Advanced Quantitative Tools: Using R

In this module, we are going to talk about some advanced quantitative tools; in particular, we're going to be using R for these tools.

Slide 2: So how do you get started using R?

So the first thing is: How do you get started using R?

Slide 3: www.r-project.org

The easiest is way, of course, is you go to r-project.org, and you can download the version for the particular system that you're interested in using. And R, as we said in the previous module, is a statistical package built upon S and S-plus. And the great thing we said was, it's what is used by statisticians. So it has many of the tools that we want.

Slide 4: Press CRAN for mirror sites

So from the cran site, we can go to one of the mirror sites. For example, this site is the Academic Computer Club in Umeå or from the Swedish University Network. And we can download a copy of R for the machine that we want.

Slide 5: R Distributions

So we can choose to download the binary image for Linux, macOS, Windows, etc.

Slide 6: Linux Distributions

We can even get the source code. So here we see how we can get a Debian, Red Hat, SuSE, Ubuntu, etc. We notice is that the SuSE releases very very old, and the reason for that is that since SuSE 11.3, R is actually included in OpenSuSE release, so there isn't any need to specially downloaded it from one of the sites - as it's already there. Just say that you want to use that package.

Slide 7: R Manuals

Now there is extensive documentation available for R, including an introduction to R, how to import and export data to and from R, how to install and maintain R, and of course, how to write extensions. The R language is defined. And there is a set of internals for the internal data structures - in the case you would like to add new information and processing functions to R. And there's even an index. And you can go to any one of these yourselves and get the appropriate document that you're looking for. The key feature is that it is open-source, so the result is that you can easily be extended to support the things that you find missing.

Slide 8: R Packages

Now, as we mentioned one of the important reasons for using R is because that's what statisticians use, in particular lots of people have created their own packages, and you can go to this website to get the list of all the available packages, either by date or by name, etc. And therefore, hopefully, you'll be able to find a tool that suits the purposes that you need. Otherwise, yes, you can write your own.

Slide 9: Obtain an R Package

So to obtain an R package, in this case, we're going to look at the package gtools. And we can say, all right. What is it? Well, this is a set of tools to assist in R programming and what we can see here is the version of it, what it depends on so, it needs an R version later than 2.10, when it was published, who are the authors, who the maintainer is, what sort of license it requires. And we can now get the manual. We can get the source package, or we can just load the binary packages, and off we go. It also has a reversed dependencies list, so we can see not only what it depends on [but] we can see what depends upon it.

Slide 10: Install an R Package (linux)

Now once we've located that package, we don't need to uncompress it; in the case of Linux, we simply type capital "R command (CMD) install" and give the name of the gzipped package and violà - it's going to install it for us.

Slide 11: Importing Data into R

Now to import the data into R, what we do is we basically say: Well, a common format of is comma-separated variables sorry comma-separated values .csv and there's an operator called read.csv were we specify the file and we say whether or not there's a header for each of the columns in the table. If there is, we set header to TRUE, and we assign the result to the table that's going to store the data that we want. Now once we have it, we can manipulate it using all the various R tools. In addition, we can say, "Well, actually I've already got it in the spreadsheet so if I have it in an Excel spreadsheet I can still load the package gdata and now you can call read.xls and even say which particular sheet of the spreadsheet I want to read and now will get loaded into a data frame which now I can reference by rows and columns."

Slide 12: Example using a *csv* File

So continuing with an example using a CSV file, (as we said a comma-separated values file) in this case we have a little function who's going to read a series of values in from different files, each of which has a header, the separator, in this case, was actually a comma, we put the data into these data frames. And now we can manage to, of course, do all of these things within our function. So you can do that again and again, which makes it very convenient.

Slide 13: Importing Any File

But it's actually possible to support any type of file so that we can use the scan function - so, in this case, we say that we want to put into this data frame the result of scanning, rebuild the string, which in this case included a partial string "std.decay.time", the separators for spaces and the values were numeric, and violà we will proceed to load that into this dataframe. And as I said, we can create a function call for this - so that we can use it again and again, if that's the kind of data that we're going to be processing.

Slide 14: Exporting Any File

Not only can we import files, but we can also output files; it has a function "cat," so we can simply write to a text file - just as we can update the data to the screen. There's also a dput file so you can put it into a file directly, and then we can get the data that we just output using dput back in by using the dget function.

Slide 15: Experiment 2: DNS lookup

So, if we think about our second experiment that we described in an earlier module, in this case, we're looking at the time to do a domain name system lookup. We captured that traffic, and we used the tool Wireshark, and we set the filter to be UDP port equals 53, which is the part number for the DNS traffic. And then, we exported the file in the so-called PDML format, and we produced the file in this case "dns-capture-20100915a.pdml" and using emacs, we removed all the lines except those containing the dns.time field, and now I'll put it into a text file. And so here we can see that we're going to simply read this table in from our text file, it has no header, since we have in this dataframe we can simply say "summary" of the dataframe and violà, we see the minimum value, the maximum value, the first quartile, third quartile, the median, and the mean values. And now, of course, we can run the function foo, which we showed in the earlier module on that dataframe for a particular length, and we can compute all of the other sorts of statistics we would like to be able to compute. Thus making it possible to run one run after another after another run - to and look at the data for different days, different times of the day, etc. and get our statistical values from that.

Slide 16: DNS lookup time graphs

So if we want to plot those values, in this case, we see on the left-hand side a histogram of the data that we read, and we put a Y label on it the "DNS query time in seconds", and we said we want to split it into 40 different bins, it automatically scaled this for us - violà [CLICK] We have our histogram. We can also compute a box plot. Again we can specify the Y label, plots our data here, and we see all of these are outliers, down here we can see our first and third quartiles. And we can see that the box is very squished - because most of our data, of course, is way down all of this bin here and only a little bit here, which we can see here in the box plot.

Slide 17: More graphs: change scale

So, we can, of course, say, "Well, let's zoom in on this"- So, for instance, in this case, when we plot our histogram, we decided where we want to make our breaks, we specified the list breaks. You can now plot our data, and we can leave off the first part of the data and so we can see the remainder of this data on a rescaled version of the plot- And we can change, of course, which character we use for our plotting, and we can change the color of the markers, etc. etc.

Slide 18: DNS response CDF

Now, we might also want to compute the cumulative distribution function. So now, we can call the empirical cumulative distribution function on our data. You can specify our Z label, our Y label, which symbol we want to use, and off we go. We also specified the size of the sample. And we called grid to put in these background grid lines. So now using this plot it is very easy to see that, in fact, 80% of the time our delays were extremely small but then there was a little plateau here and then it climbs a bit more, and then we have another plateau, and then it slowly climbs up here until we're now up around 95% or so at 0.2 seconds or 200 milliseconds. So that means this is around 90% of the time we're below a tenth of a second and about 95% below two-tenths of a second. But the great thing we saw is that most of the queries were answered very fast - but we also have these ones out here -that were taking quite a long time to be found. And, of course, if you're interested in DNS, you can think about: "How can we perform this better?". So we can introduce DNS proxies, we can look at our caching times, etc. But as we saw, it was very useful to be able to make the plot show the data that we want to show.

Slide 19: Advanced plot Formatting

So we will talk a little bit about advanced plot formatting.

Slide 20: Plot Formatting

So in this particular case, we see another function that's been defined as called cup.measures. It's going to read in some data from a CSV file; it's going to store the data from that first data file, take a portion of it, and put it into this data frame. Here is the information about the things that we would like to do. And now here's our plot command. So we plot the numbers 1 to 14; we use the data from this, which was 14 of our values. We say here's the X label we want to have, here's the Y label we want to have; we can space the Y labels out, we can set our limits, we can put a title on the whole table, we can put a subtitle on it, etc. And the big advantage here, and I should note this plus means it's a continuation line, is that we can produce a beautiful looking plot with just what we want.

Slide 21: Add Labels to the Points

So if you want to add labels to the point, we can load the library called "plotrix". We can now get the labels now out of the data frame that we computed, put them into another data

frame called "plotlabels". Now we can say, "Okay! this thigmophobe.labels is the first fourteen from that, use the plot levels, use the color darkblue, and font number 2." So the labels are going to appear in darkblue, it turns out that the label font is a bold font.

Slide 22: Add Another Plot to This One

We can add another plot to this. So we can overlay multiple plots; in this case, we're going to add another plot, and we're going to put them in darkgreen.

Slide 23: Do Some Statistics and Add to Plot

And we can add some more statistics to it. So, we can have a mean, twice the standard deviation - so add lines to indicate that. We put those into the plot. We add our labels to it.

Slide 24: Finish Plot

And now we finish up the plot; we create a box around it, we apply the axes that we want, etc.

Slide 25: Example Finished Plot [Goldvasser 2012]

And now the advantage is this beautiful looking plot, which was used in a paper by Dov Goldvasser et al. in 2012. The labels up here, our main label here out Y label for the mean plus two standard deviations, are lying down here for the mean minus two standard deviations. The subtitle we wanted in, in this case, it was fourteen different scans that were done, numbered 1 to 15, here was the diameter, in this case, that was a hip prosthesis that was being measured; specifically, the cup of the hip prosthesis. And these are the individual labels on it corresponding to different observers and different trials. And we can see that the data that falls well within plus or minus two standard deviations. And we can see some trends for particular users. And if one looks at the table for what are the colors of what are the particular studies - you can get lots and lots of data out of this, So we can see a rather complex plot can be generated rather simply using the advanced plotting functions of R.

Slide 26: Figure Legends

Now, if you want to add figure legends, so that we don't have to say somewhere else, "okay, which one of these is which figure", we can place them in different positions. We can use different symbols. We can put them outside of the graph.

Slide 27: Example Plots [Goldvasser 2012]

Now, we plot all of that data and now what we can say is, "yes, different symbols for used it is the same data [as] before" and now it's very easy to say, "here was scan 1, trial 1, scan 1, trial 2" and we can look at how do those points trend over the set, in this case, it was repeated studies of the same physical phantom, a king of hardware, and here we can see a big

difference between the expected difference and the measurements that were made. And so here's our plus and minus a tenth of a millimeter. And we see that basically nearly all of those points lie within a tenth of a millimeter of the expected value.

Slide 28: Remarks

Well, one thing so we can notice in the previous set of slides: Is that the example functions that we made, we're just using R functions. But it's also the case that we can add to them. But if we are going to write an R function. What is the easy way of writing the function? Well, the easiest method I've found is to simply go inside your emacs, create a shell window and run R. And enter the commands that you want. Once you found the sequence of things that you want to do, take those and put them into a named function. And now you can apply them again and again. And if you want, you can parametrize it, etc. But it makes it very, very easy to quickly find the set of things you want and to build a functional behavior that you like.

Slide 29: Error bars

Now, we will shift gears a little bit to one of my perennial favorites for a plot, and that is error bars.

Slide 30: Why show error bars?

And you might ask yourself, "Why do you bother showing error bars". And the reason is you want to convey to our readers the expected range of the values that we expect. So between the whiskers on the error bars and these whiskers show our confidence interval (and that might be set to 90% or 95% or 99% percent) - that corresponds to in the 90% case, that 10% of the values could be expected outside of that range. And we would say "it still basically fits our model, it's not surprising, we expect some values outside that range", but as we increase our confidence interval, then a smaller and smaller probability should be for "true values" so correctly measured values and that are values that we believe should be outside that range. So if we have a very high confidence interval, then there should be a very small number of points outside that range.

Slide 31: Error Bars in R

Now, in R itself, it's very easy to plot error bars using the package gplots. If we go look at the reference manual for it, it gives instructions for a function called plotCI, not surprisingly plot a confidence interval. And there's a nice example of the kinds of plots that you can make and the code to draw them at this URL, and I encourage you to take a look at them.

Slide 32: Example of Error Bars in R – read in data and format for finding CI

So what can we do? Okay. So, in this case, we're going to create a function. We load our gplots library; we read the data, then we say here the expected values for our different sets.

We do the computations. We now say, "aha!" Now that we have the data in the vector, we can compute these particular portions of it, and we can now print the values that we just put into this data frame. And what do we see?

Slide 33: Error Bars in R – find 99% CI

Well, we can now calculate the mean of them, and you can compute the square root of the variance of them and plot the standard deviation from the expected value.

Slide 34: Error Bars in R – Plot Graph with error bars

And now we can find our error bars and so 99% that corresponds to 2.58 times the standard deviation divided by the square root of the length, and now we've got our upper and lower confidence intervals. And we ca output those values and, of course, we can plot them, if we build up a plot. We can do our calculations. We compute the confidence intervals. We frame it up; in this case, we're going to make a dataframe that's going to show us one frame. We put a box [and] we place our axes.

Slide 35: Error Bars in R – Resulting Plot [Goldvasser 2012]

And now, what do we end up with? This lovely plot here. We now see the expected difference from our experimental data, and we see here the error bars in the expected difference for each of those different measurements that were made. And we now see the error bars for each of those measurements at the 99% confidence interval value.

Slide 36: Selected Topics in Quantitative analysis

So we've talked a bit about plotting now we're going to shift gears once again. We're going to talk about some other topics in quantitative analysis. And one of the first of these is power. Statisticians talk about power in terms of statistical power.

Slide 37: Power

Slide 38: Parametric and non-Parametric Data

And so if you have data that are normally distributed (so it follows a Gaussian distribution), then we can say that we can describe it using parametric statistics. Those statistics we've talked about earlier: mean, variance, and standard deviation. And we can calculate those in R by simply saying, "mean of our data frame", "variance of the data frame", and "the standard deviation of something" the square root of the variance or we can say directly calling the function speed on our data frame, and while we have our standard deviation - so it's easy to calculate the parametric statistics.

Slide 39: Consider the case of measurement of the diameter of a femoral component

So let us take a look at those measurements again. Now, not of the cup but of the femoral head - it's the little ball that goes into the cop in a prosthetic hip.

Slide 40: Normal or Parametric Data

And, in this case, we read our data in, and now we're going to compute a one sample t-test. So to do this, we got our data, and we get our values. And what does this tell us? The result of the t-test using this data tells us that our p-value is 2.2 times 10^(-16). And the alternative hypothesis is that the true mean is not equal to zero. The 95% confidence interval is 11.05714 to 11.07268. And the mean is 11.06491. Well, what does that mean? Well, that means the data here says that the true meaning is not equal to zero. That is our hypothesis is that it's highly significantly different from zero - so the data isn't zero. That's not surprising because the mean of the data is up around 11 millimeters.

Slide 41: Normal or Parametric Data - 2

Now, we could think of doing another computation, so, in this case, we're going to say instead of t-tests that we just did, we're going to now do another t-test, but we're going to say that the mu (the mean) is different from zero. And we're going to say that it's 11.066, which we saw was near the mean of what we found here. And now, whenever we do the one sided t-test, we see that we have a p-value of 0.069. The alternative hypothesis is that the true mean is not equal to 11.066, etc. We have a 95% confidence interval that it is this range. And here is again the mean value. Now we see that this is a lot better because this p-value is much bigger than 0.05, that means the data is _not_ significantly different from 11.06, so 95% confidence interval means 0.5 for a p-value. As our p-value is bigger than that, our probability of being in here is even better than 95%.

Slide 42: Two Sample Student's t-test

Now we can also do a two sample Student's t-test. So, in this case, we're going to take paired data we can compute the values based on the paired set of data, and we can, of course, now draw conclusions based upon what the two sets of data mean.

Slide 43: Non-Parametric Data

We can also compute statistics on nonparametric data, which means the data isn't normal. And one of the most common methods for doing this is the Mann-Whitney U test. And so in R, we can use the function call for the Wilcoxon Signed Rank Test - which is very similar to Mann Whitney by now passing are dated in, passing our mu value in, and now we can see that (yes) we get a p-value of 0.4092 The alternate hypothesis is that the true location isn't equal to 11.066. And we got a warning message saying that the Wilcoxon with this argument can't compute that exact p-value, because there ties in the data.

Slide 44: Statistical Power

So what is statistical power? Which is, after all, what we started this discussion to talk about. Well, statistical power tells us how we can make inferences based upon how likely something is in the data. So, for example, in this case, 119 measurements were made up of a quantity in units of millimeters. We use the one sample Student's T-test, and we found the mean of the samples was 11.06 millimeters. So the question we ask is: "If someone else were to perform those measurements, how likely would it be that they would also find a mean 11.06 millimeters?". And that's the power of the inference. Right. It's about - can we take the experience that we have for one person making these measurements and is another person who repeats the same experiment the same number of measurements likely to find the same mean value?

Slide 45: Statistical Power – Effect size

So, to calculate the power, what we need to do is compute the effective sample size or the so-called "effect size". And generally, we split effect sizes into one of three values. Maybe we expect a small effect size or medium or large effect size. And we need some experience; we need some data to be able to decide how big an effect we expect. So if the expected difference is very small, we need a lot of samples. Typically, we will try to get a power level of at least 80%. So: Can we can compute it?

Slide 46: Statistical Power - Effect Size

Well, it turns out the effect size is associated with our data, it's dependent on the type of test that we're going to use, whether we are going to use analysis of variance (ANOVA), Student's t-test, proportion, chi-square, or whatever the test is. But once we know it, we can now calculate the effect size. The effect size is going to tell us how many samples we actually have to take to see with the power level of our data is.

Slide 47: Statistical Power – Calculate Effect Size

Well, fortunately, R has a package that we can use to get the effect size, and it's called power "pwr". So, in this case, we loaded our data and put it into a data frame. We load the library "pwr". And, in this case, we're going to call cohen.ES() - which is the conventional effect size described by Cohen in 1982. We said that we expect a medium-size effect. Now, the result of the calling this function is the tells that the effect size is 0.5.

Slide 48: Statistical Power–Calculate Power Given Sample Size

We are going to use that 0.5 [actually 0.2 - small effect size] now as our significance level when we call the power t-test ("pwr.t.test"), here we have input 199 samples we now add to the significance level we are using, we set it to type="one sample," and the alternative is "two.sided". We see that for a one sample t-test power calculation, N equals 119, d = 0.2 (and that was our effect size). We saw that for a significance level (sig.level) 0.05 that the power was 0.58 and that the alternative was two sided. That says with 119 samples, we have

only had a 58% power level, and that is not very acceptable, because we'd like to have an 80% or 90% power level. That means we need to do [have] more samples! So we could say, "Alright! What would happen if we change the effect size to medium." Then d would be equal to 0.5, and then the power level is 0.9997192 - that is way above 90%, so we would say it's acceptable. So from this data what we've learned is that if we thought it was a small effect (d=0.2) we don't have enough samples at a 119 but if we say the effect is expected to be medium size (d=0.5) then actually 119 [samples] was more than acceptable.

Slide 49: Statistical Power–Calculate Sample Size Given Power

So returning back again to the case where we have a small effect size. How many samples would we need to get a 90% power? So you put in a power of 90% the number in this case set to NULL - we don't know it is. We have said it's a small effect size. We want the same significance (95%) level. So now, what do we get? Well, it tells us that for a 90% power level, we need 265 samples - if the effect size is small. If the power levels is reduced to 80%, then, of course, we can reduce the number of samples that we need. If the effect size were actually medium (d=0.5), then the number of samples we need is actually only 44. And we see this is way less than the 199 that we actually had, so that means that with this [number of samples] that we don't have enough to see a _small effect_ but we certainly have enough samples from 119 to see a _medium size affect_. Now, I encourage you to read more about our analysis and think about a power analysis calculation to figure out how many samples you actually need to be able to see the size of the effect that you are expecting.

Slide 50: Writing your own functions in R

Next, we'll talk a little bit about writing your own functions in R, and I said earlier that you could think about running emacs, running a shell buffer within it, and typing in your R commands. And now, as you enter your commands, you can find out the sequence of commands you want to execute, group them into a function, name it, save it, and re-use that function.

Slide 51: Simple Functions in R

Well, that's fine if you want to sequence of R commands to be executed. What happens if I actually want to invoke a C program or if I want to be able to write a C program that involves R to be able to calculate the statistics as part of that program? We can do all of this!

Slide 52: Function to determine kidney function using a radionuclidic method:

So, in this case, who created a function called GFR1, it takes a set of parameters. We put some print statements in, so we output the information about each of those parameters - so if they're not provided, it will print messages telling us what we are expected to enter into this function. We calculate a bunch of values based upon those parameters, and we output an answer.

Slide 53: R Plot function

But now what we're going to do is we're going to create an R plot function. It's going to take a file name, and it's going to print the file name, it's going to create an X11 window to put the graphics in, and it's going so output into that X window the name of the file that we just passed, and it's going to compute the minimum and maximum of the data that are read to scale the plot, plot our data the Y label is going to be called the value, the X label is going to be a pixel, we set our limits for the range between the minimum and maximum we output our main title our subtitle, and now we go to sleep for five seconds. So it's going to output this data and then we're going to call it's a C system function- the system function sleep 5 - so the data will appear, then after five seconds it'll go away.

Slide 54: (Lung project)

So here's an example of that plot. We see pixels going across this way, values going across this way, and we'll see the data as we go across a particular line of an image. And this [plot] would pop up in a little buffer. And the nice thing is (yes) right now we have just been able to mix R and system functions in this case the C system function. And pass the command to it. But I can also call this function from a C program.

Slide 55: Call This Function from a C Program

So, in this case, I'm now writing in C. I sprint() into a buffer a string - I form the file name, I create the file, I can load my data file in. Again, I create the file name based upon the things that I've constructed with these parameters, "_profile". I read it in, and I write data, I close it. I run my program and violà; I can use the system command to now invoke R, passing it the R function that I want to have executed on the data that I just read it. And I dynamically can do that.

Slide 56: Form of rfile.input and rfile.sh

So, in this case, rfile.input contains the string. The shell file consists of echo $0 $1 - first argument; in this case, we remove the old output file. We run our program, in this case, we are passing the input file to R. We are putting the output in that and violà, we're all set. So it's convenient to be able to easily write scripts that are going to process the data that we want.

Slide 57: Normality Tests

Now, a very useful test is the test for normality. And the easy way to see that is a so-called quantile-quantile log, otherwise known as a qqnorm plot. And what this is going to show is we're going to have the data being tested on one axis, and the corresponding quartiles of the normal distribution plotted on the other [axis]. And we're going to look and see this does this fall along a line. Now a density plot is basically a smoothed version of the histogram. And we can also plot the histogram and the density plot to show the population shape. And we

will plug the qqnorm and a boxplot, and that's going to make it very easy is for us to see the outliers. So what does this do?

Slide 58: Inference – Check data normality

Well, we create a function called eda.shape. We create a two by two array of plots - one of which is a histogram, a box plot, we calculate the density plot, we compute the qqnorm plot, we add the quartile-quartile line, and off we go.

Slide 59: Result of simple check for Normality

Here's a simple test. So we had - this is the histogram. So these are the values of X. those were their frequencies. We see the smoothed density plot here. We see the box plot of our data here. And now we have the quartile-quartile plot here. And the qqline along this. We see some outliers at this end and some outliers at that end, but basically most of our data falls along what we would expect given this normal distribution.

Slide 60: Inference #2 – put a title above a multiple graph

Now, we can, of course, get fancier because we can add the label above the histogram saying "Histogram". We can put a label above the boxplot, put a label about the density plot, we can add various text, and set the style.

Slide 61: Result

So now here we have our main title, histogram, box plot, density plot, normal Q-Q plot (which we see we actually had done in our previous plot), but we didn't have the box label. Here in this later version, we now have it saying "Boxplot".

Slide 62: Compare the two versions

So our first version looks like that, while our second version looks like this. This an example of a figure that I might think of directly putting into a paper about this experimental data.

Slide 63: Call a unix program from an R function

Now, we can also call unix programs from an R function. So, in this case, we are writing in R, and at this point, we're going to invoke unix passing it this information: "test_file", filename. We now give it the data that we're going to process - which we scanned in from this array. If that is true, then we execute this, otherwise, if it is false, then we take the data from this other file, and we put it into the data frame. So we're looking at if that's true, we use this data frame; otherwise, we get our data and will put it into our data frame. We can do that many times. We can do if else's, etc. around it. So the result here is we see that it's very straightforward to call any unix program we want from inside of an R function.

Slide 64: "if" Statements and "for" Loops

IF THEN ELSE statements work as we expect. IF with blocks ELSEs with blocks. You can have FOR loops. etc.

Slide 65: Data analysis

[Prof. Noz]

Slide 66: Consider the case of 6 measurements of 7 patients

Okay, let's consider the case of 6 measurements of 7 patients. Now the 6 measurements were 3 rotations and 3 translations. So, what we want to know is - is there a dependence within the group of seven patients? Is there a dependence within the group of 6 measurements? Or is there a dependence between the measurements and the particular patients. And the way to do within and without measurement is to look at an ANOVA.

Slide 67: Data Analysis – Linear Regression

Now, before you begin an ANOVA, you have to set up your -- read in your data and set up factors - which are the basis of your data. So, in this case, we have the 6 measurements, and we have 7 patients.

Slide 68: Data Analysis – Linear Regression

So we set up our data in a 6 by 7 array. And we have to form a data frame that is basically like this. So we would have this many many times for 6 times 7 array.

Slide 69: Data Analysis – Linear Regression

Now, we look at our data analysis - first, we say, "Well first we're going to graphically examine the data". We want to know if the data itself has any problems. So the first thing we do is look at the mean and the median of both the patients and the measurements separately, and then we plot the data - the patient data dependent upon the measurements. And then, we look at an interaction plot, which is a plot of the measure and the patients, and then we look at it in reverse a plot of the patients and the measure.

Slide 70: Data Analysis – Linear Regression

And it looks like this. The first - the top two plots - give you the data mean and the data median for the measurements separately from the patients. And the second set of plots (in the middle) tell you the measurements and the patience and how the data is constructed - so you have box plots to see if there are many outliers or how the data falls. And then, the bottom two plots give you the interactions. And if there is an interaction between the measurements and the patients or between the patients and measurements, it will not - you will not be able to do a successful ANOVA.

Slide 71: Data Analysis – Linear Regression

Now, the structure when we have when we do an ANOVA - we just have a simple function in R that is called aov(). We can use this function in two ways. We can say of the measurements - there's an interaction between the measurements and the patients. And we certainly want to know that that's independent. And then we look at: Is there a difference between the factors? Is the measure itself independent of the patients? And that's an important point because if it's not, then the inferences were going (to join) to infer from the data could be wrong. In the first case, you can see that the summary of the ANOVA doesn't even give us an F value or p-value. Whereas in the second case we see that the measurements are highly different and one would expect that because we're looking at a coronal angle, a sagittal angle, and an axial angle and three translations - so you would expect the measurements to be all different and the p-value is teeny tiny saying that they are independent. The patients, however, do not depend upon the measurements - as you can see - because the second line is _not_ significant. So, we are assured that the measurements are independent of the patients, and the patients are all the same [i.e.,] have all the same problem - so we expect that the patients will be somewhat alike.

Slide 72: Data Analysis – Linear Regression

Now, once we get the ANOVA, one of the important things in dealing with linear regression is to examine the residuals. If the residuals are okay, then you know that your ANOVA has been successful - that it's giving you the right answers. Now, to do that, we used a histogram plot a qqnormal, and what is called a fitted plot.

Slide 73: Plots of the residuals

And you can see them plotted here. You can see the histograms of the residuals are very closely grouped around zero. You can see from the Q-Q plot that it is a fairly straight line, that there are not many outliers. (But) And the fitted values you can see if they are well scattered. So there shouldn't be any dependence between the different measurements.

Slide 74: Multiple Plots 1

Now, one can do multiple plots - trying to summarize this data. And you can see a long R function here. Which we just wrote from different things and

Slide 75: Multiple Plots 2

it goes on and on -

Slide 76: Use as: eda10.shape(file1,file2,file3, file4)

but this is something that you might want to, for example, put in a publication that summarizes your data. Now, in this case, it was a precision experiment where we had 10 patients, and we also had 24 model studies, and we had 2 readers for the patient part of the

data - and we wanted to check that the two examiners got the same value for each (of the boxes) for each of the patients that they examined. We wanted to know what the interdependence of the examiners were. Where they independent or dependent? And then we drew Q-Q plots for each of those characteristics. So you see on the density plot we have the model studies are the first one, and you can see the type of line. And then and the data for examiner one is plotted in this matter. The data for examiner two is plotted here. And then the inter-examiner dependence is here. And this gives you a good visual picture of what you did in (your) both in your test and what the results of the test are.

Slide 77: Multiple ECDF functions

And you could also have multiple EDCF functions - which we have talked about before. This gives you the cumulative empirical distribution functions. And in this case, I took the original edf, and I modified it to do something that _I wanted it to_.

Slide 78: Result with multiple ECDFs

And this is the result. And you can do this in R, which is very nice. You can take a function that they have and modify it to suit your own needs. Now, some of this I have to caution you you might want to consult a statistician about at least _what is the right way to examine your data, but the nice thing about R is that once you understand how to examine your data you can find almost anything that you need in R or one of the R packages and you can just go off and use it with good knowledge that it will be correct. Provided use, of course, correctly. Another nice thing is the packages are usually signed by people, and actually you're able to send them an email, and they actually answer some. It's really nice.

Slide 79: References

And there are many many more references that you can read about this.

Slide 80: References

And yet more references.

Slide 81: Additional R References

Additional R references can be found here and fitting distributions with R, there's a nice description about this by Vito Ricci, which you can take a look at. Good luck doing the statistical analysis that you would like. And most importantly, thinking about how many samples you actually need to have to see the effect of the size that you expect.