جامعة نيويورك أبوظبي

NYU | ABU DHABI

PSYCH-UH 1004Q: Statistics for Psychology

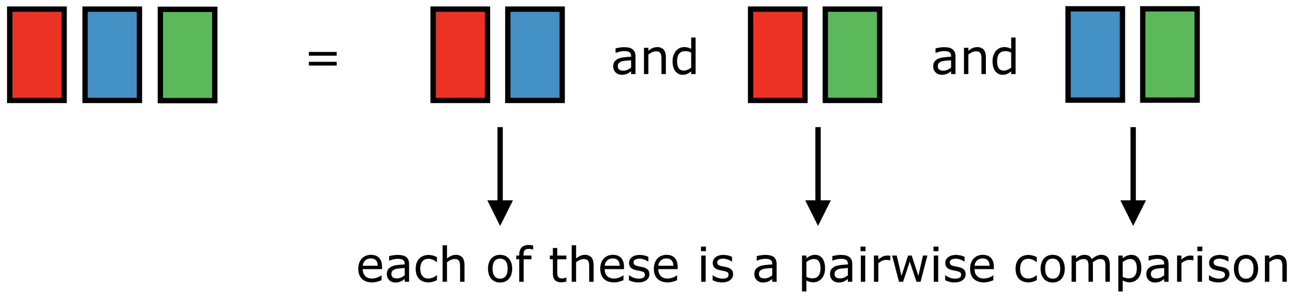
Class 21: Multiple comparisons - Dunn’s correction and Tukey’s HSD

Prof. Jon Sprouse
Psychology

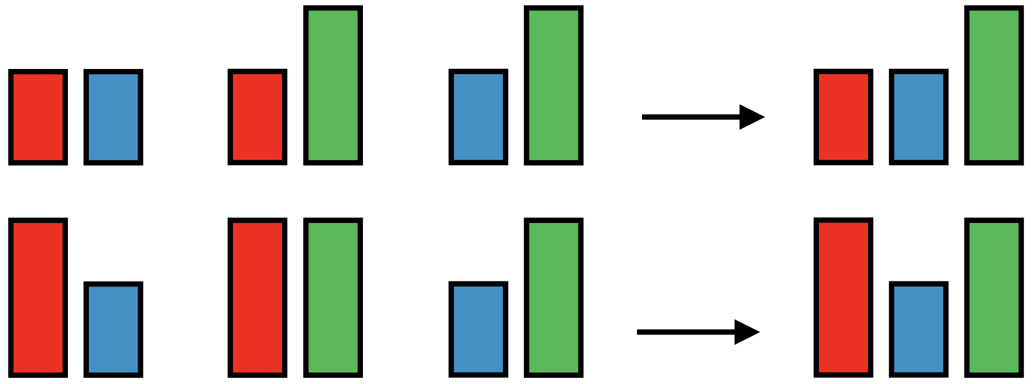
Quick recap of the logic of the multiple comparisons problem

Multiple pairwise comparisons

When we compare two conditions to each other, we call it a **pairwise comparison.** With three conditions, we have three possible comparisons:



When H_0 is false, a very logical approach to figuring out how the conditions differ from each other would be to perform a pairwise comparison for each possible pair of conditions. This will show you which of these patterns you have:



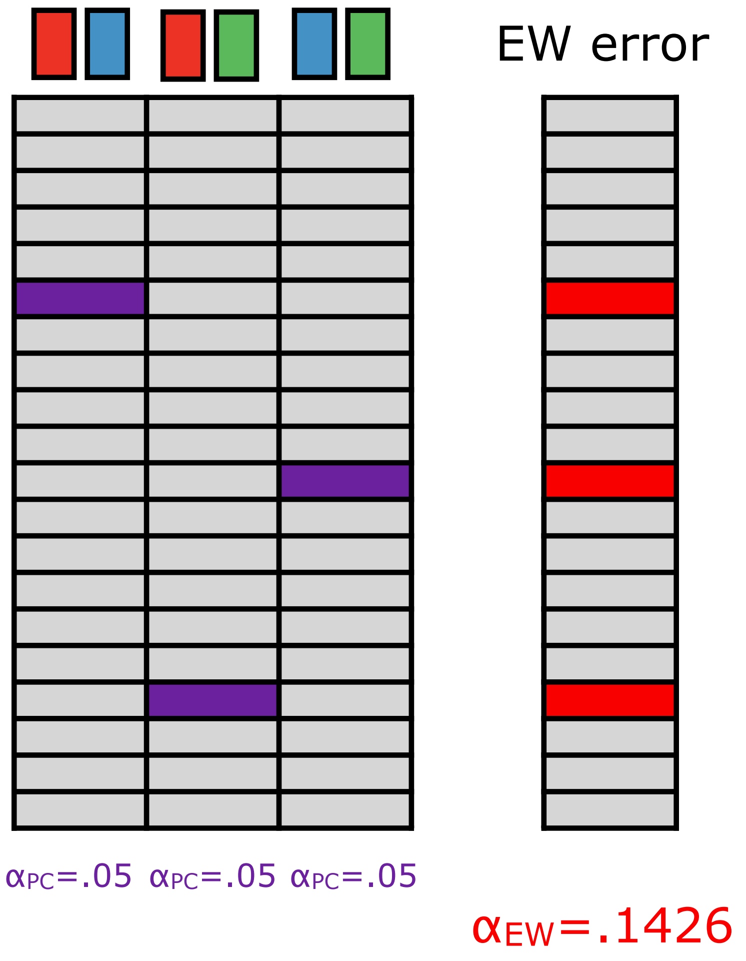
This is just two examples. But you can see how running each of the pairwise comparisons will show you the full pattern. So, this is something you will want to do!

We call this the multiple comparisons problem

When an experiment contains multiple comparisons (that is, more than one comparison), the α_{EW} will be higher than α_{PC}.

The issue is that α_{EW} is probably the error rate you care about when you have an experiment with multiple comparisons. You don't want to report that your experiment found an effect when it did not. Imagine you are testing 3 medicines. You don't want to say that one of them worked when it did not!

Basically, the multiple comparisons problem is that you think you are keeping your error rate low (e.g., .05), but when you are reporting multiple comparisons, the error rate that you care about is α_{EW} and it is higher than .05.



How big of a problem is it?

We can actually calculate the α_{EW} based on the α_{PC} that you choose and the number of comparisons:

$$\alpha_{EW} = 1 - (1 - \alpha_{PC})^{C}$$

Deriving this equation is not straightforward because is the probability for at least 1 (so 1, 2, 3, etc) error. But one way to think about it is to flip it around.

The probability of making 0 errors is easy to calculate. For one comparison it is $1-\alpha_{PC}$. So, typically .95.

For two comparisons, the probability of making 0 errors is .95 x .95 (because that is how joint probability works for independent events).

For three comparisons, the probability of making 0 errors is .95 x .95 x .95.

We can generalize this to say that the probability of making 0 type I errors is $(1-\alpha_{PC})^{C}$. Therefore the probability of not-getting-0, which means at least 1, is the inverse:

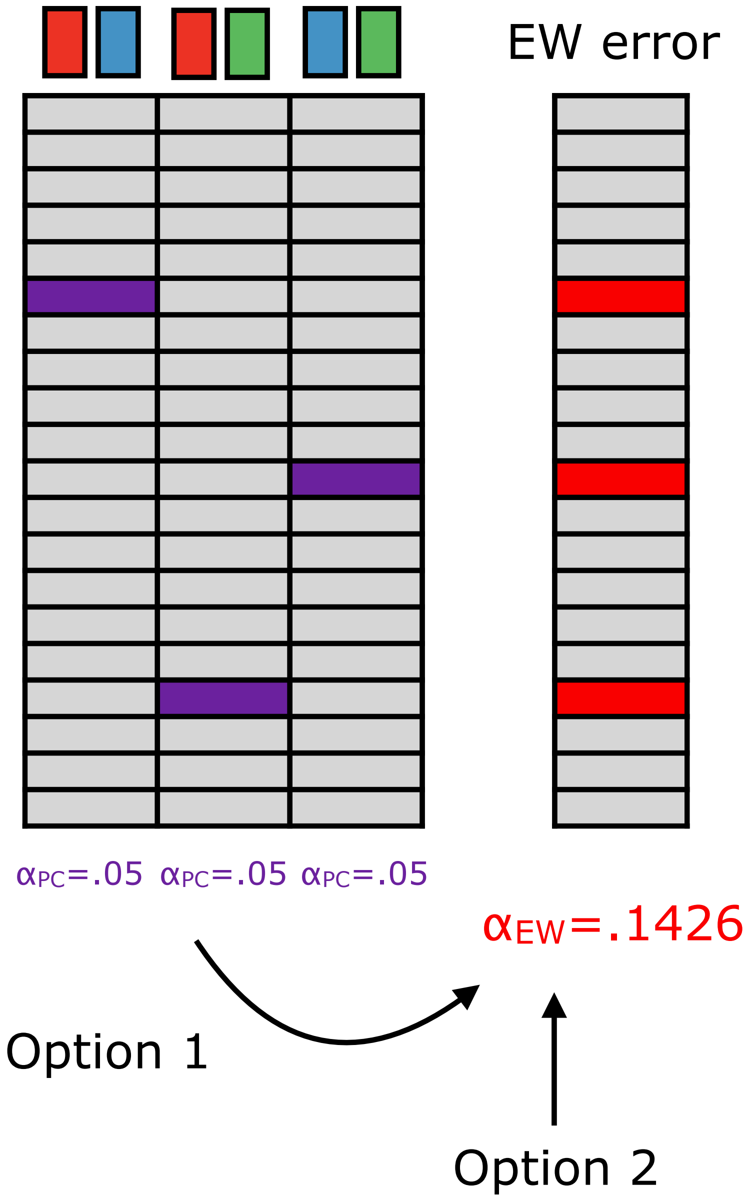
$$\alpha_{EW} = 1 - (1 - \alpha_{PC})^{C}$$

Two approaches to addressing this

Option 1: If there is a predictable relationship between α_{PC} and α_{EW}, then we can set α_{PC} to the rate that will give us the α_{EW} that we want. This what the **Dunn (a.k.a. Bonferroni) correction** does.

$$\alpha_{EW} = 1 - (1 - \alpha_{PC})^{C}$$

Option 2: We can use α_{EW} as a criterion directly, and set that to .05. This means deriving a new test statistic based on α_{EW}. This is what **Tukey's Honestly Significant Difference** does.



Dunn’s correction

(a.k.a. Bonferroni correction)

Selecting α_{PC} to control α_{EW}

The Dunn (a.k.a. Bonferroni) correction is straightforward to apply. You simply change the alpha criterion that you use to claim that a result is statistically significant.

$$\underbrace{\alpha_{EW}}_{\text{to control this}} = 1 - (1 - \underbrace{\alpha_{PC}}_{\text{select this}})^{C}$$

Notice that with 1 comparison, α_{PC} equals α_{EW}:

$$\alpha_{EW} = 1 - (1 - \alpha_{PC})^{1}$$

$$\alpha_{EW} = \alpha_{PC}$$

But how do we choose the right α_{PC}? Well, it turns out that there is a very simple way to do this. We simply divide α_{EW} by the number of contrasts:

$$\text{alpha criterion} = \frac{\alpha_{EW}}{C}$$

So, if you have 3 contrasts, and if you want an α_{EW} of .05, you simply divide .05 by 3. Then you use the resulting number as your alpha-criterion for deciding statistical significance.

for an α_{EW} of .05: $\dfrac{.05}{3} = .0167$

Why does it work?

The Dunn (a.k.a. Bonferroni) correction works because the following inequality is always true:

$$X \geq 1 - (1-(X/C))^C$$

This inequality was discovered by Carlo Bonferroni. Take a look at it. It tells us that the quantity on the right will always be less than or equal to the quantity on the left. This is simply a mathematical fact.

We can plug α_{EW} into this inequality, and it tells us that our actual α_{EW} (on the right) will always be less than our desired α_{EW}.

$$\text{desired } \alpha_{EW} \geq 1 - (1-(\alpha_{EW}/C))^C$$

$$.05 \geq 1 - (1-(.05/C))^C$$

Now, we can notice that the quantity on the right looks just like the equation for experimentwise error, except α_{PC} has been replaced with α_{EW}/C. So, we can see that using α_{EW}/C as our alpha-criterion will give us experimentwise error that is <u>less than or equal to</u> α_{EW}.

$$\alpha_{EW} = 1 - (1 - \alpha_{PC})^C$$

Demonstrating that the Dunn correction works

Let’s do the same simulation from before with 3 pairwise comparisons per experiment. We know that it will yield an error rate near .14. But this time, we will use the Dunn correction to set our alpha level to .0167. If the Dunn correction works, we should see an error rate near .05.

Step 1: Simulate 10,000 experiments with 3 comparisons.

```
boot.replicates=replicate(10000, expr=rnorm(150, mean=0, sd=1))
```

Step 2: Write a function to calculate 3 t-tests.

```
t.function <- function(dataset){
  t1=t.test(dataset[1:50], dataset[51:100], var.equal=T)
  t2=t.test(dataset[1:50], dataset[101:150], var.equal=T)
  t3=t.test(dataset[51:100], dataset[101:150], var.equal=T)
  return(c(t1$p.value,t2$p.value,t3$p.value))
}
```

Step 3: Apply the function to the simulated experiments.

```
experiments = apply(X=boot.replicates, MARGIN=2, FUN=t.function)
```

Demonstrating that the Dunn correction works

Step 4: Write two functions. One that looks for errors at *p*<.05, and one that looks for errors at *p*<.0167 (the Dunn corrected alpha level).

```
error.uncorrected <- function(dataset){
    errors=sum(dataset<.05)
    return(errors>0)
}
```

```
error.corrected <- function(dataset){
    errors=sum(dataset<.0167)
    return(errors>0)
}
```

Step 5: Apply the function to simulated *t*-test results.

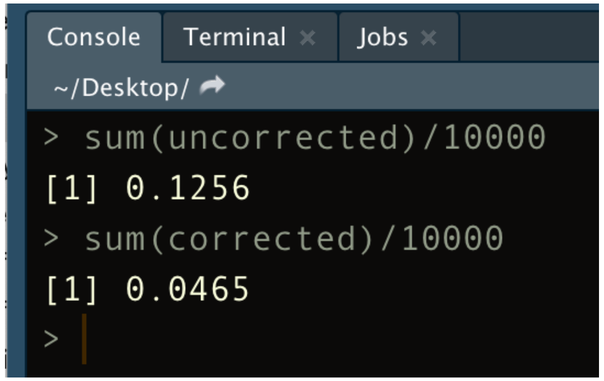
```
uncorrected=apply(X=experiments, MARGIN=2, FUN=error.uncorrected)

corrected=apply(X=experiments, MARGIN=2, FUN=error.corrected)
```

Step 6: Count the errors for each method.

```
sum(uncorrected)/10000
sum(corrected)/10000
```

And we see that the corrected error rate is approximately .05, just like we want!



```
> sum(uncorrected)/10000
[1] 0.1256
> sum(corrected)/10000
[1] 0.0465
>
```

A note on the name: Dunn vs Bonferroni

The most common name for this method is the Bonferroni correction. 99% of people will use that name. Carlo Bonferroni (1892-1960) was a male mathematician who studied various inequalities as part of his mathematical research. **He did not work on statistics (as far as I know)**.

I, and many other people, think it should be called the Dunn Correction. **Olive Dunn** (1915-2008) was a statistician who worked out the mathematical properties of the Dunn correction in a 1961 paper. What I cannot find out is if she was the first to propose the correction or not. What is clear is that Carlo Bonferroni did not propose it, because he didn't work on statistics. He was a pure mathematician.

If Olive Dunn invented the test, then it should be named after her. All statistical tests are named after the statisticians who discovered them, including correction procedures: Tukey, Scheffe, Fisher, etc. We never name things after the mathematicians who worked out the math that is used in the tests (This is often calculus, so it would be Euler, Leibniz, etc).

If she didn't invent it, but just worked out its properties, then it could still be named after her, or perhaps hyphenated: Dunn-Bonferroni (like the Holm-Bonferroni method). I suspect the fact that it is not is because of sexism.

The advantages of Dunn’s correction

Advantage 1: It definitely works. This is obvious from the equations. But you can also test it for yourself by modifying the R code in the previous slides to test any number of *t*-tests.

$$\alpha_{EW} \geq 1 - (1-(\alpha_{EW}/C))^{C}$$

$$\updownarrow$$

$$\alpha_{EW} = 1 - (1 - \alpha_{PC})^{C}$$

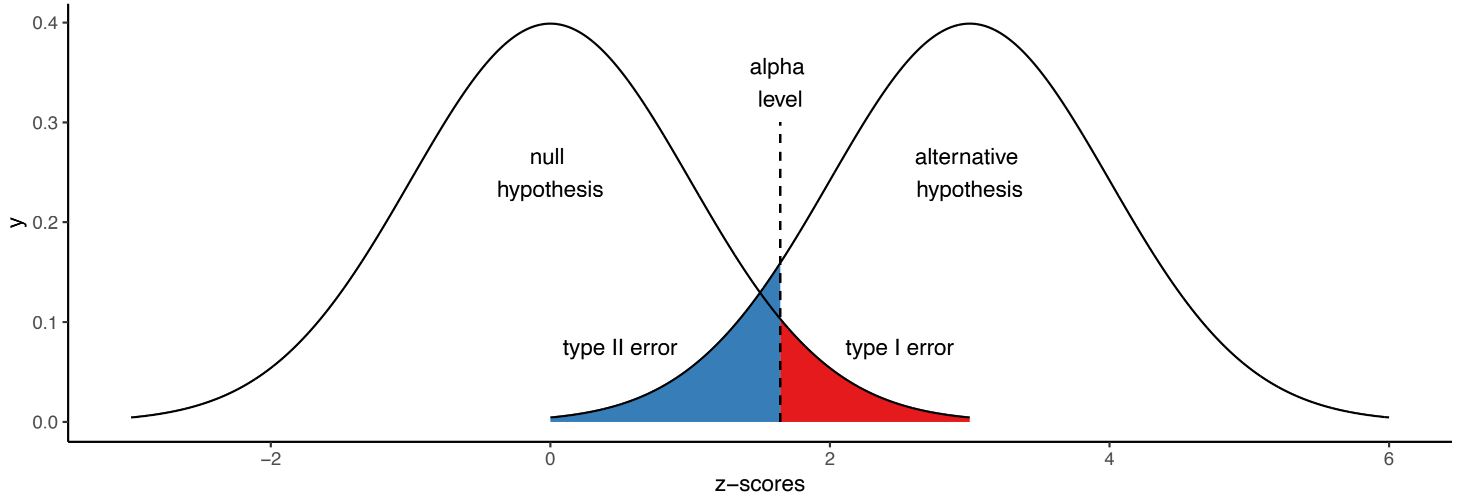
Advantage 2: It can be used for any type of test. You are most likely to use it with *t*-tests, but you can also use it with any other statistical test. This is because it works directly with *p*-values, not test statistics, so it is very general. It can be used with any null hypothesis test that yields a *p*-value. (Dunn also showed that you can extend it to confidence intervals.)

Advantage 3: It is conceptually straightforward, so it is widely accepted as a method. Reviewers and readers will definitely agree that it adequately controls the experimentwise error rate. (You should cite Dunn 1961.)

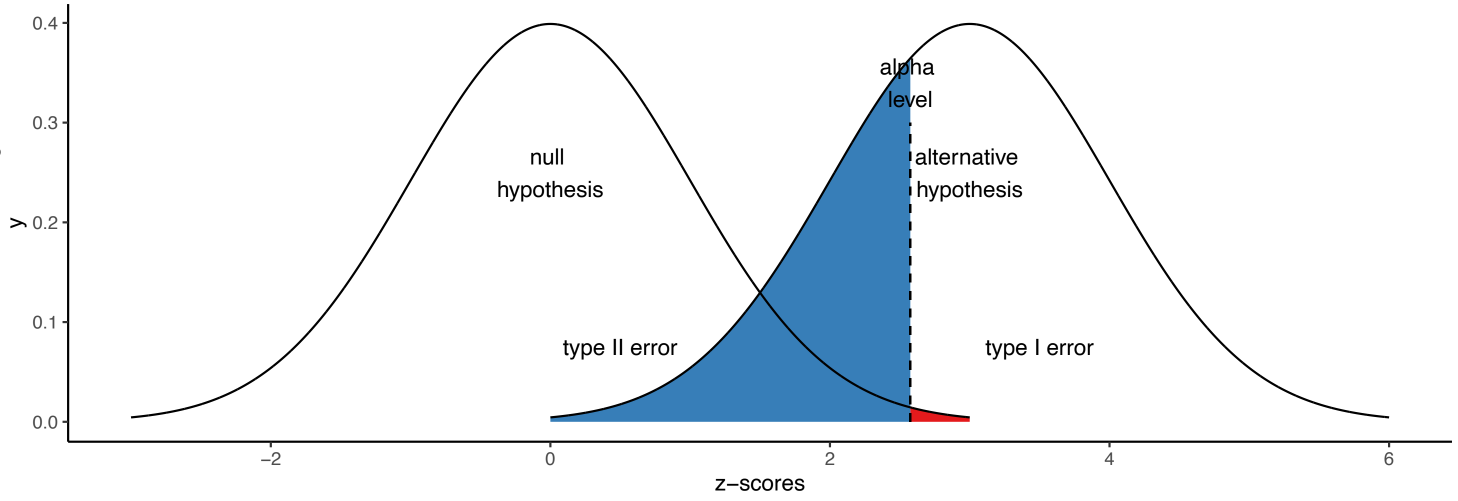
The disadvantage of Dunn's correction is a severe loss of statistical power

The Dunn correction achieves control over α_{EW} by shifting the alpha criterion. We already know that shifting the alpha criterion always increases type II errors. In other words, we lose statistical power. You can see that increasing the comparisons will increase type II errors (reducing power):

1 comparison, *p*=.05



10 comparisons, *p*=.005*.* (There are 10 comparisons when you have 5 conditions.)



Planned vs Post-hoc comparisons and the Dunn correction

Maybe we just look at a subset of comparisons?

Given that increasing the number of comparisons decreases power, it is tempting to only look at a subset of comparisons that are in an experiment, rather than all of them. There are two ways to do this, and the way you choose impacts how you have to correct your data!

Planned Comparison:	This is a comparison that you specify before running your experiment (and crucially before looking at any data). Basically, you have a specific hypothesis, and decide that the best way to test it is to compare certain levels to each other.
Post-hoc Comparison:	This is a comparison that you decide to run after looking at your data. Basically, you see a difference in your data, and are curious to know if it is significant.

For planned comparisons, you set C to the number of comparisons that you plan to make. It works the way you expect it to!

For post-hoc comparisons, you have to set C to the total number of comparisons. This is because post-hoc comparisons will include all of the errors. I will show you in the next slides.

Planned comparisons

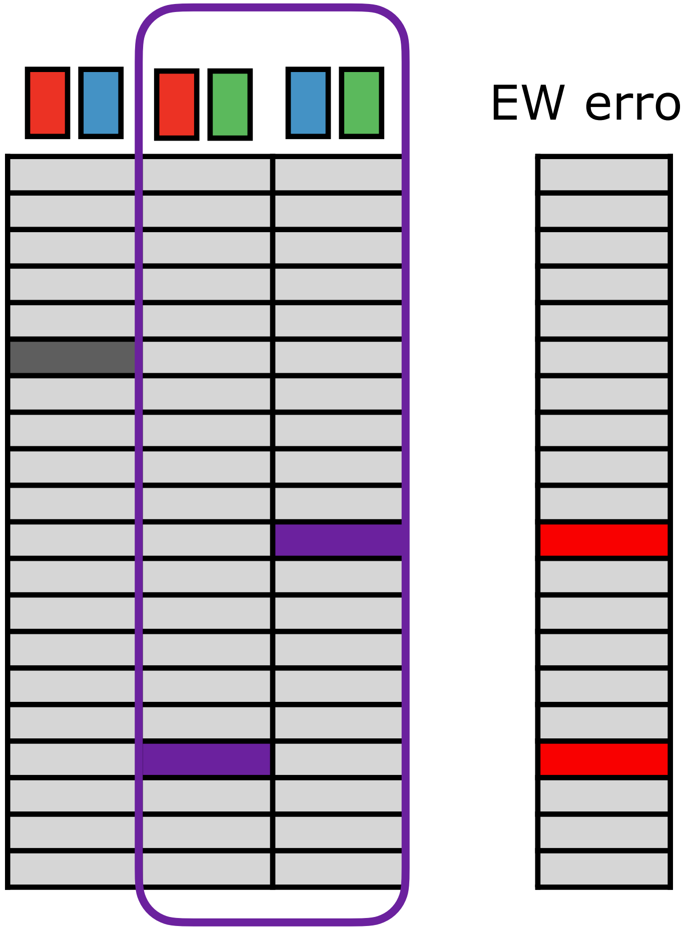
Let's say you run a 3 condition experiment, and decide that there are only two comparisons that you actually care about:

If we only look at two comparisons, and don't look at all at the first comparison, we won't see its errors. I have grayed out its error to illustrate this.

Therefore, when we calculate the experimentwise error, the only errors that will influence the count are the ones from the two comparisons that we look at. So our α_{EW} will be $1 - (1 - \alpha_{PC})^2$ because we have 2 comparisons.

So we can use Dunn's correction with $C=2$:

for an α_{EW} of .05: $\frac{.05}{2} = .025$



$\alpha_{EW}=.0975$

Demonstrating Dunn and Planned Comparisons

We can simulate this like before. We will run 3 comparisons, but only look at the *t*-values for 2 of them!

Step 1: Simulate 10,000 experiments with 3 comparisons.

```
boot.replicates=replicate(10000, expr=rnorm(150, mean=0, sd=1))
```

Step 2: Write a function to calculate 3 t-tests, but only report 2 of them.

```
t.function <- function(dataset){
  t1=t.test(dataset[1:50], dataset[51:100], var.equal=T)
  t2=t.test(dataset[1:50], dataset[101:150], var.equal=T)
  t3=t.test(dataset[51:100], dataset[101:150], var.equal=T)
  return(c(t2$p.value,t3$p.value))
}
```

Step 3: Apply the function to the simulated experiments.

```
experiments = apply(X=boot.replicates, MARGIN=2, FUN=t.function)
```

Demonstrating Dunn and Planned Comparisons

Step 4: Write a function that looks for errors at *p*<.025 (the Dunn corrected alpha level).

```
error.corrected <- function(dataset){
    errors=sum(dataset<.025)
    return(errors>0)
}
```

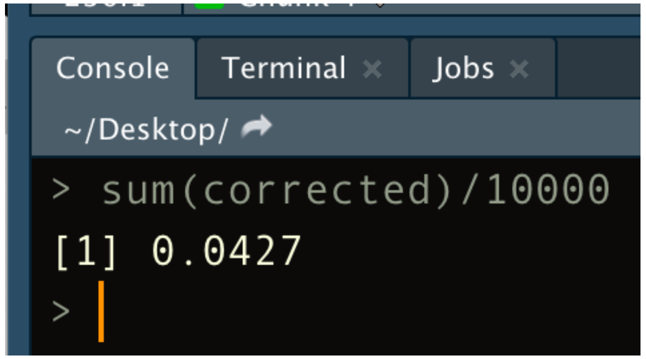
Step 5: Apply the function to simulated *t*-test results.

```
corrected=apply(X=experiments, MARGIN=2, FUN=error.corrected)
```

Step 6: Count the errors for each method.

```
sum(corrected)/10000
```

And we see that the corrected error rate is approximately .05, just like we want!



```
> sum(corrected)/10000
[1] 0.0427
>
```

But notice that we did this by looking at only two planned comparisons and a C of 2. This shows that the Dunn correction works as expected with planned comparisons.

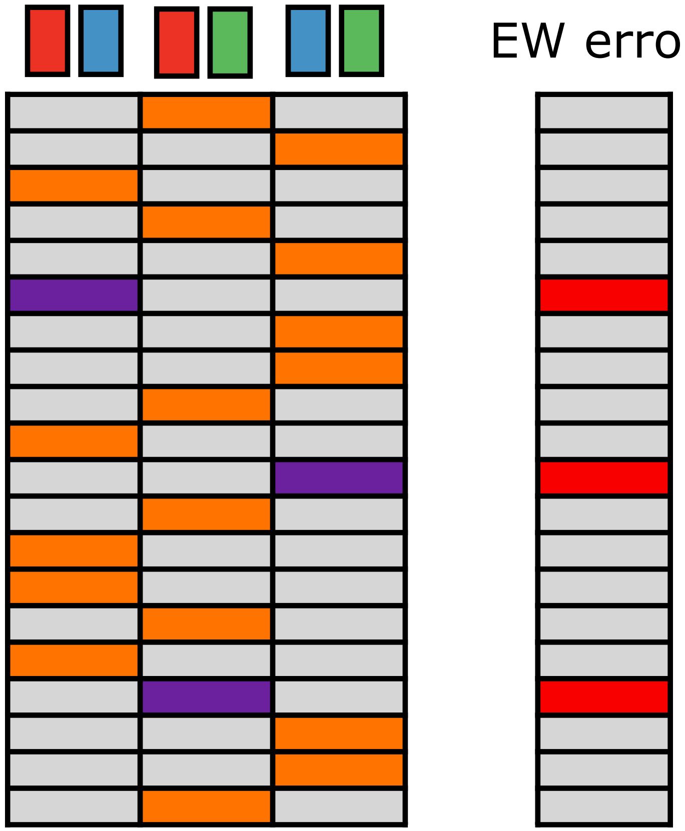
Post-hoc comparisons

Let's say you run a 3 condition experiment, and decide to look at the one comparison that "looks significant". What this means in practice is that you will choose the largest effect in the three comparisons:

I will signify the ones that we choose with orange shading. These are the largest effects out of the three comparisons.

But what about the experiments where there is an error? Which comparison is the largest? The error! By definition, the error is the largest effect.

So, if we choose based on the largest effect, we are guaranteed to select all of the errors.



α_{EW}=.1426

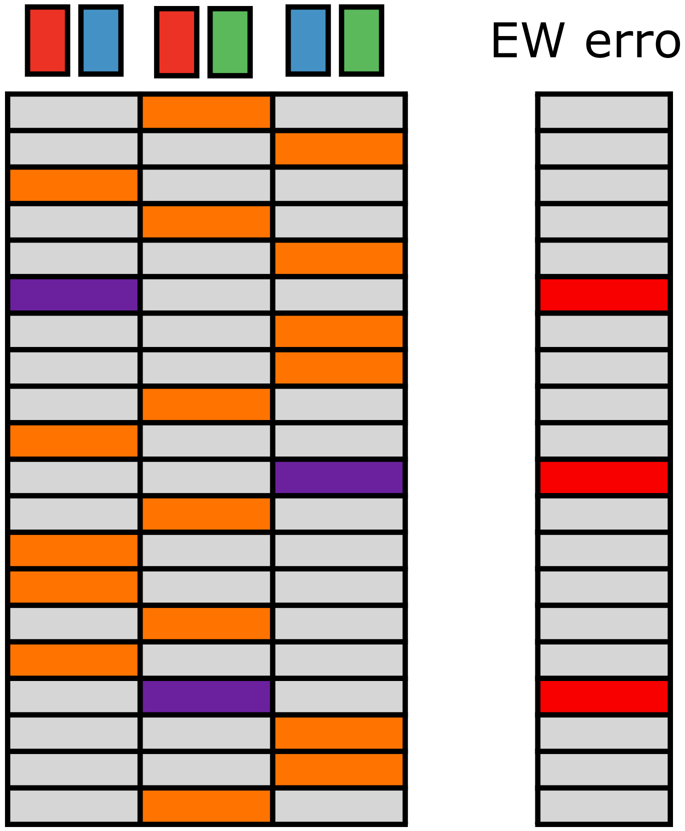
Post-hoc comparisons and Dunn's correction

Now, let's apply Dunn's correction in this scenario. Our first option is to use C=1. We are only testing 1 comparison per set of 3, so this might seem to be reasonable.

for an α_{EW} of .05: $\frac{.05}{1} = .05$

But when we do that, we see that we use *p*=.05 for our alpha criterion. That is no correction at all. So, based on the logic we just saw, we will find all of the errors. Our α_{EW} will be .1426, not .05.

This also shows us what we have to do. We have to use C=3. If we do that, we will get the correction that we need to bring α_{EW} down to the level we want to be at.



α_{EW}=.1426

For post-hoc comparisons, you have to set C to the maximum number of comparisons. This is because choosing comparisons post-hoc will always select all of the errors across all of the comparisons, regardless of the number of comparisons you choose.

Demonstrating Dunn and post-hoc comparisons

We can simulate this like before. We will run 3 comparisons, but only look at the *t*-values for 2 of them!

Step 1: Simulate 10,000 experiments with 3 comparisons.

```
boot.replicates=replicate(10000, expr=rnorm(150, mean=0, sd=1))
```

Step 2: Write a function to calculate 3 t-tests, but only reports the smallest *p*-value of the three (therefore the largest effect).

```
t.function <- function(dataset){
  t1=t.test(dataset[1:50], dataset[51:100], var.equal=T)
  t2=t.test(dataset[1:50], dataset[101:150], var.equal=T)
  t3=t.test(dataset[51:100], dataset[101:150], var.equal=T)
  return(min(c(t1$p.value,t2$p.value,t3$p.value)))
}
```

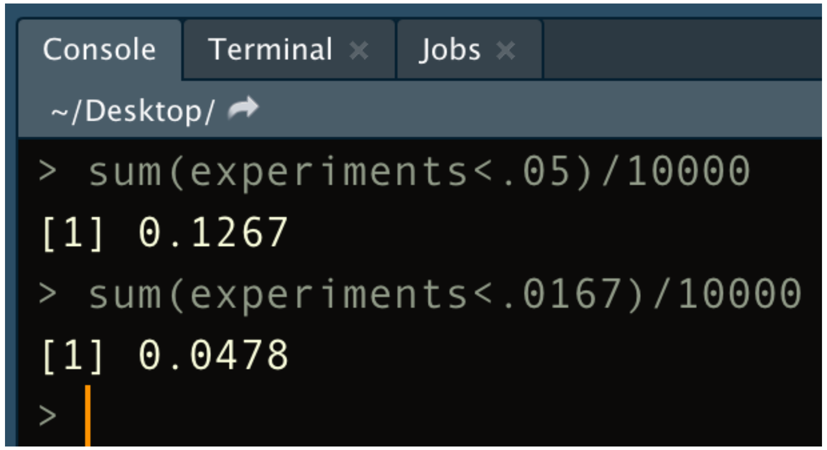
Step 3: Apply the function to the simulated experiments.

```
experiments = apply(X=boot.replicates, MARGIN=2, FUN=t.function)
```

Demonstrating Dunn and post-hoc comparisons

Step 4: We don’t need an error function, because we only have 1 *p*-value per experiment. We can just sum up the number of *p*-values at the level for C=1 (.05) and the level for C=3 (.0167)

```
sum(experiments<.05)/10000
sum(experiments<.0167)/10000
```



```
> sum(experiments<.05)/10000
[1] 0.1267
> sum(experiments<.0167)/10000
[1] 0.0478
>
```

And what we see is that the C=1 level yields an α_{EW} that is approximately .1425, whereas the the C=3 level yields an α_{EW} that is approximately .05.

This shows that, with post-hoc comparisons, even if you only do one comparison out of the set of possible comparisons, you still end up with the maximal error rate. So you need to correct using the maximal number of comparisons (in this example, 3). Then you will get the α_{EW} that you want.

Tukey’s Honestly Significant Difference

Tukey's HSD

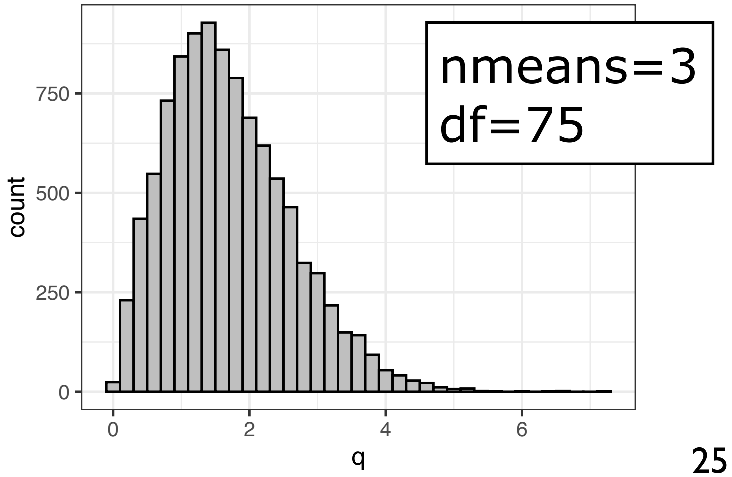
John Tukey (1915-2000) is probably the person who did the most to draw attention to the multiple comparison problem (in addition to lots of other cool stuff in his life, including coming up with the word "bit"). He said it was "dishonest" to not correct for multiple comparisons, so he called his method the "honestly significant difference". Hence, Tukey's HSD.

Tukey's approach is to create a new test statistic that takes α_{EW} into consideration as it is generated. This means that the *p*-values that are derived from the distribution of the test statistic will control for α_{EW} inherently. **In other words, the *p*-value is α_{EW}.**

studentized range statistic

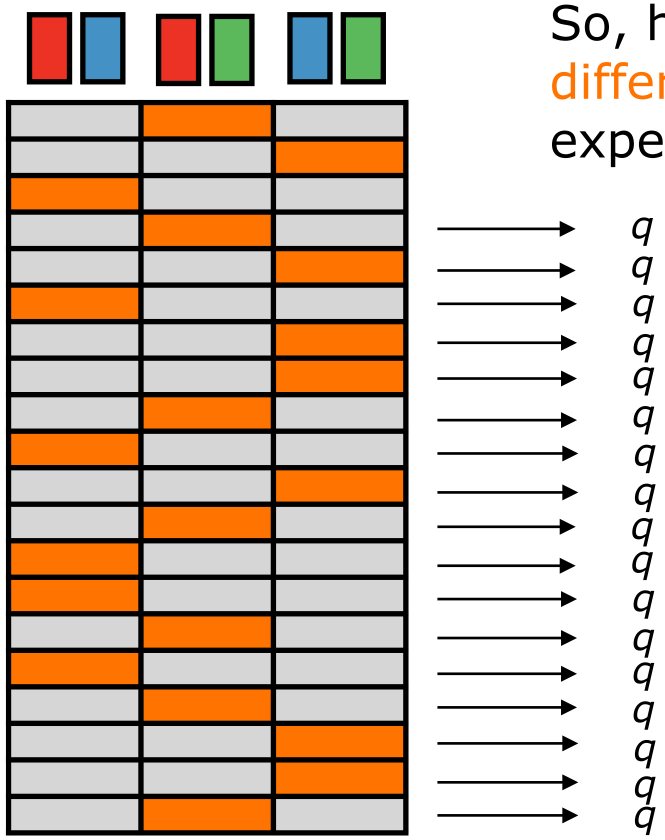
$$q = \frac{|\bar{x}_i - \bar{x}_j|}{\sqrt{\frac{MS_W}{n}}}$$

The statistic he created is related to *t*, as you can see from the equation. It is a family of distributions that take two parameters: the number of conditions in the ANOVA, and the df_W from the ANOVA. There is no built-in function in R to generate the distribution, but I created an empirical one here for you.



Tukey's HSD and α_{EW}

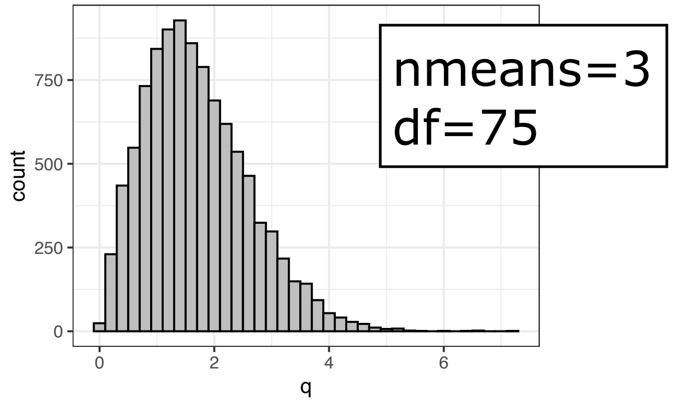
Tukey's insight was that, if there is going to be one or more errors in an experiment, the largest difference in the pairwise comparisons will be an error.



So, he created a null distribution for the largest differences. This is the critical comparison in each experiment - it is the one that is most likely an error.

To generate this null distribution for the largest differences under H_0, we calculate a statistic called *q* for the largest of the pairwise comparisons over and over:

$$q = \frac{|\bar{x}_i - \bar{x}_j|}{\sqrt{\frac{MS_W}{n}}}$$



Then we use this distribution to find a *p*-value. We only reject H_0 when we pass .05. We will make an error 5% of the time. And this error will be an experimentwise error because it contains **every experiment that has an error!**

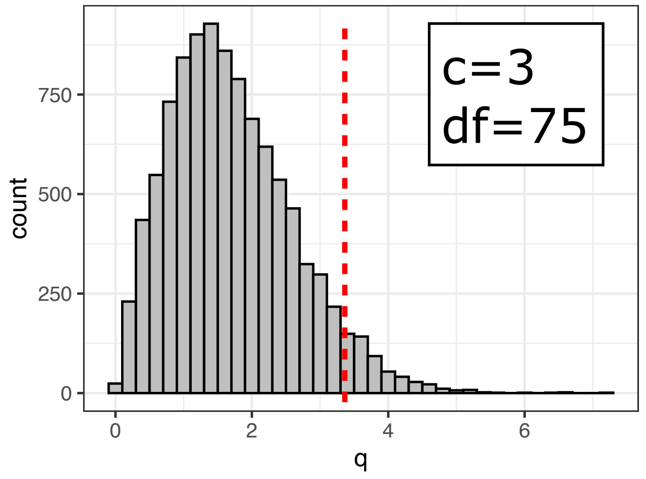
Using Tukey’s HSD

Tukey’s HSD looks just like a *t*-test, and that is not an accident. You can use it just like a *t*-test. Plug in the two means from your comparison, and derive a *p*-value from the resulting test statistic (*q*).

The only thing to remember is that you need to calculate an ANOVA first. This is because Tukey’s HSD uses MS_W in its estimate of the standard error.

studentized range statistic

$$q = \frac{|\bar{x}_i - \bar{x}_j|}{\sqrt{\frac{MS_W}{n}}}$$



So, we can approach this just like we do other tests. We can either calculate it by hand, or we can use R.

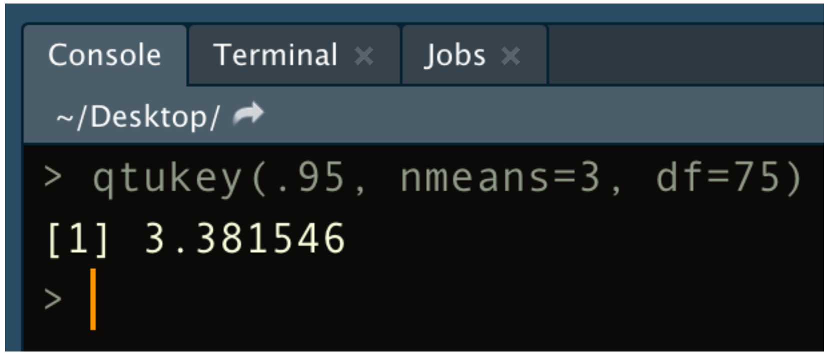
Calculating Tukey's HSD by hand

Step 1: Calculate the ANOVA. You will need MS_W and df_W from this.

Step 2: Calculate the *q* statistic for each comparison you want to make.

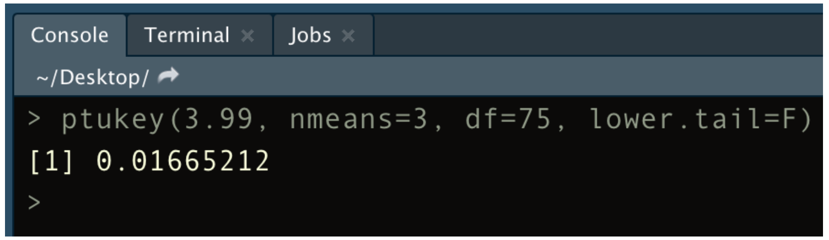
$$q = \frac{|\bar{x}_i - \bar{x}_j|}{\sqrt{\frac{MS_W}{n}}}$$

Step 3: Look up the critical *q* statistic for your chosen α_{EW}. To do this, you can either use the table in the back of the book, or use the qtukey() function in R. You can use this to make a Neyman-Pearson-style decision based on your observed *q* from step 2.



```
> qtukey(.95, nmeans=3, df=75)
[1] 3.381546
>
```

Step 4: Look up the precise *p*-value of your observed *q*. You can use the ptukey() function in R.



```
> ptukey(3.99, nmeans=3, df=75, lower.tail=F)
[1] 0.01665212
>
```

Step 5: Make a statistical decision just like you normally do with an alpha criterion of .05, knowing that you are controlling α_{EW} at this level.

Calculating Tukey's HSD using R

I will create a quick 3-condition data set for us to use:

```
red=round(rnorm(10, mean=3, sd=.75), 1)
blue=round(rnorm(10, mean=5, sd=.75), 1)
green=round(rnorm(10, mean=7, sd=.75), 1)

data = tibble(group = rep(c("red", "blue", "green"), each=10),
wellbeing = c(red, blue, green))
```

Step 1: Create the ANOVA using aov() and save it to a variable name.

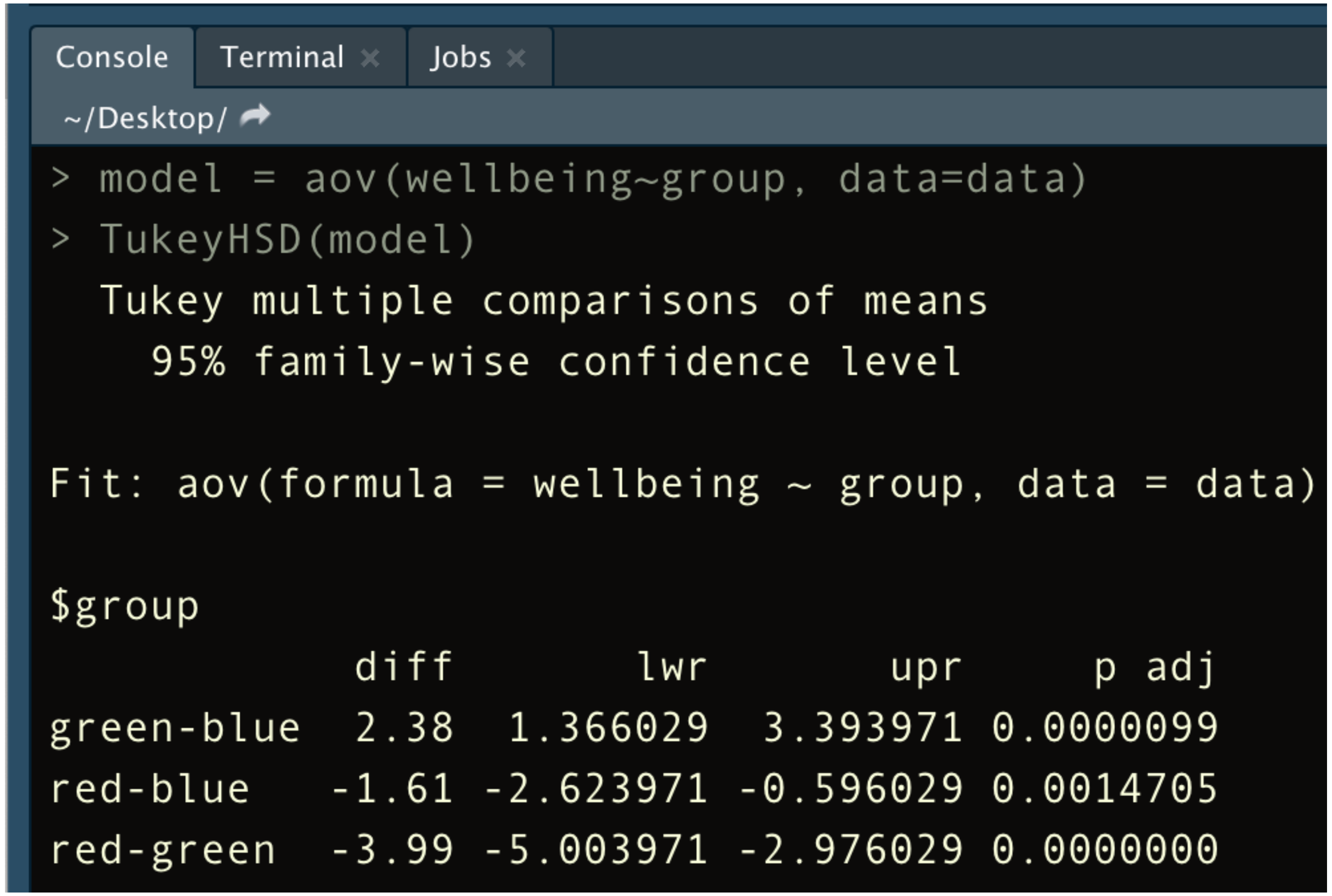
```
model = aov(wellbeing~group, data=data)
```

Step 2: Use the function TukeyHSD() to calculate the *p*-values.

```
TukeyHSD(model)
```

Calculating Tukey’s HSD using R

The output of TukeyHSD():



```
> model = aov(wellbeing~group, data=data)
> TukeyHSD(model)
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = wellbeing ~ group, data = data)

$group
             diff       lwr       upr     p adj
green-blue   2.38  1.366029  3.393971 0.0000099
red-blue    -1.61 -2.623971 -0.596029 0.0014705
red-green   -3.99 -5.003971 -2.976029 0.0000000
```

It gives you the difference between means, the bounds of the confidence intervals, and the **adjusted *p*-values** for each of the pairwise comparisons.

A quick note on corrected *p*-values

Methods like Tukey's HSD that create a new test statistic yield *p*-values. These *p*-values are called **corrected *p*-values**. This is because the *p*-value itself is already corrected for multiple comparisons. So you the alpha-criterion that you use will control the experimentwise error.

The *p*-values that our standard test statistics yield (z, *t*, *F*) are called **uncorrected *p*-values**. This is because the alpha-criterion that you use will not control the experimentwise error.

When you report *p*-values for an experimental design with more than one pairwise comparison (so, more than two conditions), you need to say whether the *p*-values are corrected or uncorrected. **This matters**. It matters both so that the reader knows if you corrected for multiple comparisons, and because corrected *p*-values do not tell us the probability under the null hypothesis. They tell us the probability after correction.

Also note that the Dunn correction does not change the *p*-values. It changes the alpha-criterion. So it does not yield corrected *p*-values. The *p*-values are still uncorrected. Sometimes people will multiply the uncorrected *p*-values by C to yield something like a corrected *p*-value because multiplying the *p*-value by C is the same as dividing the alpha-criterion by C for all decisions. But the resulting number is not really a *p*-value any longer because it can go above 1! (e.g., an uncorrected p=.75 with 3 comparisons would yield p=2.25!)