Everybody welcome. Today I'm going to go through

the multivariate confidence regions demonstration.

As always, we have our friend library here.

We'll also get another library called car.

Car stands for companion for applied regression.

You're completing the companion to applied regression.

It has a function useful for drawing ellipsoids.

We will also use our old friend,

GGally; readr and dplyr.

Lastly, we'll load the package expm for this little operation,

which is taking matrix to some power.

We discussed this in the lectures on matrices.

Again, we will use the microwave ovens data.

We will use it to illustrate the Hotelling's T squared

confidence regions and as well as to develop some intuition for their construction.

We're going to load the dataset and we're going to transform it as we did last time.

Again, here's what they look like.

For convenience, we're going to extract the sample size,

the number of parameters,

or number of variables for the sample mean using the colMeans function,

which we talked about earlier.

The sample covariance,

or sample variance covariance matrix and also for convenience,

its inverse, which we obtain by sole function.

Now let's think about how to draw this ellipsoid.

One thing we can do is something crude.

We can just say,

we know from lecture that this ellipsoid is defined by this expression that n

times the mu minus

the value of the coordinate times

the sample variance inverse times m minus x.

M here is the sample mean and x is the coordinate,

if we want to know whether it's in the ellipsoid or not is

less than this critical value which,

again from the lecture.

Then well let's try a whole bunch of points

and test them for being incited, this ellipsoid.

There's also a more efficient way to actually take eventually the Eigen decomposition.

Let's do that. Brute force,

we'll use a seq function.

This is basically saying take 200 numbers starting from

0.5 and ending at 0.65 and similarly for opens.

Then expand grid, there's slightly obscure function,

but basically what it does is,

it takes every combination of the closeds vector and the opens vector.

Lastly, we'll use the apply function.

For every element in this grid,

and again, I would encourage you to try realign yourself and see what happens.

For every rule in the squid,

we check whether the point is in the ellipsoid.

Then we provide additional arguments,

namely the sample size, the mean,

and the standard deviation squared.

The way you're going to use Chi,

we're going to be using confidence level of 0.95.

Then we're going to plot these points here,

and we're going to colour code them according to z.

Makes sense? Z here is whether a zero or one depending on whether it's an interval.

Then we're also just for convenience,

we're going to plot a point at the mean just to see where that is.

Then we'll begin.

Then we'll run a few more things.

Namely, we'll take the Eigen decomposition of S and here's what it looks like.

Here the eigenvalues, here are the eigenvectors,

and the columns are the eigenvectors.

Then the scaling factor, this,

you might recognise it from here except that n has been moved from,

the divided both sides by n.

Then we draw lines,

and basically we draw lines from m minus this eigenvector times

the square root of the eigenvalue times

the scaling factor then plus that and the same thing for minor axis,

which, so here, the major axis we get by the first bigger eigenvalue.

This one. The minor axis we get from the smaller eigenvalue,

here. Here's what we see.

Remember, when our plots,

col equals 0 or false, it just skips.

When col equals 1, it plots a black points.

These points represent those points on that grid we constructed over here,

that are actually inside the confidence ellipsoid.

Then obviously the mean constructs this point here.

Last but not least,

the axes are drawn using these two lines.

It turns out that, of course,

R has this capability,

it could do it for you.

We still will start by plotting open against closed just to set up the plot.

Then we're going to use this LMC bind,

closed, open, one data equals ovens 4.

We'll talk more about what that does later in the course,

but for now we're taking the close and open and variables,

we are putting them together into one matrix.

Then we're just fitting a linear model,

just to intercept to it.

Then we feed the resulting function called confidenceEllipse.

Here is where it draws to the same ellipsoid.

You can check whether these coordinates match.

As with the other confidence intervals,

we can be 90 percent confident that

this ellipsoid contains a true coordinate pair

that represents the population mean for the two variables.

We've talked about two ways of breaking

down individual confidence intervals and for each variable.

One way to do that was the method of projecting the ellipsoid.

This method, actually a function called a contrastCI here,

which basically it implements the formula in the lecture.

We could evaluate these to get contrast CI's for unit vector.

This actually works for any linear combination L. But for now,

if we just want to pick the first variable,

we would just have the first element to one and the second to

zero and plug-in and similarly for the other.

If we wanted to,

we could actually plot them at the same time as ellipsoid.

In fact, what we would see is that these lines,

these intervals represent the shadows of the ellipsoid.

That is the projection of the ellipsoid onto the horizontal and the vertical axis.

It's not an exact fit because of the way our plots,

this ellipsoid, notice that it's actually a whole bunch of really short segments.

It's an approximation. Bonferroni,

all we do there is we just have

a one sample T-interval with higher individual confidence level.

Instead of 0.95 here we use point 0.975 and that's that.

Interestingly enough, these are floating narrower.

If you're familiar with Bonferroni confidence intervals and Bonferroni adjustment,

you'll know that it's a bit conservative.

That is, it considers the worst-case scenario for correlation between

the two variables and it's

guaranteed not to fall below

the target confidence level or exceed the target miss probability,

in this case 0.05.

However, in this case,

they're actually a bit narrower.

This is because here,

these confidence intervals cover all possible linear combinations.

It could be 0,1;

it could be 1,0, it could be 1,1,

it could be 1,2,

it could be minus 1,1 for their difference, that will be useful.

It covers all of them at the same time,

whereas Bonferroni only covers these two.

That's the price you have to pay. This concludes this demonstration.

Good luck with your challenge.