Stats with Sparrows - 13

Julia Schroeder

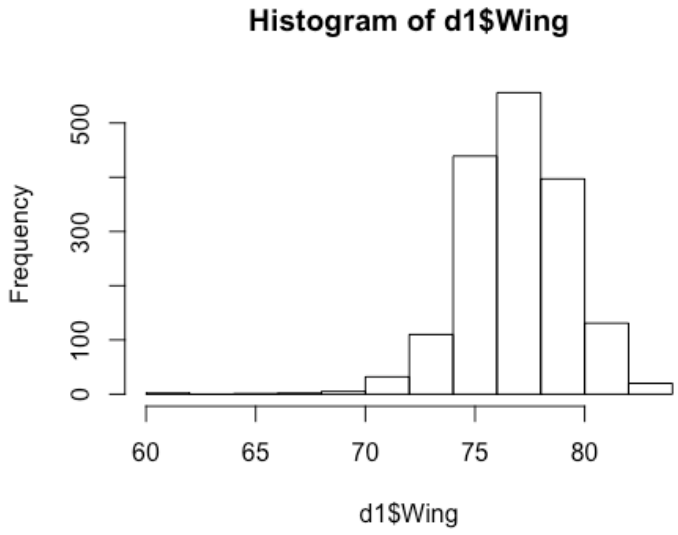
13

Again: housekeeping!

```
rm(list=ls())
setwd("H:/StatsWithSparrows")

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

We have understood linear models with one continuous explanatory variable. To go further from here, we move boldly towards the ANOVAs. ANOVAs test for differences between groups. They sound scary, but they aren't scary. They are actually easier things than linear models. They are a certain kind of linear models. And you've done one already. To make things more exciting, we use a new variable: wing length. Let's check it out first:

```
d1<-subset(d, d$Wing!="NA")
summary(d1$Wing)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    60.0    76.0    77.0    77.4    79.0    84.0

hist(d1$Wing)
```

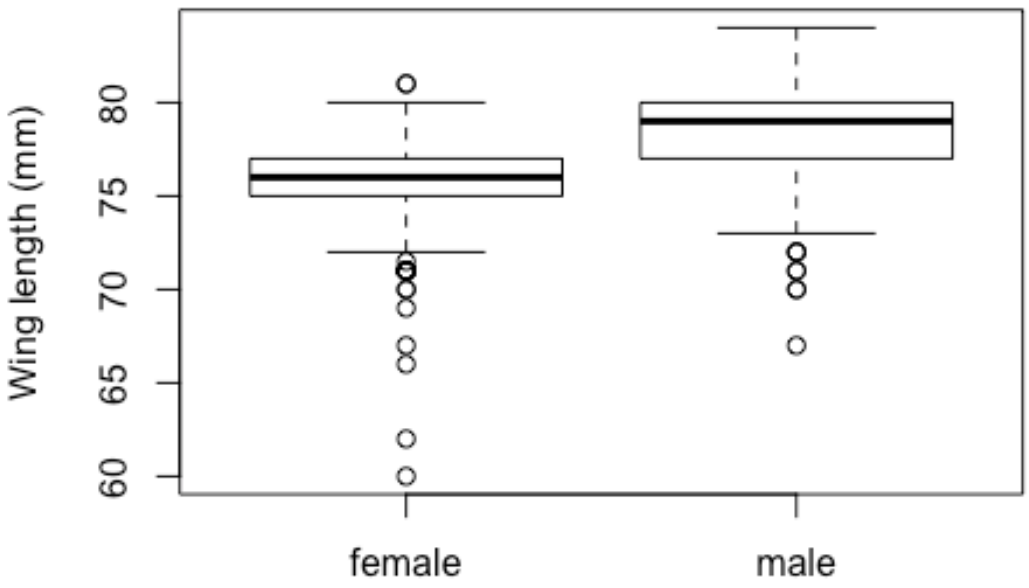


Ok, there are some missing values, and some "outliers". These likely come from young birds, or from birds that are moulting. We leave them in for now, but keep this in mind.

```r
model1<-lm(Wing~Sex.1,data=d1)
summary(model1)
```

```
##
## Call:
## lm(formula = Wing ~ Sex.1, data = d1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.0961  -1.0961  -0.0961   1.3683   5.3683
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 76.09611    0.07175 1060.50   <2e-16 ***
## Sex.1male    2.53562    0.09998   25.36   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.057 on 1693 degrees of freedom
## Multiple R-squared:  0.2753, Adjusted R-squared:  0.2749
## F-statistic: 643.1 on 1 and 1693 DF,  p-value: < 2.2e-16
```

```r
boxplot(d1$Wing~d1$Sex.1, ylab="Wing length (mm)")
```



Good, we understand most of these output. What don't we understand? There are the R-squares (we understand those), and the F-statistics. They actually come from an ANOVA that R has already performed for us. And the best thing is, we can access it.

```
anova(model1)
```

```
## Analysis of Variance Table
##
## Response: Wing
##             Df Sum Sq Mean Sq F value    Pr(>F)
## Sex.1        1 2722.0 2721.98  643.15 < 2.2e-16 ***
## Residuals 1693 7165.3    4.23
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All of the sudden, there are lots of squared stuff! All the sums of squares. Let's have a closer look. There are the sums of squares of the between-group variance (Sex.1), and those for the residuals. Then, there are the Mean squares for each of these two (within- and between-groups). The F-value, that is calculated by dividing the mean squares of the between-group estimate by the residual mean squares (or whithin-group mean squares). Then we ask R to up the p-value of the F-distribtuion. Thank you Ronald Fisher!

Ok, now we know that wing length differs between the two sexes. But, how much does it differ? Remember, effect size and all? To find that out, we actually have to do a t-test. Such a test that one runs after the main analysis, to see which group differs, and how much it differs, from another group, are called "post-hoc" tests.

```
t.test(d1$Wing~d1$Sex.1, var.equal=TRUE)
```

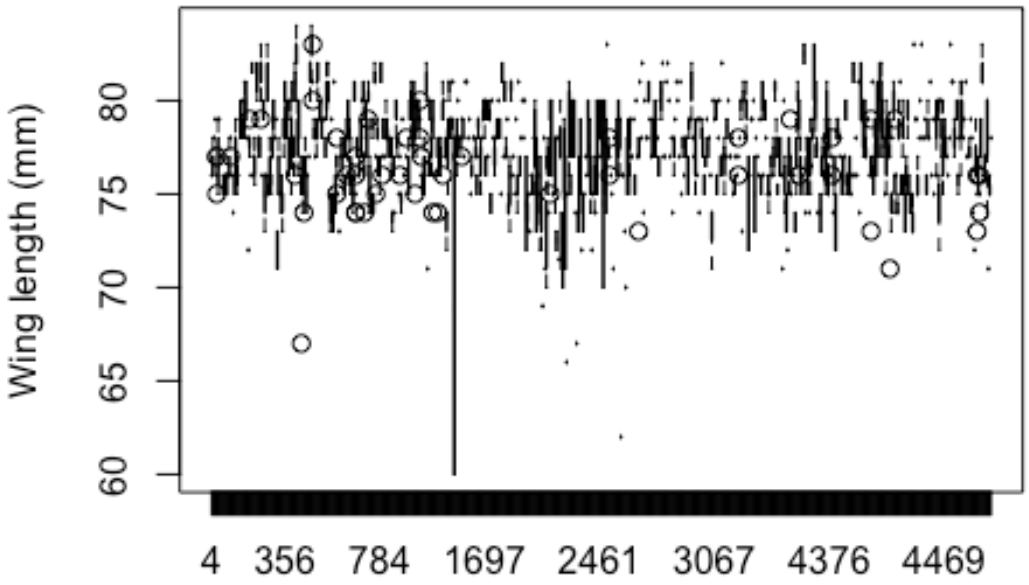
```
##
##  Two Sample t-test
##
## data:  d1$Wing by d1$Sex.1
## t = -25.36, df = 1693, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.731727 -2.339518
## sample estimates:
## mean in group female   mean in group male
##             76.09611             78.63173
```

This is ok for two groups. But what if we have lots of groups? Let's give this a try. Each sparrow is measured more than once. Now, it is interesting to know if each wing length measurement is the same for the same sparrow everytime it is caught, or whether the wing length changes over time. For instance, one could imagine that sparrows grow longer wing feathers when they get older. We can test that with an ANOVA - it tests, if we remember from earlier, whether the *variation within-groups is smaller than the variation among-groups*. In this case, group is birdID. Note that in english, the term *between* is only used for two groups, not for more, then we use *among* instead. Your task now is to run an ANOVA,

and then a linear model, with wing length as response, and BirdID as *factorial* explanatory factor. Create a box plot.

Uhhh. Here, we have loads and loads of groups. A couple hundred groups, actually. This is when an ANOVA can be really helpful. Let's see where this gets us:

```
boxplot(d1$Wing~d1$BirdID, ylab="Wing length (mm)")
```



Uhh, that's a lot! of individual birds. Can we get an idea of how many individual birds there are, and how often they are measured? There are several ways to do this, but here, we'll use dplyr

```
install.packages("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
```

```
tbl_df(d1)
```

```
## # A tibble: 1,695 × 12
##    BirdID Cohort CaptureDate CaptureTime  Year Tarsus  Bill  Wing  Mass
## *   <int>  <int>      <fctr>      <fctr> <int>  <dbl> <dbl> <dbl> <dbl>
## 1    4409   1994   23-Mar-00          NA  2000   18.0    NA    76  28.1
## 2    4402   1995   23-Mar-00          NA  2000   18.3    NA    80  29.8
## 3    4406   1995   23-Mar-00          NA  2000   18.8    NA    79  28.8
## 4    4407   1995   23-Mar-00          NA  2000   18.8    NA    79  29.3
## 5    4413   1999   23-Mar-00          NA  2000   19.3    NA    78  28.5
## 6    4414   1999   23-Mar-00          NA  2000   19.6    NA    81  32.1
## 7    4417   1999   23-Mar-00          NA  2000   18.8    NA    77  30.3
## 8    4437   1999   23-Mar-00          NA  2000   18.3    NA    73  28.3
## 9    4399   1995   24-Mar-00          NA  2000   20.0    NA    81  30.1
## 10   4406   1995   24-Mar-00          NA  2000   19.8    NA    79  29.8
## # ... with 1,685 more rows, and 3 more variables: Sex <int>, Sex.1 <fctr>,
## #   Observer <fctr>
```

```
glimpse(d1)
```

```
## Observations: 1,695
## Variables: 12
## $ BirdID      <int> 4409, 4402, 4406, 4407, 4413, 4414, 4417, 4437, 43...
## $ Cohort      <int> 1994, 1995, 1995, 1995, 1999, 1999, 1999, 1999, 19...
## $ CaptureDate <fctr> 23-Mar-00, 23-Mar-00, 23-Mar-00, 23-Mar-00, 23-Ma...
## $ CaptureTime <fctr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
## $ Year        <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 20...
## $ Tarsus      <dbl> 18.0, 18.3, 18.8, 18.8, 19.3, 19.6, 18.8, 18.3, 20...
## $ Bill        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## $ Wing        <dbl> 76, 80, 79, 79, 78, 81, 77, 73, 81, 79, 79, 76, 76...
## $ Mass        <dbl> 28.1, 29.8, 28.8, 29.3, 28.5, 32.1, 30.3, 28.3, 30...
## $ Sex         <int> 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,...
## $ Sex.1       <fctr> male, male, male, male, male, male, male, female,...
## $ Observer    <fctr> NO, NO, NO, NO, NO, NO, NO, NO, NO, NO, NO, NO, N...
```

Well, those are two helpful functions we commit to our memory for future use. Now let's get to the main task: we want to do some serious sub-totalling. What we need to do is group the data. Now, each row has one data entry. That means, some information in the data is repeated. For instance, when a bird is caught more than once, it gets more than one lines. We want to count how often birds where caught only once, twice, thrice ect. To do this, the best thing to do would be first to sort all data by BirdID, the identifier of individual birds. Then, we'd count how many rows each BirdID occupies. Then, we'd group them again by how many double, triple ect. observations we have, until we have a good summary that fits our needs. That sounds complicated. I'm sure you could do this with your magical skills you learned last week. But we'll try another way:

With dplyr, we can "pipe" objects to another function. That is super helpful when we don't want to write lengthy code with if's and for's:

```
d$Mass %>% cor.test(d$Tarsus, na.rm=TRUE)

##
##  Pearson's product-moment correlation
##
## data:  . and d$Tarsus
## t = 22.374, df = 1642, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4454229 0.5195637
## sample estimates:
##       cor
## 0.4833596
```

It takes the first object before the percentage sign (here
d

$Mass)andinsertsitasthefirstargumentintothefollowingfunction.Itevengetsinserted*before*d$
Tarsus.

Let's see what we can do with this:

```
d1 %>%
  group_by(BirdID) %>%
  summarise (count=length(BirdID))

## # A tibble: 618 × 2
##    BirdID count
##     <int> <int>
## 1       4     1
## 2      11    10
## 3      13     1
## 4      16     1
## 5      22     9
## 6      23     4
## 7      24     2
## 8      28     1
## 9      29     1
## 10     32     3
## # ... with 608 more rows
```

Ok - let's slow down. Here, we take our data d, and then group it by BirdID (the identifier for each individual bird). Then we take this data and pass it on to the function summarise. This is a super helpful function, it makes one row of many rows of data. Here, it takes each group of data on the same BirdID, and summarises it by counting how many lines are in this group.

The output is not very helpful. We see that well, bird 4 shows up 1 time in the dataset, bird 11 10 times, bird 13 only 1 time, and so forth. But it is not really helpful yet because we need to summarise this *again*, this time by count. Let's do that. I have to make a confession first: dplyr actually has a shorter way of doing this. I just didn't tell you right away because I wanted you to understand the group_by and summarise functions first. The shortcut to the above is this:

```
d1 %>%
  group_by(BirdID) %>%
  summarise (count=length(BirdID))
```

```
## # A tibble: 618 × 2
##    BirdID count
##     <int> <int>
## 1       4     1
## 2      11    10
## 3      13     1
## 4      16     1
## 5      22     9
## 6      23     4
## 7      24     2
## 8      28     1
## 9      29     1
## 10     32     3
## # ... with 608 more rows
```

```
count(d1, BirdID)
```

```
## # A tibble: 618 × 2
##    BirdID     n
##     <int> <int>
## 1       4     1
## 2      11    10
## 3      13     1
## 4      16     1
## 5      22     9
## 6      23     4
## 7      24     2
## 8      28     1
## 9      29     1
## 10     32     3
## # ... with 608 more rows
```

You can see, the output is exactly the same. Ok, now on to further summarising:

```
d1 %>%
  group_by(BirdID) %>%
  summarise (count=length(BirdID))  %>%
    count(count)
```

```
## # A tibble: 12 × 2
##    count     n
##    <int> <int>
## 1      1   222
## 2      2   147
## 3      3    88
## 4      4    48
## 5      5    47
## 6      6    26
## 7      7    15
## 8      8    10
## 9      9     6
## 10    10     7
## 11    11     1
## 12    12     1
```

You see, this gives us exactly the information we need. Why didn't I use the below code? It seems much more elegant?

```
count(d1, d1$BirdID) %>%
  count(count)
```

The reason is that we can't access the count variable here, because the name clashes with a function. There are ways around this, but for now, it is good to see that there are different ways of achieving things, and sometimes, knowing that helps. But now let's get back to stats. In our dataset of 1695 observations of wing length, 222 birds have been measured only once. So they are not very good data for getting within-group estimates. But 147 have been measured 2, and even more have been measured 3, 4, 5, up to 12 times! That is an amazing dataset of *repeated measures*. Let's run the ANOVA now, and let's not forget to tell R that we use BirdID as a factor!

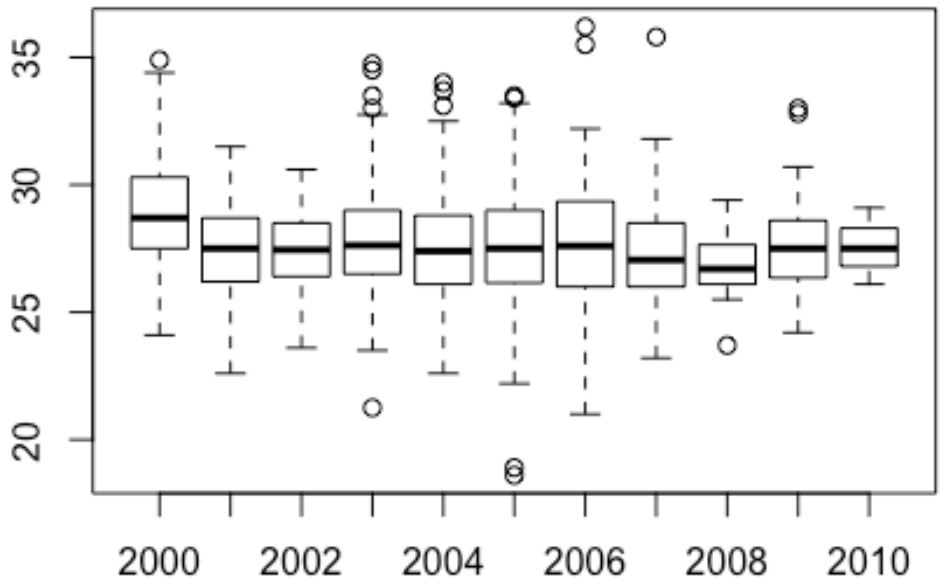
```
model3<-lm(Wing~as.factor(BirdID), data=d1)
anova(model3)
```

```
## Analysis of Variance Table
##
## Response: Wing
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## as.factor(BirdID)  617 8147.3 13.2047  8.1734 < 2.2e-16 ***
## Residuals         1077 1740.0  1.6156
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ok. Here we see clearly that there is statistically significantly more variation among groups (BirdID) than within-groups (residuals). The mean squares are much larger among groups than within. So here the difference between an ANOVA and a t.test becomes really clear: a t.test is about estimates, about the difference in size. An ANOVA is about the difference in variance among groups vs within groups. That's the important concept here to get clear with.

Now, I want to move back to linear models. When we ran the ANOVA with sex, we did post-hoc testing to ask what the parameter estimates were - how much did the groups differ in value, not variance? It would be impossible to test for BirdID, but if we have fewer groups, that is possible. Let's see how this pans out if we look at if birds caught in certain years were less heavy than in other years.

```
boxplot(d$Mass~d$Year)
```




```
m2<-lm(d$Mass~as.factor(d$Year))
anova(m2)
```

```
## Analysis of Variance Table
##
## Response: d$Mass
##                      Df Sum Sq Mean Sq F value    Pr(>F)
## as.factor(d$Year)    10  340.2  34.020  7.8866 1.721e-12 ***
## Residuals          1693 7303.0   4.314
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now, let's have a look at this output. We can tell right from the start that yes, there are differences between years - the between year variation is larger than the among-year varition. But between which years? There are 11 years! That's a lot of post-hoc t-tests.

There are issues with multiple testing that we've covered in the lecture. So how can we solve this? Well, let's remember to get the ANOVA output, we actually ran a linear model. So, let's have a look at the summary output from that one instead.

```
summary(m2)
```

```
##
## Call:
## lm(formula = d$Mass ~ as.factor(d$Year))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0051 -1.4051 -0.1089  1.2645  8.4874
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              28.8537     0.1375 209.772  < 2e-16 ***
## as.factor(d$Year)2001  -1.3620     0.2518  -5.410 7.22e-08 ***
## as.factor(d$Year)2002  -1.3871     0.2903  -4.778 1.92e-06 ***
## as.factor(d$Year)2003  -1.0596     0.2105  -5.035 5.30e-07 ***
## as.factor(d$Year)2004  -1.3182     0.1686  -7.816 9.49e-15 ***
## as.factor(d$Year)2005  -1.2486     0.1695  -7.367 2.71e-13 ***
## as.factor(d$Year)2006  -1.1411     0.2349  -4.858 1.29e-06 ***
## as.factor(d$Year)2007  -1.4176     0.2546  -5.568 2.99e-08 ***
## as.factor(d$Year)2008  -2.0810     0.6411  -3.246  0.00119 **
## as.factor(d$Year)2009  -0.9842     0.4544  -2.166  0.03046 *
## as.factor(d$Year)2010  -1.2871     1.2070  -1.066  0.28642
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.077 on 1693 degrees of freedom
##   (66 observations deleted due to missingness)
## Multiple R-squared:  0.04451,    Adjusted R-squared:  0.03887
## F-statistic: 7.887 on 10 and 1693 DF,  p-value: 1.721e-12
```

It does look a bit confusing at first, but this is exactly what we want. The difference to when we ran a lm with a continous explanatory variable (also named covariate), or with a two-level variable (remeber the sex example), is that this time, we have a factorial explanatory factor instaed. But, it's just ONE factor, why do we get so many lines then? The answer is that a lm uses a trick when it calculates estimates for factors. It actually recodes the data first, to each resemble a single, two-level factor (like in the sex example). R creates new variables (called dummy variables), but not one for each year, exactly *one less* than we have years. This might be confusing now but it will make sense to you soon. Each of these variables get a name - like, for example, "as.factor(d$Year)2001","as.factor(d$Year)2002", and so forth, until we reach "as.factor(d$Year)2010", the last year. Each of these variables contain zeroes, and ones. Each observation then gets 10 new columns, and there is a 1 only if the bird was measured in the respective year! This makes sense when you think about it: we can only calculate stuff using continous values. Years numbers, or names, or colors

(categorical or ordinal data) are really, really hard to interpret mathematically. So R has to use a "hack". We can have a look at how R does that, by looking at the design matrix:

```
t(model.matrix(m2))
```

```
##                         1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## (Intercept)             1 1 1 1 1 1 1 1 1  1  1  1  1  1  1  1  1  1  1  1
## as.factor(d$Year)2001 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2002 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2003 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2004 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2005 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2006 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2007 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2008 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2009 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2010 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
##                        21 22 23 24 25 26 28 29 30 31 32 33 34 35 36 37 38
## (Intercept)             1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## as.factor(d$Year)2001  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2002  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2003  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2004  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2005  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2006  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2007  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2008  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2009  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2010  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##                        39 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56
## (Intercept)             1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## as.factor(d$Year)2001  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2002  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2003  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2004  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2005  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2006  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2007  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2008  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2009  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## as.factor(d$Year)2010  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##                        57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 73 74
##
```

I DELETED LOTS OF STUFF HERE TO SAVE TREES (and your eyesight)

```
## (Intercept)             1    1    1    1    1    1    1    1    1    1
## as.factor(d$Year)2001   0    0    0    0    0    0    0    0    0    0
## as.factor(d$Year)2002   0    0    0    0    0    0    0    0    0    0
```

```
## as.factor(d$Year)2003    0    0    0    0    0    0    0    0    0    0
## as.factor(d$Year)2004    0    0    0    0    0    0    0    0    0    0
## as.factor(d$Year)2005    0    0    0    0    0    0    0    0    0    0
## as.factor(d$Year)2006    0    0    0    0    0    0    0    0    0    0
## as.factor(d$Year)2007    0    0    0    0    0    0    0    0    0    0
## as.factor(d$Year)2008    0    0    0    0    0    0    0    0    0    0
## as.factor(d$Year)2009    1    1    1    1    1    1    1    1    0    0
## as.factor(d$Year)2010    0    0    0    0    0    0    0    0    1    1
##                       1770
## (Intercept)              1
## as.factor(d$Year)2001    0
## as.factor(d$Year)2002    0
## as.factor(d$Year)2003    0
## as.factor(d$Year)2004    0
## as.factor(d$Year)2005    0
## as.factor(d$Year)2006    0
## as.factor(d$Year)2007    0
## as.factor(d$Year)2008    0
## as.factor(d$Year)2009    0
## as.factor(d$Year)2010    1
## attr(,"assign")
##  [1] 0 1 1 1 1 1 1 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$`as.factor(d$Year)`
## [1] "contr.treatment"
```

Here you can see a subset of the dummy variables, for some observations. Also, note how the intercept is always coded as 1. Why is that? This column is used to calculate the parameter estimates. Even in a linear model, we cannot put our *all possible* combinations of years, and compare them. That would be ludicrous. By convention, when computing linear models with fixed factors (factorial explanatory variables), we compute the parameter estimate for ONE level (here 2000). This is considered the *reference level*. The output for that is the mean body mass in the year 2000. For each other year, *R computes the difference to the reference level*. Let this sink in. That's super important. But, what about our linear model equation? It doesn't fit any longer, or, does it?

$$y_i = b_0 + b_1 x_2 + b_2 x_3 + b_3 x_3 \ldots \varepsilon_i$$

It can be expanded to fit more bs than just one! The one thing to remember here is that there is the one thing called "reference level" - the intercept, when running a model with categorical (factorial) explanatory variables is the 0th one. You cannot decide which one that is in R - R uses by default the lowest in alphanumerical order, or simply the one that comes first. It is something that you must write down in tables and results.

Let's inspect the summary output again:

```
summary(m2)
```

```
##
## Call:
## lm(formula = d$Mass ~ as.factor(d$Year))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0051 -1.4051 -0.1089  1.2645  8.4874
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              28.8537     0.1375 209.772  < 2e-16 ***
## as.factor(d$Year)2001  -1.3620     0.2518  -5.410 7.22e-08 ***
## as.factor(d$Year)2002  -1.3871     0.2903  -4.778 1.92e-06 ***
## as.factor(d$Year)2003  -1.0596     0.2105  -5.035 5.30e-07 ***
## as.factor(d$Year)2004  -1.3182     0.1686  -7.816 9.49e-15 ***
## as.factor(d$Year)2005  -1.2486     0.1695  -7.367 2.71e-13 ***
## as.factor(d$Year)2006  -1.1411     0.2349  -4.858 1.29e-06 ***
## as.factor(d$Year)2007  -1.4176     0.2546  -5.568 2.99e-08 ***
## as.factor(d$Year)2008  -2.0810     0.6411  -3.246  0.00119 **
## as.factor(d$Year)2009  -0.9842     0.4544  -2.166  0.03046 *
## as.factor(d$Year)2010  -1.2871     1.2070  -1.066  0.28642
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.077 on 1693 degrees of freedom
##   (66 observations deleted due to missingness)
## Multiple R-squared:  0.04451,    Adjusted R-squared:  0.03887
## F-statistic: 7.887 on 10 and 1693 DF,  p-value: 1.721e-12
```

In the year 2000 (the one that's missing in this list), the average body mass was 28.85 mm (SE 0.14). In 2001, the average mass was 1.36 g lighter, and the difference was statistically significant at p <0.001. And so forth. Interestingly, all years following 2000 have lighter birds, and it's more often than not more than one gramm. That seems quite systematical. Maybe people in 2000 used a different scale? So, is the difference mostly to 2000, and not among the other groups?

Excercise:

Run a linear model on a subset of data, excluding the year 2000. Examine both, the anova and summary output, and come to a conclusion about have a good think about biological relevance.