Lab for Lecture 16

Ethan Ashby

4/7/2020

Single Factor Type I ANOVA (part 2)

What we learned last time was that, with data coming from the model

$$y_{ij} = \mu_i + \epsilon_{ij}, \qquad i = 1, \ldots, k \qquad j = 1, \ldots, n_i$$

and assuming the same thing about the ϵ that we did in regression (independent, homoscedastic, and normal), we could ask the question

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_k \quad \text{vs} \quad H_a : \text{at least one is different}$$

by looking at how different the $\bar{y}_{i\cdot}$ were from each other, which we do by comparing to the overall mean $\bar{y}_{\cdot\cdot}$.

If they live close to the overall mean, we should be unwilling to conclude H_a. If they tend to live far from the overall mean, we should reject H_0.

How should we measure this dispersion from the overall mean? The math guides us. We have our familiar decomposition of the total sum of squares:

$$SSTO = SSE + SSTR$$

where

$$SSTR = \sum_i n_i (\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2$$

This does exactly what we were thinging, it's like the numerator of the variance of the group level averages, with the added n_i, which we might not have thought to put in there if we were constructing something based on the above idea ourselves.

It's great that it's in there though, because it causes this fact which we argued last time:

$$\text{Under } H_0 : \quad E(MSTR) = \sigma^2$$

$E(X)$ is another way of writing the mean, i.e. $E(X) = \mu_X$

We know that MSE does this as well, regardless of which hypothesis is true. Which is to say $E(MSE) = \sigma^2$ always. The degrees of freedom of MSE is $\sum n_i - k$ because we have a total of $\sum n_i$ points but need to estimate the k different $\bar{y}_{i\cdot}$ before we can think to estimate the deviations away from those means.

This then gave us our F-test, which was

$$F = \frac{MSTR}{MSE}$$

which should be about 1 if the null is true, and seemingly bigger than 1 if the alternative is true.

Let's try to argue this latest claim.

$$SSTR = \sum_i n_i (\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2$$

but under the alternative, this isn't a variance. The variance looks at how far random variables differ from their mean. But each of these has different means. Let's try to make this a variance:

$$\sum_i n_i (\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2 = \sum_i n_i (\bar{y}_{i\cdot} - \mu_i + \mu_i - \bar{y}_{\cdot\cdot})^2 \approx \sum_i n_i (\bar{y}_{i\cdot} - \mu_i)^2 + \sum_i n_i (\mu_i - \bar{y}_{\cdot\cdot})^2$$

Now that first term on the right looks like a proper variance. If we divide through by the degrees of freedom, we make that a variance, and the second term will, on average, be what you might expect

$$E(MSTR) = \sigma^2 + \frac{\sum_i n_i(\mu_i - \mu_\cdot)^2}{k-1}$$

where $\mu_\cdot$ is the average of the μ_i.

The way I think about this formula is the answer to the question: why aren't all the $\bar{y}_{i\cdot}$ the same?

(1) The data is noisy (this is the σ^2 term)
(2) They might be estimating different things. (this is the $\mu_i - \mu_\cdot$ term)

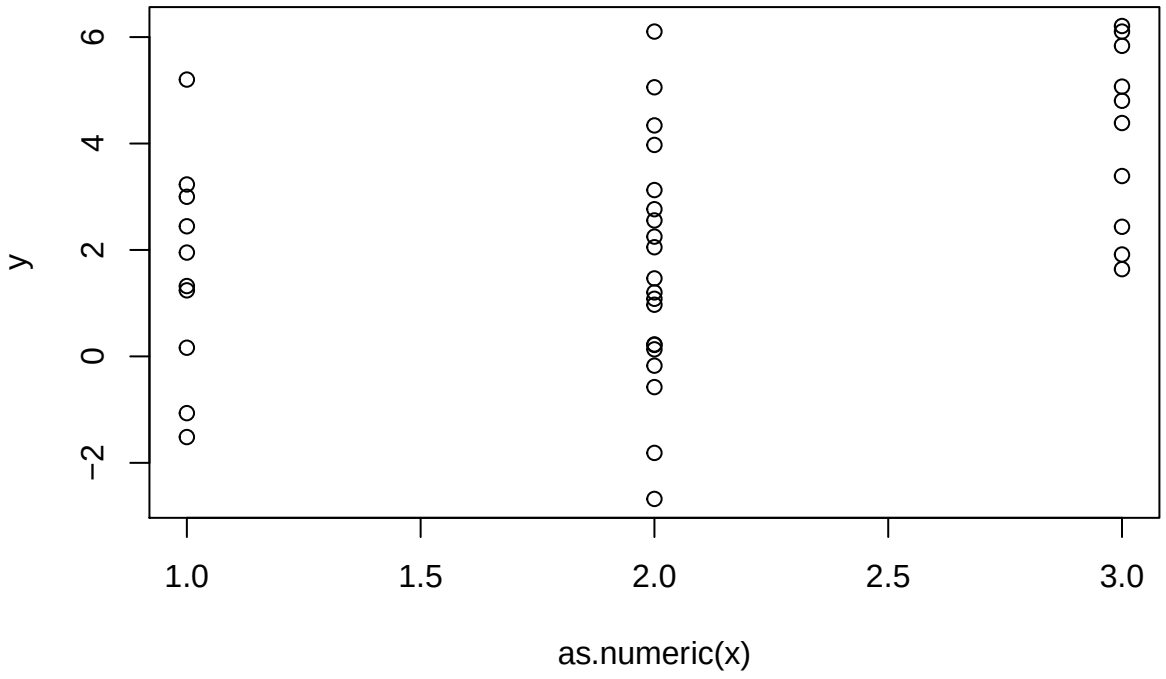
And then you can see where the rest of the constants come from via the algebra.

Let's verify this result via simulation.

```
mus <- c(1,2,3)
n1 <- 10; n2 <- 20; n3 <- 10
sig <- 2
```

So from our formula: $E(MSTR) = 2^2 + \frac{10(1-2)^2+20(2-2)^2+10(3-2)^2}{3-1} = 14$

```
y <- c(mus[1]+rnorm(n1, 0, sig), mus[2]+rnorm(n2, 0, sig), mus[3]+rnorm(n3, 0, sig))
x <- as.factor(c(rep('A', n1), rep('B', n2), rep('C', n3)))
plot(as.numeric(x), y)
```




```
anova(aov(y~x))$Mean[1]
```

```
## [1] 24.77318
```

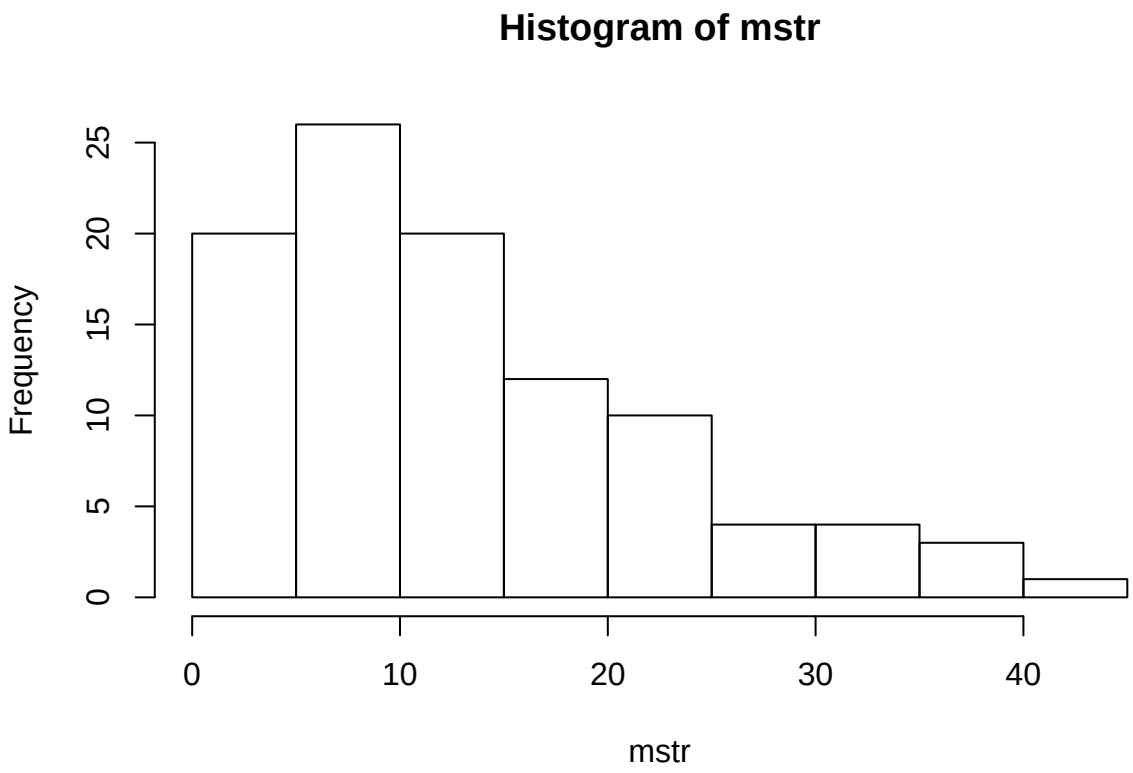
That of course, is just one time, $E(MSTR)$ is the average. Let's try it a bunch.

```
mstr <- c()
for (i in 1:100) {
  y <- c(mus[1]+rnorm(n1, 0, sig), mus[2]+rnorm(n2, 0, sig), mus[3]+rnorm(n3, 0, sig))
  mstr[i] <- anova(aov(y~x))$Mean[1]
```

```
}
mean(mstr)
```

```
## [1] 13.45235
```

```
hist(mstr)
```



This is only 100 simulations, if you did more, you would tend to get something closer to 14.

So this second part of $E(MSTR)$ controls the expected size of the F statistic. Since you will reject when F is large, you are more likely to do that when

(1) the true means are more spread out and

(2) your sample sizes are bigger.

This is the quantity that controls the power of the test. You may also notice that the second term, for a fixed size, will dominate the F statistic more if

(3) σ^2 is small.

since

$$E(\frac{MSTR}{MSE}) \approx \frac{\sigma^2 + \frac{\sum_i n_i(\mu_i - \mu_.)^2}{k-1}}{\sigma^2}$$

As $\sigma^2 \to 0$, this goes to ∞, and as $\sigma^2 \to \infty$ this goes to 1.

Right now, it seems like (2) is the only thing we have control over, the sample sizes. So as far as designing a good experiment, make n_i large if you can, or large enough to get a suitable power at least. We'll take a look at (3) later. (1) is a lost cause (design better drugs and test those?)

Checking Assumptions

As we discussed, there are assumptions on the error terms, and they are identical to the regression assumption (minus the linearity assumption, but we still assume $\mu_\epsilon = 0$). How should we check these assumptions? Residual plots you say? Great, moving on.

Global vs Local Analysis

I asked a sneaky question at the end of the last lab. Is using the F statistic the right way to choose a variable for doing classification? While this is the basis of Linear Discriminant Analysis (LDA), the answer is no.

A large F value tells you that the $\bar{y}_{i\cdot}$ tend to be different, but it doesn't exclude the possibility that some of the $\bar{y}_{i\cdot}$ are very close to each other. We only know that this doesn't happen globally. At least one thing is off. But if I'm doing classification, I'd perfer the variable that has distinct $\bar{y}_{i\cdot}$ for every class of Iris. These are different concerns.

This class isn't about classification, but this brings up a valuable point. Just because I reject the null hypothesis, I don't necessarily know why I did it. The same was true with regression. We had $F = \frac{MSR}{MSE}$ which tested if any of the information I had in my data was valuable, but rejecting this null didn't tell me which variables were important. We had a lot of work to do still. We do here as well.

We need to be careful here. Statistical hypothesis testing usually allows an error rate of .05 for type I errors, which is to say that, if the null is true, about once every 20 times, we will conclude incorrectly that the alternative is true. But this holds for a single test. If we do lots of test (like say over a scientific career), the probability that we make this mistake at some point increases.

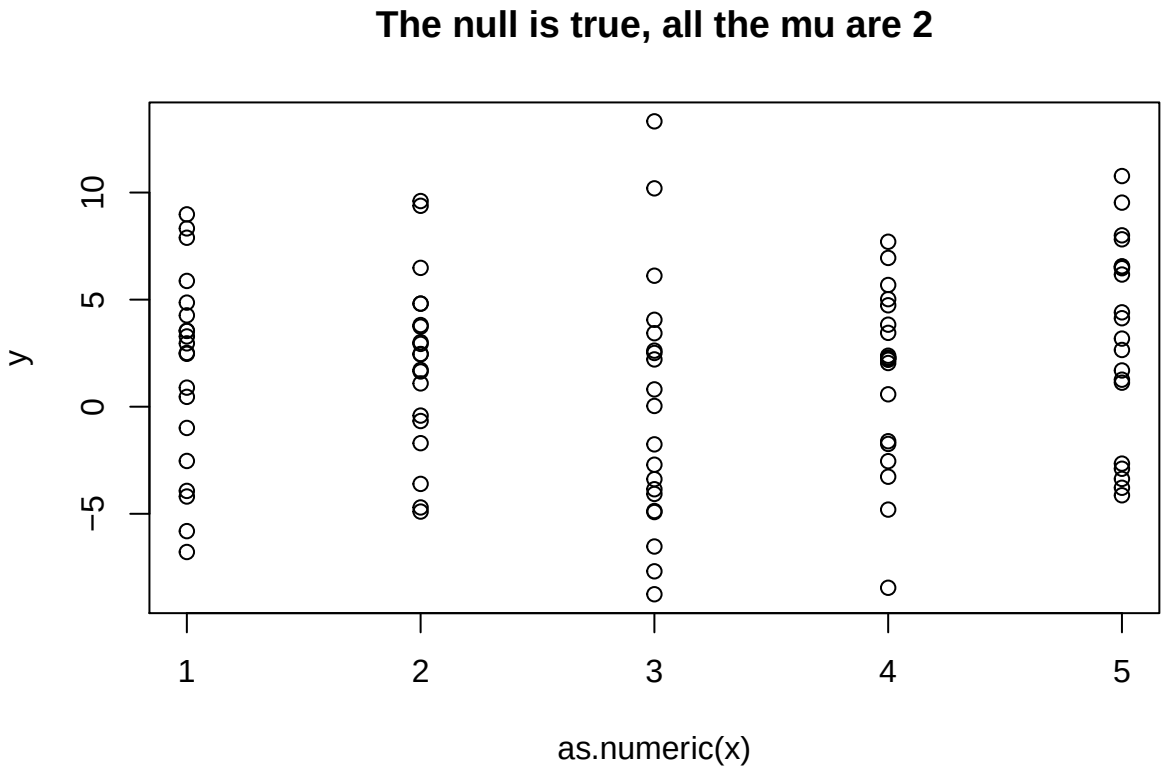
But this shouldn't happen in the course of a single experiment.

What we are thinking about now is, supposing we have rejected the null, trying to figure out why we rejected it. Which of those means μ_i are different from others?

So we are going to be asking follow up questions like $H_0 : \mu_i = \mu_{i'}$ How many of those will we ask? It's looking like $\binom{k}{2}$. Let's try it.

I'll generate some data under the null:

```
mus <- c(2,2,2,2,2)
y <- 2 + rnorm(100, 0, 5)
trt <- c('a','b','c','d','e')
x <- as.factor(rep(trt, 20))
plot(as.numeric(x), y, main="The null is true, all the mu are 2")
```



But let's ask the $\binom{5}{2} = 10$ individual questions

```
ps <- c()
k <- 1
for (i in 1:4) {
  for (j in (i+1):5) {
    ps[k] <- t.test(y[x==trt[i]], y[x==trt[j]])$p.val
    k <- k + 1
  }
}
ps
```

```
##  [1] 0.81769411 0.25001633 0.81094770 0.34860420 0.16285015 0.61608521
##  [7] 0.44082102 0.31959165 0.05335361 0.22245303
```

Hopefully we didn't reject the null hypothesis (since the null is true every one of those 10 times). Let's see how often we would find something interesting.

```
set.seed(47)
type1 <- 0
for (m in 1:100) {
  y <- 2 + rnorm(100, 0, 5)
  ps <- c()
  k <- 1
  for (i in 1:4) {
    for (j in (i+1):5) {
      ps[k] <- t.test(y[x==trt[i]], y[x==trt[j]])$p.val
      k <- k + 1
    }
  }
  type1 <- type1 + (min(ps)<.05)
}
```

```
type1/100
```

```
## [1] 0.25
```

Uh oh. A quarter of these terrible science experiments where all the scientific ideas are wrong just resulted in publishable results!

We should fix this.

One way we fix this is by only doing local analysis if the global analysis calls for it. So ANOVA has already fixed part of the problem. But there might be lots of local tests for which the null is true, even if the F-test correctly told us something interesting was happening. We want to make sure we don't screw this part up.

An easy way to fix this: Bonferroni's Inequality. We are going to prove a special case of this using basic probability that likely gets covered in an intro class.

We are going to do two tests. The null is true for both tests. We know nothing about how they relate to each other (maybe they involve some of the same variables?)

Let A_i be the event that we make a type I error on the i^{th} test. So $P(A_i) = \alpha_i$. What might we choose α_i to be?

I want to avoid the situation above where I end up finding something pretty often. I'd like it to be the case that I only find something, anything, 5 percent of the time.

$$.05 = P(A_1 \text{ or } A_2) = P(A_1) + P(A_2) - 2P(A_1 \text{ and } A_2) \leq P(A_1) + P(A_2) = \alpha_1 + \alpha_2$$

I used the fact that probabilities are non-negative to get the inequality.

This tells me that if I use $\alpha_i = .025$ for each test, I will be ok. (or better, construct 97.5 percent CIs)

The general formula says that if you are going to do m tests, use $.05/m$ for each one (or $(1 - .05/m)100\%$ confidence intervals).

Let's try it on the experiment above.

```
set.seed(47)
type1 <- 0
for (m in 1:100) {
  y <- 2 + rnorm(100, 0, 5)
  ps <- c()
  k <- 1
  for (i in 1:4) {
    for (j in (i+1):5) {
      ps[k] <- t.test(y[x==trt[i]], y[x==trt[j]])$p.val
      k <- k + 1
    }
  }
  type1 <- type1 + (min(ps)<(.05/10))  #this is the only change made, using .05/10 for alpha
}
type1/100
```

```
## [1] 0.04
```

Virtual high fives all around.

This is a pretty blunt tool, it doesn't realize that there is a lot of correlation in these 10 tests (each variable shows up in 4 of them). In ANOVA we can do better. This will be using something called Tukey's Honest Significant Difference (it will still do the controlling of the overall error rate at .05 or whatever you want, but will do it better - smaller confidence intervals or more power, your choice).

We won't talk about where this comes from today, come back next time, but I will give you code to generate them.

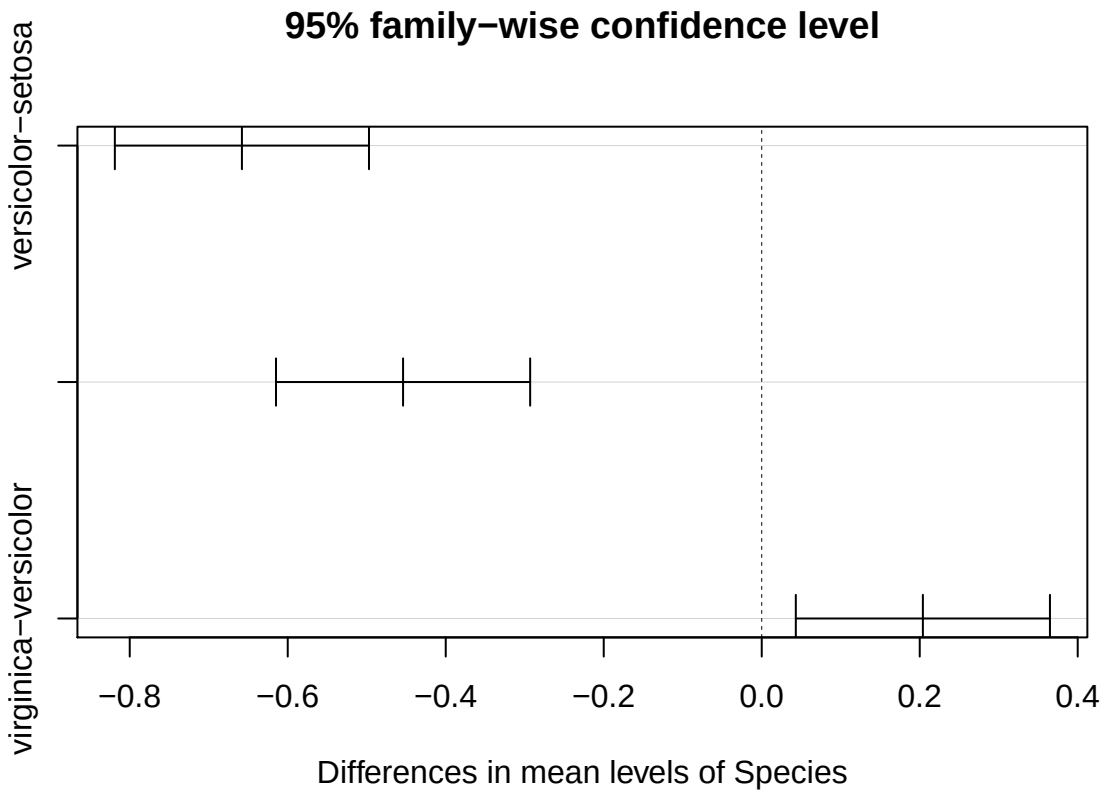
```
data(iris)
fit <- aov(Sepal.Width ~ Species, data=iris)
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: Sepal.Width
##            Df Sum Sq Mean Sq F value    Pr(>F)
## Species     2 11.345  5.6725   49.16 < 2.2e-16 ***
## Residuals 147 16.962  0.1154
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(fit)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Sepal.Width ~ Species, data = iris)
##
## $Species
##                         diff         lwr        upr     p adj
## versicolor-setosa    -0.658 -0.81885528 -0.4971447 0.0000000
## virginica-setosa     -0.454 -0.61485528 -0.2931447 0.0000000
## virginica-versicolor  0.204  0.04314472  0.3648553 0.0087802
```

```
plot(TukeyHSD(fit))
```



Try this on a data set my former 58 students might remember. A group of of 12-13 y/o boys were given a video game to

play. The only thing that changed between the various treatments was the age recommendation on the box. The game was always the same. They were asked, on a scale of 1-10, to rate the game.

```
ratings <- c(6, 6, 6, 5, 4, 8, 6, 1, 2, 4, 8, 7, 8, 5, 7, 9, 5, 8, 4, 7,
 7, 9, 8, 6, 7, 4, 8, 9, 6, 7, 10, 9, 6, 8, 7, 6, 8, 9, 10, 8)
age <- c(rep('7+', 10), rep('12+', 10), rep('16+', 10), rep('18+', 10))
videogame <- data.frame(rating=ratings, age=as.factor(age))
head(videogame)
```

```
##   rating age
## 1      6  7+
## 2      6  7+
## 3      6  7+
## 4      5  7+
## 5      4  7+
## 6      8  7+
```

Run an analysis and interpret the results in the context of the experiment at hand.

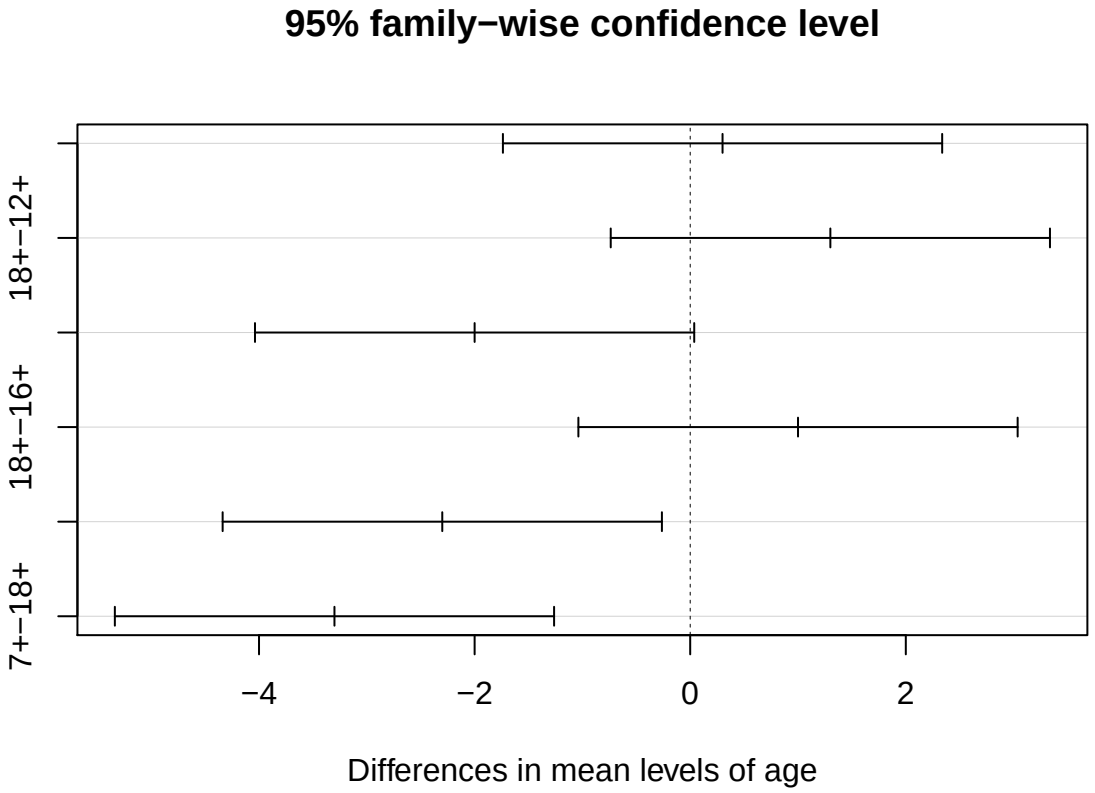
```
fit <- aov(rating~ age, data=videogame)
anova(fit)
```

```
## Analysis of Variance Table
##
## Response: rating
##           Df Sum Sq Mean Sq F value   Pr(>F)
## age        3   57.4 19.1333  6.6874 0.001053 **
## Residuals 36  103.0  2.8611
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(fit)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = rating ~ age, data = videogame)
##
## $age
##         diff        lwr         upr     p adj
## 16+-12+  0.3 -1.7373017  2.33730169 0.9785365
## 18+-12+  1.3 -0.7373017  3.33730169 0.3291989
## 7+-12+  -2.0 -4.0373017  0.03730169 0.0559407
## 18+-16+  1.0 -1.0373017  3.03730169 0.5553447
## 7+-16+  -2.3 -4.3373017 -0.26269831 0.0217743
## 7+-18+  -3.3 -5.3373017 -1.26269831 0.0005757
```

```
plot(TukeyHSD(fit))
```



There is a significant difference between ratings for kids who were told the game was for 7 y/o and 16 or 18 y/o. On average, when kids were told that a game was for 16 year olds, they on average rated the game 2.3 points higher. When they were told the game was for 18 year olds, they on averge rated the game 3.3 points higher.