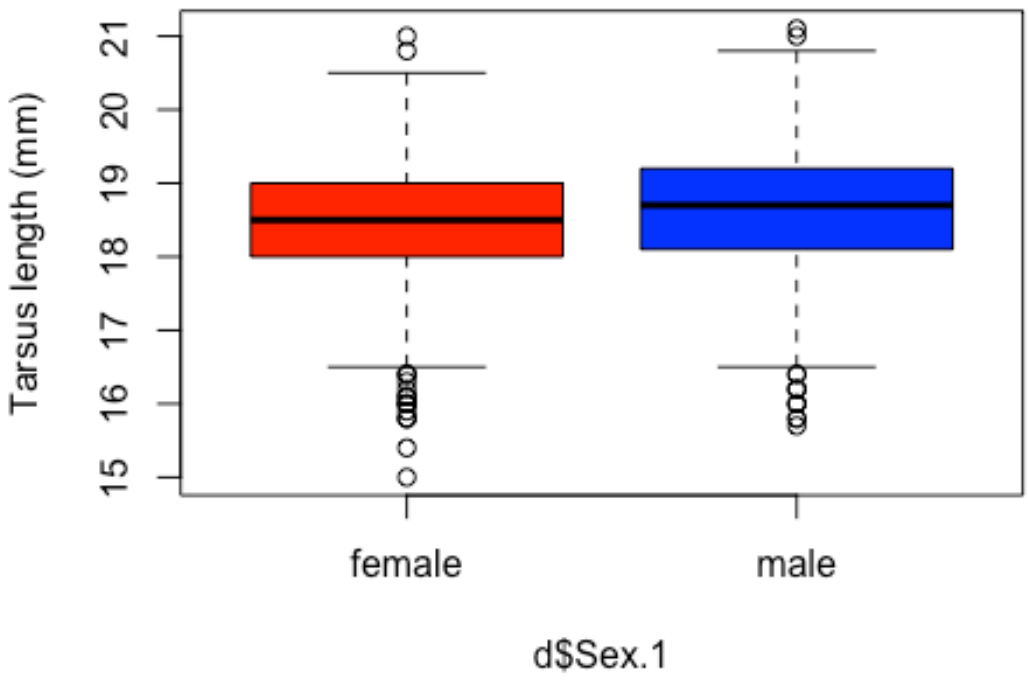
The 95% Confidence Interval (95% CI) is a very important property. It is the range of values the encompass the population true value with 95% probability. That means we will make an error in 5% of the times. Here it means that the true mean of the population that we sampled, and then used the data to calculate the 95CI, lies within this confidence interval 95 times out of 100 sampling events.

Now, let's a have a look at a sparrow example. I'll plot the tarsus length between the two sexes.

Exercises:

(1) What do you think why did I use dSex.1 here and not dSex? Discuss in your group.

```
boxplot(d$Tarsus~d$Sex.1, col = c("red", "blue"), ylab="Tarsus length (mm)")
```



What do you think these boxes and lines represent? You can find that our using.

```
?boxplot
```

Use the help function in R often and frequently. Many of them even detail examples when you scroll to the bottom.

(2) Discuss when a median or mode, might be useful over a mean.

(3) Discuss how the precision of a variable affects which bin size is good for histogramms Z-scores: What is the variance of z-standardized data?

(4) Which R function did you find to compute z-scores? How does it work?

(5) Why did I prefer d$Sex.1over d$Sex? What other solution could have have done to achieve the same goal?

(6) What is the difference between a distribution with mean = 0, var = 2, and mean = 0, var = 200?