Lab 8

Univariate Statistics with R

Three different activities this week! These may require a bit of thinking; we've tried to include enough information to help you find solutions, but 'our' solutions aren't the only ones, obviously.

A brief note on `with()`

You may have noticed a really useful function in the solutions for last weeks lab - the `with()` function. The way in which this function works is to allow you to specify a dataframe that an expression uses and therefore not need to include it when specifying variables. See the following example:

```
cor.test(schools$iq, schools$exam)
```

This command can be written as:

```
with(schools, cor.test(iq, exam))
```

This is a nice trick that can make your code more readable and also save time and effort when you have an expression where you are referring to a dataframe multiple times. Take this code which will print any values of the `iq` variable in a dataframe called `schools` that are above 2.5 standard deviations from the mean:

```
schools$iq[schools$iq > mean(schools$iq, na.rm = T) + 2.5 * sd(schools$iq, na.rm = T)]
```

This could be written as:

```
with(schools, iq[iq > mean(iq, na.rm = T) + 2.5 * sd(iq, na.rm = T)])
```

`with()` is essentially a convenience function. Make use of it if you feel it produces more readable code and saves you precious time and typing energy!

Last Week's Linear Model

Task 1: Start by opening last week's (Lab 7) project. Since we're continuing to work on the same data, it's easier to extend the same project.

Task 2: Re-run the code you ran last week to load and merge the school datasets. (Just select the relevant lines of code and hit Ctrl-Return, or Command-Return on Mac).

Don't have the relevant R code saved? That's a problem. It really is important to save code, in scripts, when you use R, so that you (and others) can replicate work you've already done. Statistical analysis is as much a *process* as an *outcome*. For now we can overcome this by loading a version of the data that has been merged already - create a new script file now and in the first line, type `load(url("http://is.gd/tsktsk"))`; put the cursor on that line and send it to the console (Ctrl-Return or Command-Return on Mac). Remember: Use the script editor from now on!

Task 3: Create a linear model and inspect it by typing the code below (my merged data is in `schools`; you may need to edit the code appropriately).

```
model <- lm(exam ~ iq, data = schools)
summary(model)
```

Tip: Models may vary very slightly from person to person, depending which outliers you decided to remove, etc.

Task 4: Which is the intercept? Which is the slope? How can you interpret each of them?

Task 5: You may conclude that the intercept isn't very meaningful! Create a new linear model, `model2`, with a more meaningful intercept. ***(Take a look at slides 9 and 15 from this weeks lecture for some guidance with this)***

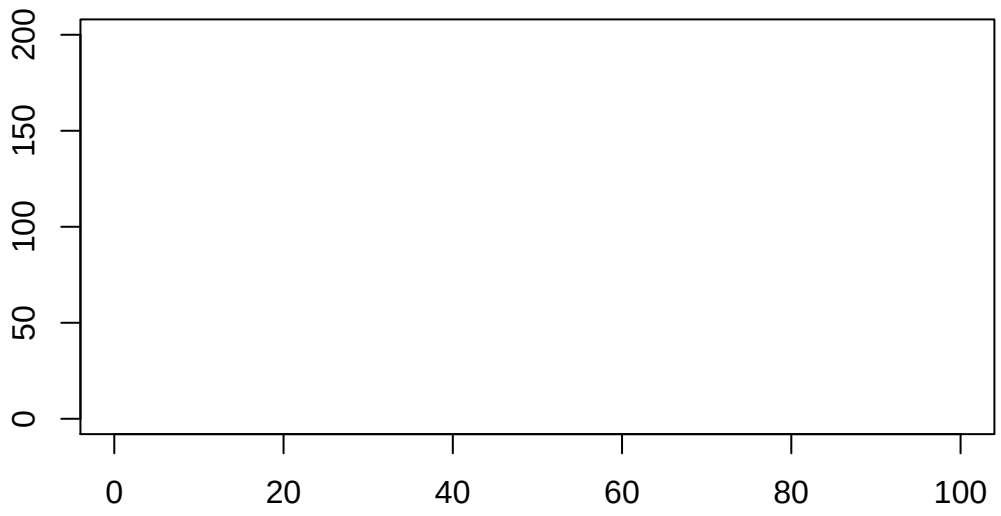
Tip: Arithmetic operators such as `+` and `-` have different meanings within a *formula* (the part which represents the relationship between variables, such as `exam ~ iq`. To 'isolate' arithmetic from the formula intepretation, use `I()`, so `exam ~ I(iq + 23)` means "`exam` is predicted by `iq + 23`"; `exam ~ iq + 23` means "`exam` is predicted by `iq` and the number `23`" (whatever that means!).

Task 6: What does the intercept of `model2` tell us? And the slope?

A Pretty Graph

Useful Stuff To Try First

Task 7: Create a blank canvas, like so:



###

Task 8: Create a vector `x` of values between 0 and 100 in a random order, and another vector `y` which depends on `x`, like so:

Task 9: Now try drawing lines, connecting each point (x, y) together. What will you get?

Hopefully you'll have ended up with a mess, because the lines will have joined each point (x, y) *in the order they appear in the vector*. Take a look at the first 5 values in each vector (remember, your values will differ from mine):

```
## [1] 92 93 29 81 62 50
```

```
## [1] 197.21925 188.16161  83.75728 178.42899 134.89761 112.76551
```

The line will start at $(92, 197.2)$ and be drawn along to point two $(93, 188.2)$, and then to point three $(29, 83.8)$, and so forth. Given that these points are in a random order we get a *squiggly mess*. However, if the values for both vectors were ordered such that the x vector went from the lowest value to the highest value the line created would be much more interpretable, see fig. 1.

Task 10: Can you use `lines()` to recreate the plot in fig. 1 using the `x` and `y` vectors created above? *Use the tip below!*

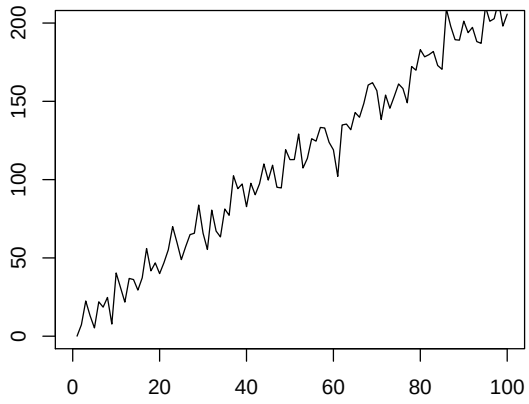


Figure 1: A simple line graph, made with 'lines()'

Tip: There are various ways of doing this, but the most ubiquitous is to use `order()`. Try `cbind(x, order(x))` to see what happens. Now try `x[order(x)]`. How does this work?

Task 11: Try adding two lines to your graph simultaneously. First create some more data (use the examples below, or your own), and bind the values together in a matrix:

Task 12: Now look at the help for `matlines()`. Can you add lines to the graph from fig. 1 so it's like the one in fig. 2?

Task 13: One more thing to play with: Try the following (note, using `model` rather than `model2`):

```
p.exam <- predict(model, schools, interval = "confidence")
head(p.exam)
```

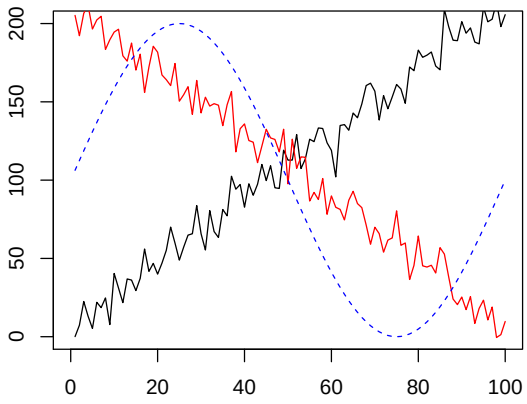


Figure 2: A simple line graph, made with ‘lines()‘ and ‘matlines()‘

You'll find that `p.exam` is a matrix, consisting of three columns: `fit` is the fitted (predicted) value of `exam` based on each pupils `IQ` score; `lwr` and `upper` are the upper and lower bounds of the model confidence interval for each value of `schools$iq`.

Have a look at `?predict.lm` for much more on `predict()` (and, if you feel so inclined, try `?predict` to see where `?predict.lm` comes from!).

Task 14: Can you put everything in this section together and recreate the graph in fig. 3?

In this figure we have the basic scatterplot between actual `iq` and `exam` values along with the regression line that describes the linear relationship between the two (and the upper and lower estimates at each point).

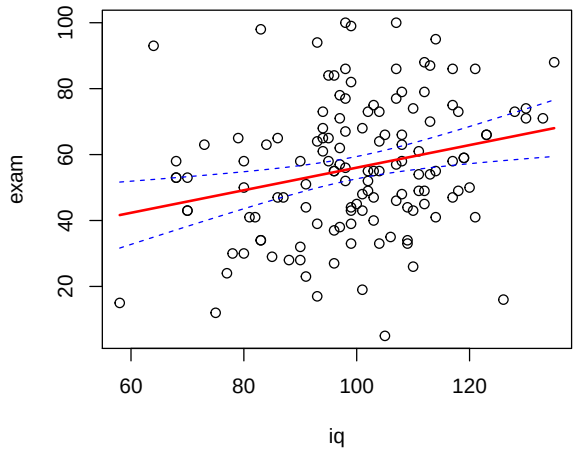


Figure 3: Best fit line and model confidence predicting ‘exam‘ from ‘iq‘

More on Regression

Earlier on you estimated the effect of iq on exam performance, and found that there was a modest effect. However, that isn't the full story: After all of the students in the study had died, Professor Frank N. Stein measured all of their brain volumes, with the idea of discovering whether brain volume was a useful predictor of iq or of exam performance.

Task 15: Load the data for this week from the Learn website, and merge it with your existing schools dataframe.

Tip: As the data is already in an R format (.Rdata) you can use the `load()` function.

No tricks this week, the merge should go straightforwardly. Your merged dataframe will have an extra column in it now, called `bvol`, for brain volume in cm^3.

Task 16: Does brain volume predict iq?

Tip: You need to build the relevant linear model, using `lm()`

Task 17: Are you sure? i.e. is the model appropriate?

Tip: Have you looked at your regression model carefully, using some of the model criticism tools you saw in the lecture? Functions you want to think about include `residuals()`, and `plot()` (where `plot()` is called on a linear model object)

Task 18: We already know that `iq` predicts `exam` performance, because we tested and graphed the relationship earlier (and in the previous lab). However, does brain volume also predict exam performance? More importantly, does brain volume help predict exam performance *over and above* what `iq` already predicts?

Tip: You'll want to test whether adding a new predictor improves on chance (using `anova()`) as part of your answer... *Part 2 of the lecture will help you here.*