Statistics with Sparrows - 03

Julia Schroeder

July 2020

HO 03

Data types

Learning aims

- Understand different data types
- Understand why continuous data are the most valuable data
- Understand that non-continuous data has limited value

```
rm(list=ls())
d<-read.table("SparrowSize.txt", header=T)
str(d)
```

Here we see the difference in data typees - BirdID is an integer, that means there are no decimals. Year, is an integer, too. Tarsus, Bill, Wing, Mass are numerical - that means they are continuous variables. Sex is an integer - here I denoted female with 0, and male with 1. Sex.1 is a character, that means, text. These data types are similar, but not exactly, the data types we're dealing with now.

So, which of these are continous, and what is categorical? The first thing to do is, naturally, have a good think. Clearly `Sex.1` is categorical, there is no question about that. And, while a number, `Sex` is, too. In `Sex`, females are denominated as 0, and males as 1, but that still does not mean males are somewhat *more* than females. As such, the 0 1 coding for sex is a hack, too, and we need to keep that into account when we look at results for this. Make a mental note!

Clearly, `Tarsus`, `Bill`, `Wing`, `Mass` are numerical - these are continuous measurements of length or mass, in milimeter and gram.

`BirdID` is a number, an integer. That means there's no decimals. There can also be no decimals, because between bird 1 and bird 2, there's no option for a bird 1.5. Also, the differences between these numbers are meaningless - we cannot substract bird 3245 from bird 12. In that sense, the `BirdID`s are really more equivalent to names, or nick-names, for

individual birds. We use numbers because that's easiest in our database - we can automate it - but really, these should be a proper non-numerical categorical variable. We can either keep that in mind (dangerous, because we might forget, pass the varible on to some statistical analysis, and R will interpret it as continous, because it's denoted as integer… so unless we can be 100% sure we'll never use it as a fixed covariate or so, we better tell R it's a categorical factor:

```
d$BirdIDFact<-as.factor(d$BirdID)
str(d$BirdIDFact)

##  Factor w/ 636 levels "1","2","3","4",..: 1 2 2 2 2 2 2 2 2 ...
```

And now, looking at the structure, we can see the difference this line makes - `BirdID` is an integer, and our new variable, `BirdIDFact`, is a factor, with 636 levels (that means, individual birds in this case), that are called names, and the names are numbers. R indicates that with "" around the number - these aren't interpreted as values, but rather, as text. We can check this by attempting to compute the mean of each:

```
mean(d$BirdID)

## [1] 290.2983
```

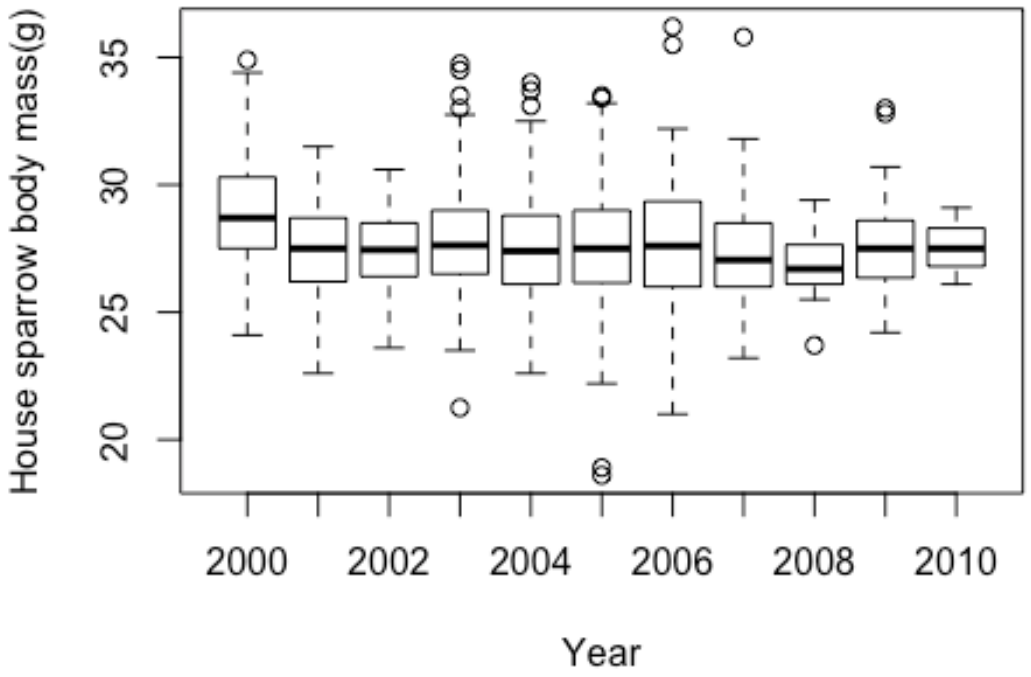
This is the mean of the old variable `BirdID` - that's clearly a completely insane value to compute, yet R does it - because it has no brains - it just follows the orders you give it. That's where your biologist skills are needed - to tell R what variables mean what biologically.

```
mean(d$BirdIDFact)

## Warning in mean.default(d$BirdIDFact): argument is not numeric or logical:
## returning NA

## [1] NA
```

And we get an error message when we try to take the mean of our new, factorial, and categorical, variable. That makes sense. We can't take a mean of a bunch of names.

Finally, we'll move on to the last variable - `Year`. Year is a tricky variable. It can be completely categorical - we can assume, that every year we go out to Lundy to measure sparrows, something is different. Some years, sparrows will be fat. Other years, the weather might be worse and they might be skinny. So, we can postulate that there will be annual variation in body mass of house sparrows. But we do not predict a directional change in this. We can thus plot `Year` as a factor:

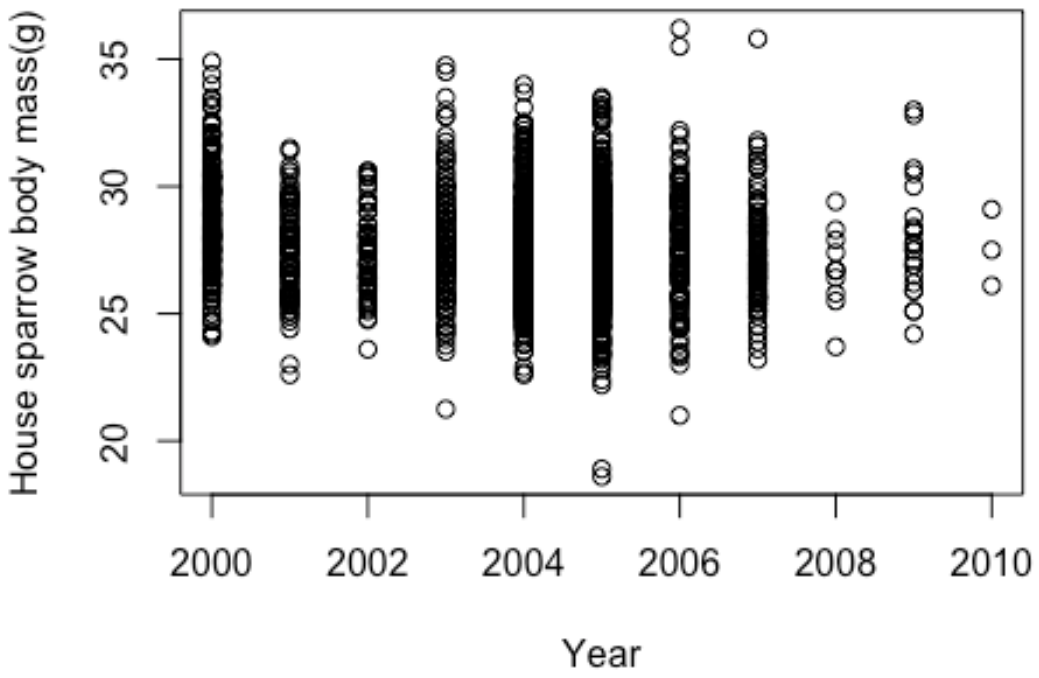
```
plot(d$Mass~as.factor(d$Year), xlab="Year", ylab="House sparrow body
mass(g)")
```



And we get a plot that shows quite some variation between years in body mass. R clearly interprets the Year variable as a categorical variable, because, even though we did not tell R to, it made a box plot. This will happen if your explanatory variable is categorical: because there is no possibility for decimals. If `Year` is interpreted as categorical, then there is no option for 2017.5. It also means there is no tangible difference between the years 2000 and 2010, and 2015, even though the difference between the first two is 10, and the latter two is only 5. `as.factor()` shapeshifts a variable into one where numbers are interpreted as text, and have no value per se attached. Note that, though, R is not completely stupid. It orders the years from early to late. However, that's not a particularly clever move by R. It is just that any alphanumerical text (anything with letters and numbers that's not a value) in R is by default ordered - you guessed it - alphanumerically. So, a year called y2020 would show up before a year called z2012. Sometimes, we can use this fact to our advantage - it is nicer to see the graph above with the years ordered, even though we know that it's meant to be categorical.

What would happen if we did not tell R that `Year` is a categorical variable (remember, we achieved that by adding `as.factor()` to `d$Year`)? Let's try!

```
plot(d$Mass~d$Year, xlab="Year", ylab="House sparrow body mass(g)")
```



Umm. That looks very different indeed! And that is because R interprets `Year` as a continuous variable, and it expects there to be at least the possibility for decimals, and because the differences between years are important, and the values mean something. 2020 is 10 years later than 2010. Now, when would that be of interest? For instance, when we think about climate change. In birds, there is the issue that the food for the birds young is dependent on temperature. Blue tits *Cyanistes caeruleus*, for example, rely on caterpillars that emerge with tree budburst. As the budburst occurs ever earlier in the year, due to spring warming, over the years, blue tits will start breeding too late, and have fewer chicks. Therefore, we can postulate, there is strong selection on early laying dates. And evolution should take care of this - over the years, blue tits should shift their laying dates to ever earlier dates. In this case, we are interpreting year as continuous variable.

In our sparrow data there is no laying date. So let's have a look at some blue tit data:

```
rm(list=ls())
```

We need to clean our workspace - if we are to read in another dataset with also a variable called `Year` there is mighty opportunity for confusion on the R front!

```
b<-read.table("BTLD.txt", header=T)
str(b)
```

Again, let's investigate the variables. `LD.in_AprilDays` is the laying date in days from the 1st of April onwards. As such, 3 is 3. April, and 32 is the first of May. Ok, that's, while an integer, a continous variable because the later they lay their eggs, the worse for their chicks. We also get information on the clutch size when the chicks are 7 days old - the number of offspring in the nest seven days after hatching. While there can hardly be a

fraction of a chick, it is still a continuous variable because the differences mean something. And a mean clutch size of

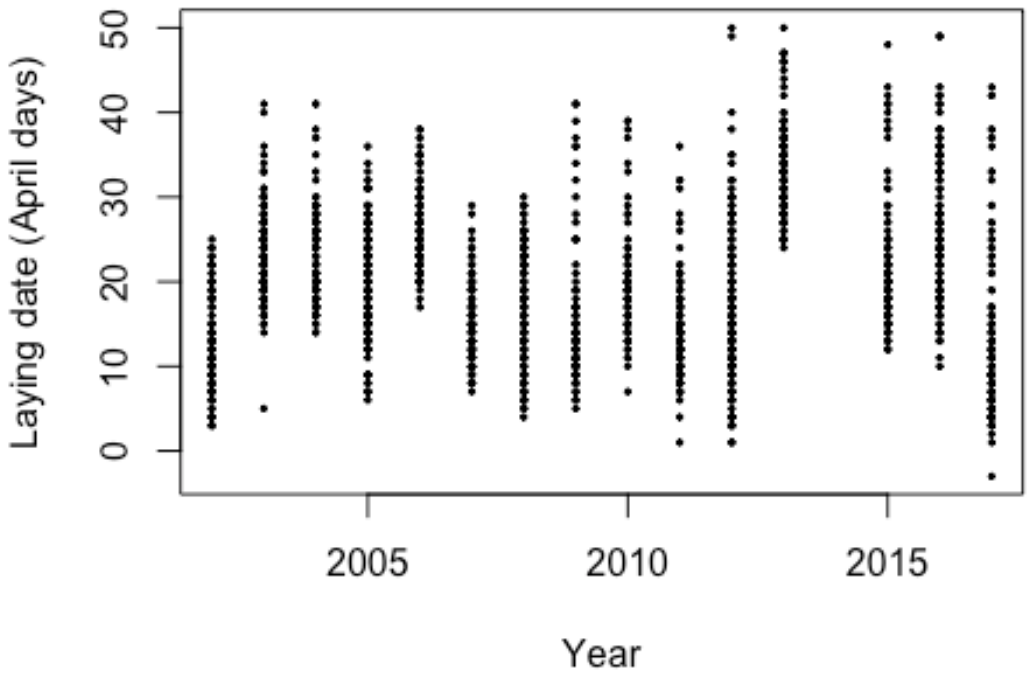
```
mean(b$ClutchsizeAge7, na.rm = TRUE)

## [1] 9.168243
```

is meaningful, so `ClutchsizeAge7` is a continuous variable.

Then there is an `IDFemale` - no males because males lay no eggs. This time, we get a number with a letter attached, and some versions of R automatically identifies that as a `Factor` - a categorical variable. If the version you're working with identifies it as a character, you may have to set it to be a factor – as we've done above. There's 1463 different females.

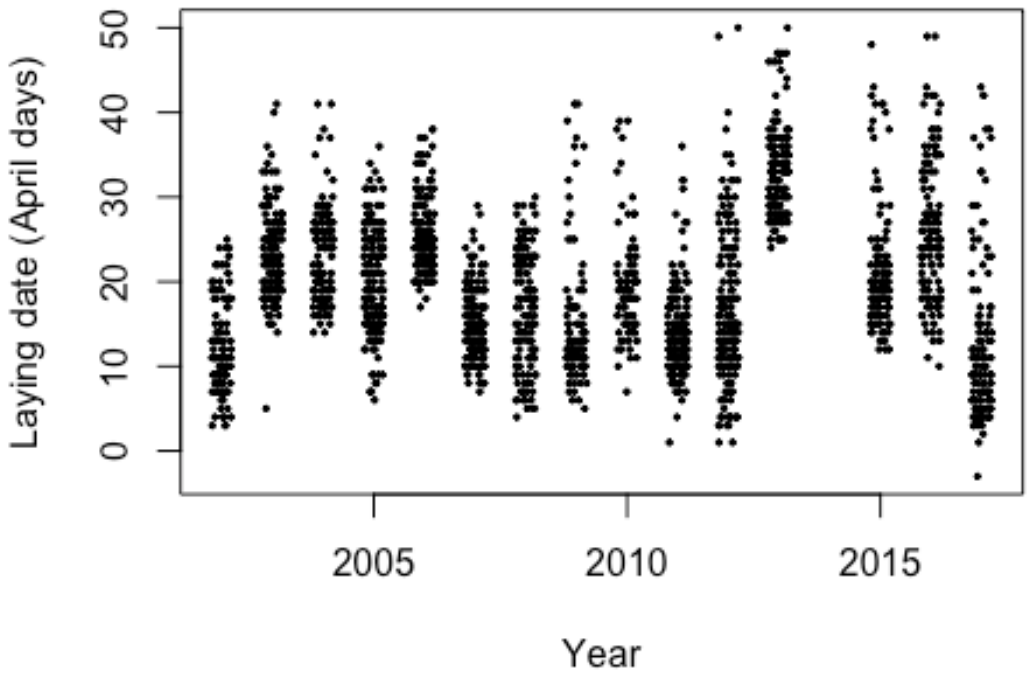
So we're left with `Year`, which is, as with the sparrows, considered by R as an integer. We, however, want it to be considered numerical - because we want to see whether there is a change from early years towards late years in laying date. So let's plot this:

```
plot(b$LD.in_AprilDays.~b$Year, ylab="Laying date (April days)", xlab="Year",
pch=19, cex=0.3)
```



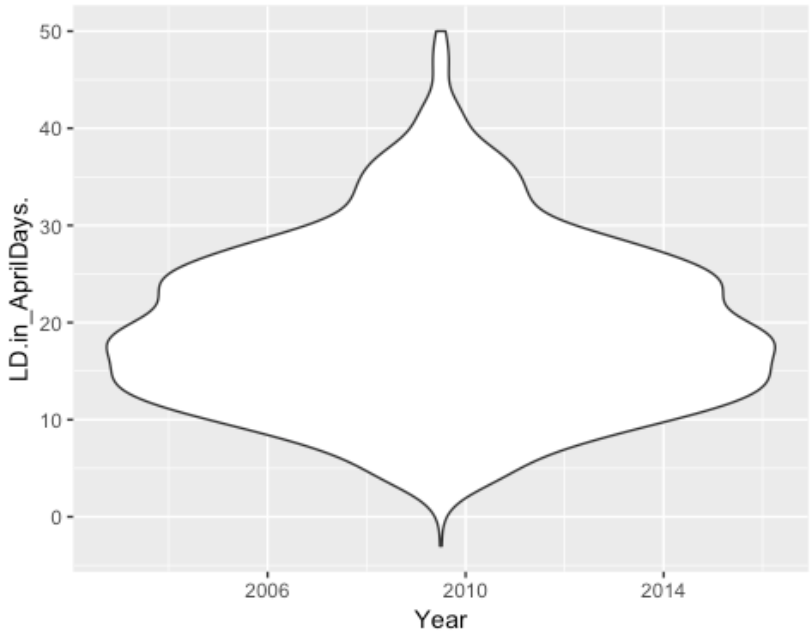
I find this really hard to see. That's becasue while both are continuous variables in the statistical sense, they are measured at a precision unit that gives us many measurements of the same value. We could change the plot to get a better view of the single points. There are multiple ways to achieve this. One, that I like particularly, is to add some random noise around each point in x-axis direction, so that we can see better. Adding that noise is done using the function `jitter()` in R:

```
plot(b$LD.in_AprilDays.~jitter(b$Year), ylab="Laying date (April days)",
xlab="Year", pch=19, cex=0.3)
```



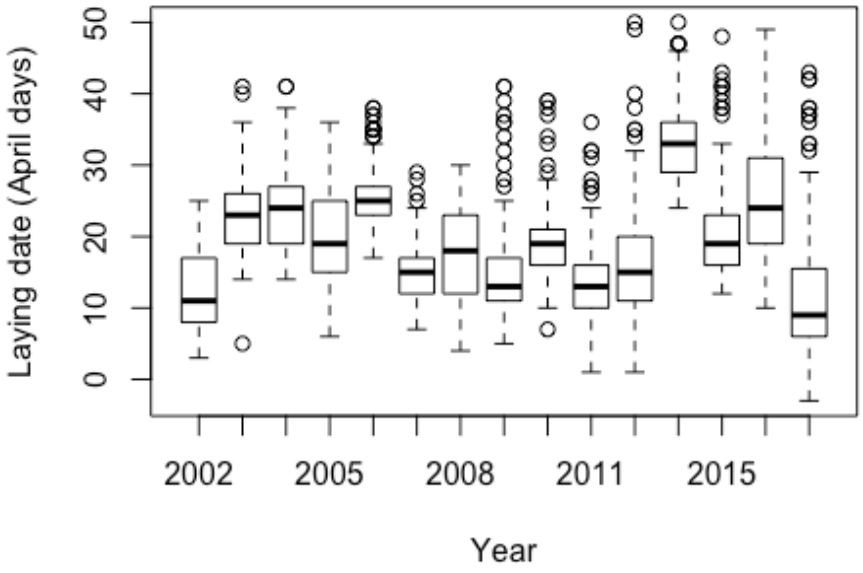
This is much better. Now we can see, at least approximately, how many points are there for each laying date in each year. What we have done here is visualise the distribution. Of course, you need to say so in the figure legend - otherwise people might think that the noise is meaningful biological data! There are other methods for this - another one that I like a lot is the violin plot. It's part of the ggplot2 package:

```
require(ggplot2)

## Loading required package: ggplot2

p <- ggplot(b, aes(x=Year, y=LD.in_AprilDays.)) +
  geom_violin()
p
```



Ummm. This doesn't look like we intended it to. It looks like R interpreted `Year` as a continuous variable - but that's what we want! Yes, it is what we want. We want to interpret `Year` as continuous variable, where later years lead to earlier laying dates and the other way around. But - we want to see the data still visualized categorically - but interpreted continuously. So, in a way, our old boxplot would do for an assessment of the biological interpretation:

```
boxplot(b$LD.in_AprilDays.~b$Year, ylab="Laying date (April days)",
xlab="Year")
```

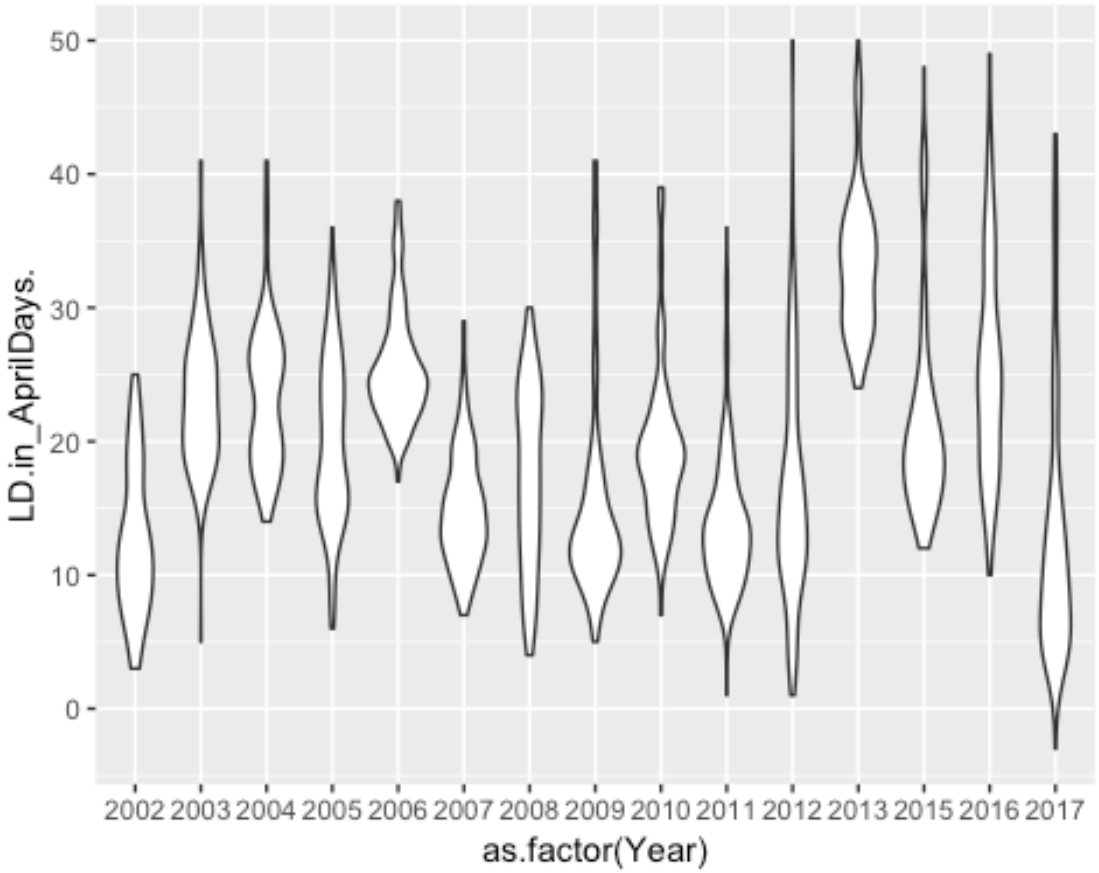


That's confusing? Well, yes, it is. Sometimes, you want to interpret a variable as continuous, and you also want to run the statistics on the variable as a continuous variable - for

instance, to calculate the mean. But for visualization, sometimes it can be best to plot the variable as categorical. Both, calculating the statistic, and making a visualization, have different purposes. And only a biologist can make sense of this – someone who only knows statistics couldn't. So, yes, you can have a variable in your dataset that R treats as integer, but that is biologically a factor. And, dependent on your research question and hypothesis, you might have to run the variable in your analysis (mean, t-test, linear model) as continuous variable (e.g. `Year` in the climate change context), and sometimes, you have to run it as a categorical variable, e.g, if you want to look at annual variation. Oddly enough, when you plot it for visualisation, your thinking has to be different - it has to go around the idea of how can you best visualize your idea, and how you can achieve that. And how will the respective plotting package interpret your variable. It's a lot of fiddling, and that's normal. When I do this, there's typically a lot of trial and error, a lot of looking up help functions in R, and googling it – that is completely normal for any coding activity. So, don't be demotivated by many red error lines in your R console - that's part of the process!

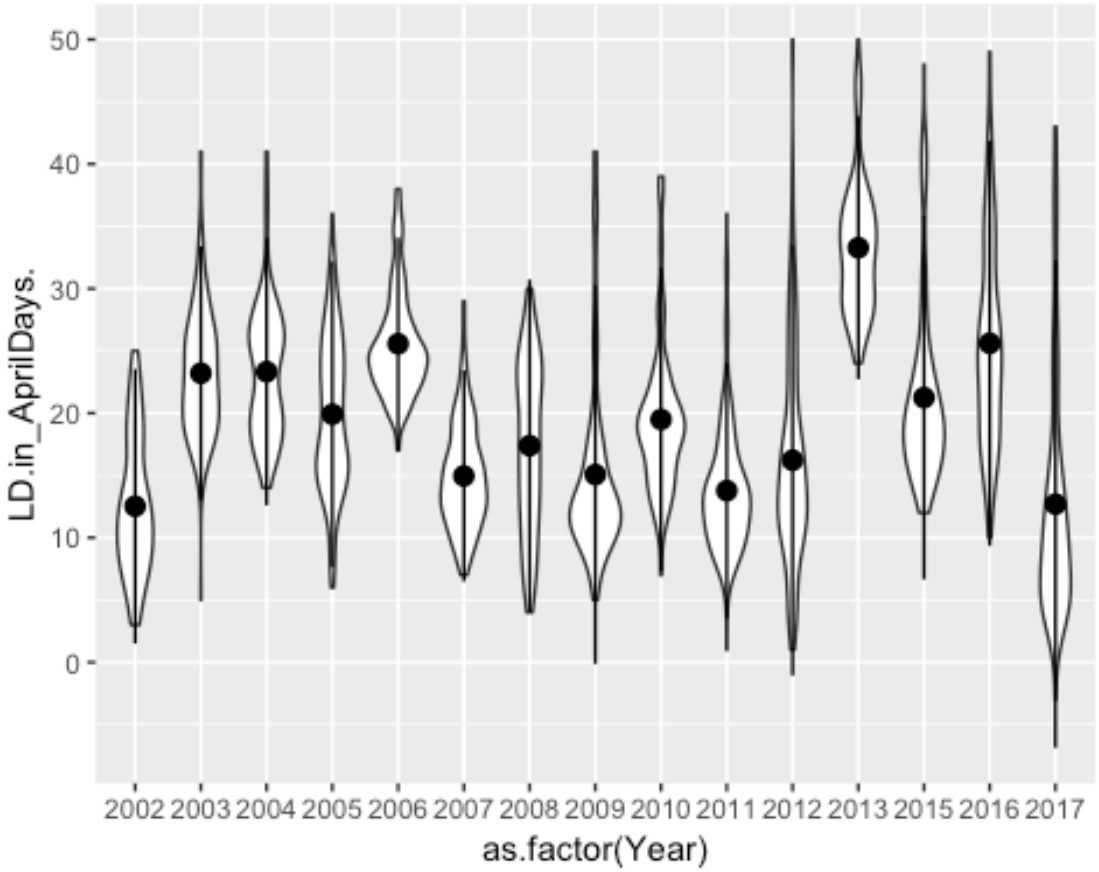
But I digress. Back to our visualisation with ggplot2! We wanted a nice violin plot. We can use `Year` as a factor (=categorical variable) here and then we get a nice plot:

```
p <- ggplot(b, aes(x=as.factor(Year), y=LD.in_AprilDays.)) +
    geom_violin()
p
```



That's much better! Now we can add some descriptive statistics (mean, and standard deviation):

```
p + stat_summary(fun.data="mean_sdl",
                 geom="pointrange")
```



Now. That's a nicer plot. We get descriptive statistics in it - we see the range of laying date in each year, we see how the data is distributed across days, and we can see where the mean is.

However, the take home from this is - when you run statistics, when you plot graphs, when you collect data - always have in mind whether your data is categorical or continuous, and what you need from it. Some continuous data can be treated as categorical - but rarely the other direction.

No excercises.