STAT0023 ICA2 (2018–19 SESSION) — GENERAL FEEDBACK

This assessment was intended to be challenging: it's the first time that most of you have been asked to produce an extended piece of statistical work on this scale. This is partly to compensate for the straightforward Moodle quiz assessments. We recognise that it's difficult, and this is why we provided a lot of support for the assessment — e.g. hints in the assignment instructions, many office hours, responses to discussion forum questions, feedback from last year and so forth.

Overall, the standard of submissions wasn't as high as I'd hoped. There were a few excellent ones, and I know that many of you put a lot of work into it. It did look, however, as though many students didn't take the advice that was given. This was particularly true in the reports: I saw very few really good exploratory analyses, and the choice of graphics was often a bit strange as well.

In this general feedback, I'll comment on the coding, on your reports and then say something about the prediction performance. Please look at my specimen answer (both script and report) to understand some of the points that I'm making, both here and in your individual feedback. E.g. if your personal feedback says "Your graphs are clear and well labelled, although some more creative choices would have revealed more structure" then you might not immediately know what "more creative choices" might mean: look at the specimen solution (look at the graphs, and also read the text carefully), and think about how much more information can be packed into a small space if you're creative about it.

Your individual feedback and provisional grades can be found on the ICA2 Moodle page: for each of you, the feedback is provided in a file called `Feedback.txt` which should be visible alongside the script that you submitted. If you can't see it, the reason is that it has gone off the right-hand edge of the Moodle tab: in this case, the only way I know to reveal it is to make the text smaller by pressing `<Ctrl>` and `-` (that's "Control-minus"). The feedback consists partly of a set of automated checks, and partly of some individual comments that are intended to help you see what you did well and what you did less well.

Coding

For the first time this year, somebody (actually three people) used SAS for this task: this was quite bold! Everyone else used R. I was pleased to see that *almost* everyone is comfortable using their chosen software to read and manipulate data, carry out basic — and in some cases, quite advanced — statistical analyses, produce graphs and so on. If you got a 'B' grade or above then you have the computing skills that will enable you to survive if you get a job involving analytics in some way. Conversely, if you got a 'C' or below then you need to work on your computing skills if you're aiming for this kind of career.

Many of you obviously spent a lot of time Googling, to find out how to do things: this was true of both the R and SAS users. Internet searching is fine, *so long as you understand* *what you're doing.* There were, unfortunately, many cases this year where people who found material online and used it uncritically: it wasn't always successful. As with fake news, you shouldn't believe something just because it's online: the contributors to StackExchange are probably not working out of troll factories, but they're not all experts, either! Be aware. This also applies, incidentally, to add-on R libraries. Some of the add-ons from CRAN are not very efficient: they're contributed by volunteers, some of whom are excellent programmers and others less so. Or perhaps the packages are designed to cope with problems that are way more complex than the one considered here. As with anything that's freely available on the internet, you have to be wary. Anything that comes with the core R installation will be better than you or I could ever write: if it's a free add-on though, take care!

On the subject of add-on libraries: people used a huge range of libraries for this assignment. Few of them were necessary or desirable: indeed, some of them made your code phenomenally slow and inefficient (I think I'm going to ban the use of `dplyr` next year, it seems to have been written with the deliberate aim of making everything much slower and more complicated than necessary). In my own specimen script, I used just two: `RColorBrewer`, for its carefully-constructed colour scales, and `maps` in order to draw a map of Afghanistan. My script doesn't use any technique that hasn't been taught explicitly during the course. The challenge in a task like this is not to use very complicated techniques: it is to be creative but organised.

On the subject of packages: if you use add-on packages, it is helpful to put your `library()` commands right at the start of your script (see my specimen answer). The reason is that if you then come back to your script later — or if you give it to someone else — they can immediately see what's needed to get the script to run. If your `library()` commands are buried in the middle of hundreds of lines of code, it can be hard to find them.

For the SAS users: it's hard to give general feedback on your coding given that there are only three of you. You've got individual feedback files on Moodle though, and I'll be happy to answer any questions that you may have. The general feedback on reports and predictions is still relevant, though.

My other general comments on R coding are as follows:

- There is a difference between *installing* and *loading* libraries. 'Installing' a library means obtaining it from an external source and putting it on your computer: you shouldn't need to install a package as part of your script. 'Loading' it means making it available to your R session after you've already installed it (usually using the `library()` command).

- Many of the scripts showed little evidence of the programming principles learned in the first half of the course. A script is *more* than a sequence of commands that you've managed to get to work by pasting them consecutively into the command prompt. A good script shows structure and design, and (typically) uses the kinds of programming constructions that you have been taught. For example, my specimen script has a clear structure; it is well commented to create a 'narrative'; and it uses functions to implement tasks that are needed repeatedly. It *certainly* does not contain any `View()` commands within it, as several of your scripts did: if I want to inspect a data frame then I can do it in interactive mode, but I don't need to do it every time I run a script.

- If you want to check that an R script works, ideally you should do it from a 'clean' start: restart your R session (via the `Session` → `Restart R` menu in Rstudio), then `source()` your script. This will enable you to check that you have created all the objects you need, by the time you need them.

- If you insist on using `ggplot`, you need to enclose your plotting commands with `plot()` if you want to guarantee that your plots will appear when your script is `source()`d. I penalised people for not doing this, because you have been taught explicitly to do it.

- In the dataset that was provided to you, missing haemoglobin values were denoted by -1. When you read data with 'dummy' missing value codes like this, the first thing you should do is to set them to `NA`. If you don't deal with them *immediately*, there's a risk that you will accidentally forget about them later and treat them as genuine numbers. Some students *did* forget this, and produced plots in which all of the -1's appeared — not helpful!

  I saw some bizarre and inefficient ways of ensuring that the -1s were set to `NA`, many involving the `dplyr()` library (did I mention that already?). The efficient, no-nonsense way to do it is to go

  ```
  AnemiaData <- read.csv("AnemiaData.csv", header=TRUE, na.strings="-1")
  ```

  If you tried anything more complicated than this, you should probably go back and do the Moodle quizzes for Week 1 a few more times.

  Another benefit of setting the missing values to `NA` is that they will be handled automatically by R in any subsequent analysis, without your needing to split the dataset into two parts (as many students did). This avoids your having to create unnecessary copies that occupy memory.

- In applied statistical work, you will often want to create new variables from existing ones — for example, by creating groups from an 'age' variable. When you do this, *do not overwrite the original variable*, unless you are *absolutely* sure that you're not going to need the original variable again. A few people got into trouble because, for example, they replaced the `Age` variable (which was originally numeric) with a factor representing age group; and then pasted my specimen clustering code from Lecture 10 in which `Age` was assumed to be numeric. The result was a script that failed to run when `source()`d — for which you were rightly penalised. If you want to do this kind of thing, in general you should use a new variable to store the new quantity (again: unless you *know* that you have no further need for the original).

- If you want to ensure that your saved graphics files will match what you see on the screen, it's helpful to ensure that your graphics devices (screen windows and saved files) are all the same size — otherwise you might find that text disappears when you save the file because it won't fit in the space available, or that the plots get squashed up unexpectedly so that you can't really see the patterns. I usually do this using the `x11()` command to open a graphics window, and then `dev.copy() ... dev.off()` to copy the contents of the window to a graphics file of the same size. There are several examples of this in the workshop scripts that you have been given. An alternative approach, which several of you used effectively, is to open the required graphics file directly using the `pdf()`, `png()` or `jpeg()` commands.

- If you're going to produce many similar plots, it is often helpful to use `par(mfrow=...)` to create an array of plots directly on your graphics device — many of you saved each plot individually and then pasted them into your reports, which can be messy. One potential problem with arrays of plots is that the default plot margins in R can be very wide if you have several plots on the screen: you can use `par(mar=...)` to change these defaults, and `par(mgp=...)` to move the axis labels closer to the plots so that you can use the space more efficiently. See my specimen script (and scripts that you've been given in workshops) for examples.

- A Gaussian GLM with identity link is just a linear model: if you use `glm(..., family=gaussian(link="identity"))` therefore, you'll get the same model fit as if you used `lm(...)`. **BUT** it will be slower, because it doesn't exploit the known features of the linear model. Effective and efficient use of statistical software requires awareness of this kind of thing. Actually, there *are* situations in which you might want to use the `glm()` version of this command — for example, if you want to compare the AIC values of models with different link functions — but my fundamental point is that you need to think about what the button does before pressing it.

Reports

Overall, I'm afraid the quality of the submitted reports wasn't as high as I was hoping. As noted above, there were few good exploratory analyses — and, actually, few scripts that used background knowledge effectively to inform the analysis and modelling. I'm not sure what happened here: there were plenty of hints in the lecture, in the assignment instructions, on the discussion forum, during office hours and (in one instance) during a workshop where I had a bit of a rant about collinearity.[1] Despite all of this, many of the submitted reports suggest that we're somehow not managing to get through. From a teacher's perspective, this is quite disappointing — and also a bit perplexing.

Specific comments on the reports are as follows:

[1]If you weren't there: the rant went more or less along the lines of "Why is everyone so obsessed with collinearity? Its effect is mainly on the standard errors of your coefficient estimates (as measured by VIF); but if you've got small standard errors already then who cares? What matters is the relationship between covariates and response." The 'sheep energy' example from Workshop 9 was designed to help you understand what collinearity is about. It's a myth that if two covariates are correlated then you should *definitely* not consider them both in the model: unless they're *perfectly* correlated, then each contains some information that isn't in the other. And I can't really see the point of calculating VIFs. All they tell you is how much smaller the standard errors would be if you had a different dataset: this is largely irrelevant, given that you *don't* have a different dataset. They are, perhaps, useful for identifying the parts of a model that are most affected by collinearity, but I wouldn't go further than that.

- When you have limited space to present an argument, you can't afford to waste it on things that aren't absolutely relevant. So you need to think carefully about what are the really important messages, and to focus on them. As an example: many students wasted space in their reports repeating material from the question. Others wasted graph / table space on non-essential plots. For something like this, you don't need to provide *all* of your plots, just the ones that illustrate your key messages. Moreover, you can often illustrate these key messages without taking up too much space: this does require that you *design* your graphics carefully though, in order that the messages can be seen 'at-a-glance'. In my specimen report, none of the graphs uses techniques that you haven't seen before: I'm just using them creatively.

- Conversely, many of you simply didn't include enough graphs, so the reader of your report couldn't really get a 'feeling' for what you were talking about. Well-chosen graphs are much more effective than verbal descriptions for communicating complex messages: they can also provide some reassurance that you've thought carefully about the problem.

- Still on the subject of graphics: colour is for communication, not decoration. I saw several pastel-rainbow boxplots in your submissions: this is very pretty, but a wasted opportunity. In my own report, boxplot colours are used to indicate sample sizes (and the colour scale is chosen carefully so that the *intensity* of colour varies, thus ensuring that the plots will work in black and white or to a colourblind person).

- One of the 'difficult' features of this particular dataset was the presence of outliers — or, equivalently, of a heavy-tailed residual distribution. There were several approaches to dealing with these: at one extreme, some of you pretended they didn't exist (e.g. 'the Q-Q plot shows that the residuals are approximately normally distributed', which simply isn't true), while at the other, several of you removed the outlying observations and then claimed that your model was a good fit to the data (what a surprise!). In general, you shouldn't remove outliers unless you're confident that they're wrong, or unless you're prepared to make a clear statement that your model is only designed to fit a subset of the data. In this assignment, removing outliers was quite a dangerous thing to do because there will still be outliers that you *haven't* removed in the prediction dataset. Your predictions were assessed using a score that rewarded you both for predicting accurately and for providing realistic error standard deviations: if you've removed outlying observations, your error standard deviations will probably be too small and therefore your prediction score will suffer.

- Another difficulty with this assignment is the low predictability of the results: nobody managed to find a model that explained much more than 20% of the variation in haemoglobin levels. Some of you acknowledged this, others tried to pretend that your model had a lot of predictive power. Don't pretend! It's important to acknowledge when something is hard to predict.

- In applied statistical work, it is usually helpful to use your understanding of the context to inform a model-building exercise: don't just rely on the statistics (p-values etc.)

without stopping to ask whether they make sense. This is one of the disadvantages of automatic model selection methods such as stepwise regression: these methods don't know what makes sense in the context of the problem. Conversely, if the statistics suggests that a particular effect is very important, it's worth asking yourself what might be driving this. In this assignment for example, the main source of systematic variation is the province in which a woman lives. This is perhaps unexpected and therefore worth commenting on, but few students attempted to provide an explanation for it. Some suggested that it is related to regional differences in prosperity, but if this were true then you'd expect to see relations with the variables representing wealth and assets in the dataset. The geographical variation in haemoglobin levels is *not* economics-related, therefore. Other students suggested that it is altitude-related: again, this seems unlikely (draw a map). There were only two alternative explanations: one (supported by evidence from the literature) was that it is related to flooding potential and the resulting risks to sanitation, while the other is in my own specimen report. Of course, we can't be *sure* that any of these potential explanations is correct: we should at least think about it however, because our job is to try and interpret the data.

- On the one hand, as noted above many students relied too much on the statistics without considering the context. On the other, there were also many who relied on the context without really considering the data. This was a particular problem in the 'exploratory analysis' section, where several students gave a good review of previous literature and (in some cases) their own beliefs about what should be important — but who made no effort to investigate whether the data supported these beliefs. Good applied statistics is certainly informed by our understanding of the context, but we should never let this dictate everything: rather, it should inform the questions that we ask, and we should then examine the data carefully and with an open mind to answer these questions.

- Many of you considered interactions in your modelling, which was good. From the comments in some of your reports however, not everyone really understands what interactions represent. Many of you seem to think that an interaction arises when one covariate influences another. This is incorrect: interactions are about how covariates combine to influence the response variable. You can't think about interactions without thinking about the response, therefore.

- There were a few cases where people had tried a reasonable method, obtained results that were obviously wrong, and concluded from this that the method didn't work for this problem so they needed to use an alternative. This wasn't very common, but it's worth a comment. Thus: if you try a reasonable method and you obtain results that are obviously wrong, then it's *very likely* that you made a mistake. Don't blame the method, therefore!

Predictions

In your individual feedback, you will all find your prediction score and your rank in the class. I have also calculated your root mean squared prediction error, defined as

$$\text{RMSE} = \sqrt{\frac{1}{1039}\sum_{i=1}^{1039}\left(Y_i - \hat{\mu}_i\right)^2}$$

using the same notation as in the question sheet. This is a more common measure of prediction performance than the score S that I used: the reason for using S is that it takes account of your prediction standard errors as well, and is able to reward those of you who gave an honest assessment of how accurate your predictions would be.

If you're interested in knowing what the real values were for your predictions, I have uploaded a file `AnemiaData.rda` to the Moodle page. If you load this into R using `load("AnemiaData.rda")`, you will find a data frame in your workspace called `AnemiaData`: this is the complete data set, in which the -1's in the haemoglobin column are replaced with the actual values.

There was *very* substantial variation in prediction performance. One of you achieved a better S score than me, and 10 of you achieved better RMSEs than me — very good! There were also some much less accurate predictions. Figure 1 shows your scores and RMSEs for all scripts, also showing the performance for my specimen answer. There is a massive range of scores, so the successive plots in the figure 'zoom in' to enable you to see more relevant detail.

Because of the huge variety of different models that you came up with, it's very difficult for me to say anything conclusive about which covariates were best. However, I *have* grouped your submissions according to the type of model. Figure 2 shows the prediction performance for each model type. The top panel shows all scores below 50 000: you can see from this that several different model types were able to produce both good (i.e. low) and bad (i.e. high) scores (there are some model types with no points on the plot: for these model types, there were no submissions with scores below 50 000). The second panel zooms in on the left-hand end of the distribution: here, you can see that the best score was obtained from a Gaussian GLM with a log link. The same model also achieved the best RMSE in the class, as shown in the bottom panel of Figure 2.

One obvious feature of Figure 2 is the range of scores obtained by people using linear models. In many cases, this was due to prediction error standard deviations being calculated incorrectly. I realised, however, that the 'hint' about how to calculate these standard deviations for linear models in the assessment instructions is potentially unclear: it could have been misinterpreted as meaning that you just have to add the argument `se.fit=TRUE` to your `predict()` command, and this isn't true. If you *did* misinterpret the hint like this, your calculated error standard deviations would have been much too small and you would have got a very large score. Really you should have spotted that your standard deviations were too small. However, I have been fairly generous in allocating marks for the prediction scores, in recognition that the hints could have been clearer.

It's also worth pointing out that the overall standard deviation of haemoglobin levels is about

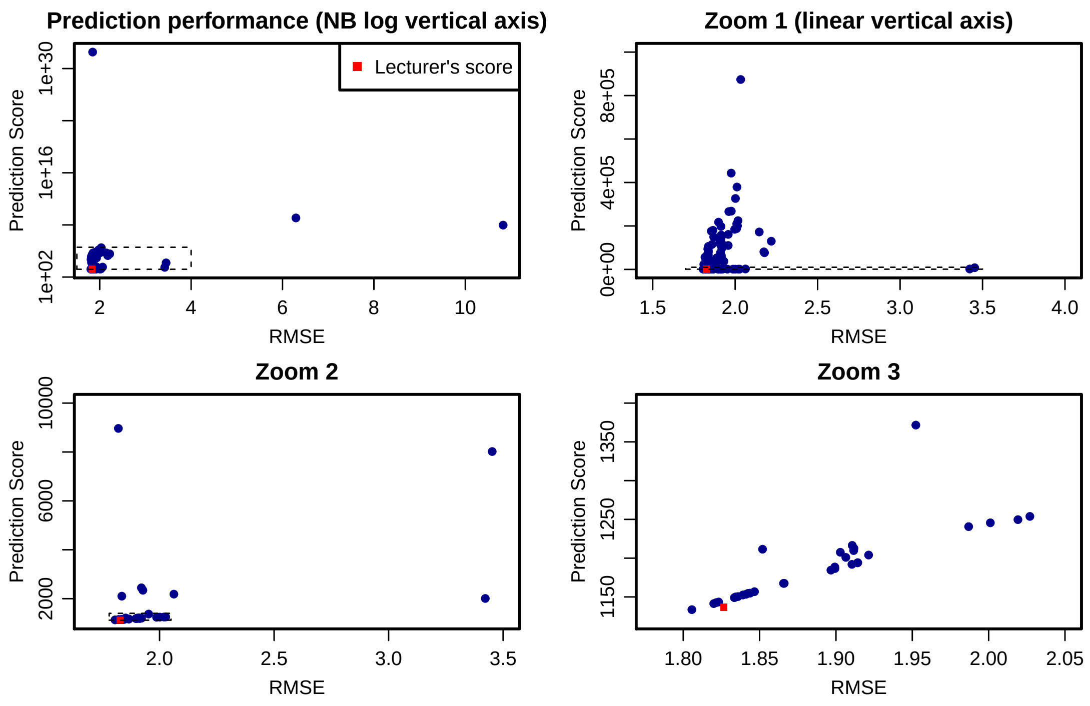

Figure 1: Performance of the predicted haemoglobin levels from the submitted scripts. The horizontal axis in each plot represents the root mean squared error of the predictions, and the vertical axis is the score S that was used to assign marks for your predictions. The top left panel shows the performance for all submitted scripts: the remaining plots zoom in on successively smaller regions corresponding to better (i.e. lower) scores. In the second and third plots, the 'zoom' region is indicated by a dashed rectangle.

2g/Dl: anyone with a RMSE greater than this is doing worse, therefore, than somebody who didn't build a model at all. If you are among the 20% of the class who achieved this, you should probably feel a bit embarrassed — and be very careful in the future when building models, particularly if you're going to use them to do things like predict investment returns!

Some of you may be interested in what decisions were taken by the best-performing scripts. Here is a summary of the decisions for the top five, in order.

Script 1 (score 1 133, RMSE 1.806). This student used a Gaussian GLM with log link. The chosen covariates were `Pregnant`, `Rural`, `HHUnder5s`, `Electricity`, `RecentBirth`, (clustered) `Ethnicity`, and `Province`, along with the interaction terms `Province:Rural`, `Pregnant:Electricity` and `Pregnant:Province`. Ethnicity was clustered into three groups, but the candidate's report doesn't say what these groups are.

Script 2 (score 1 141, RMSE 1.820). This student used a linear model (fitted in SAS!), with a very large number of covariates and interactions (too many to list here). `Age` and `WealthScore` were each categorised into three groups.

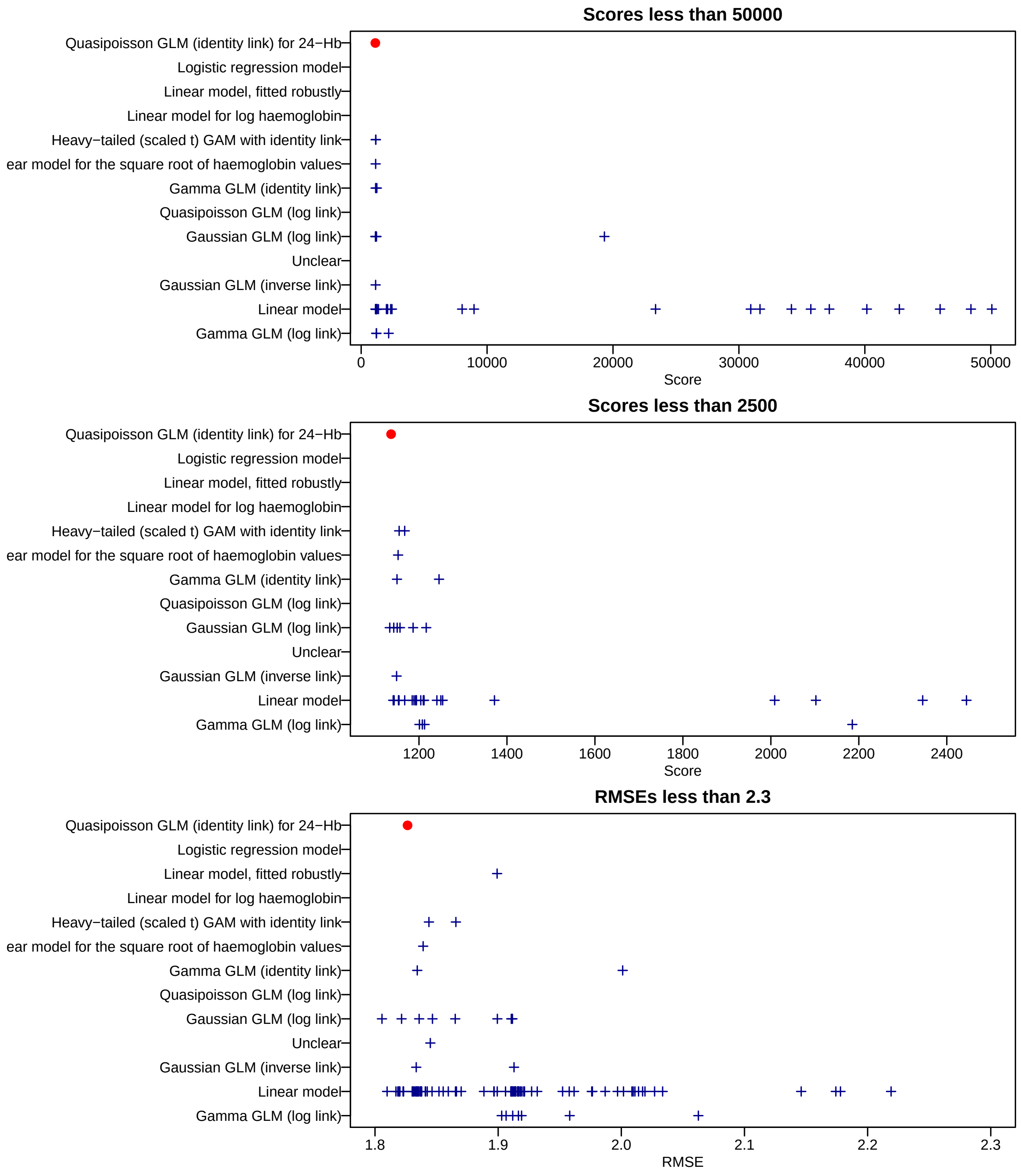

Figure 2: Prediction performance by model type. Each cross represents one student's submission: the red dots are from the lecturer's specimen answer. The top two plots show the distributions of the score S that was the basis for the marking scheme: the top plot shows submissions with scores below 50 000, and the second plot shows submissions with scores below 2 500. The bottom plot shows the root mean squared error (RMSE) by model type, for submissions with a RMSE below 2.3.

Script 3 (score 1 143, RMSE 1.822). This student used a Gaussian GLM with log link. The chosen covariates were `HHUnder5s`, `Province`, `RecentBirth`, `Pregnant` and clustered `Ethnicity`, with interaction terms `Province:Pregnant` and `Province:RecentBirth`: Ethnicity was clustered into three groups: Dari / Turkmen, Pashto / Uzbek and Other / missing.

Script 4 (score 1 143, RMSE 1.823). This student used a linear model. The chosen covariates were `Education`, `RecentBirth`, `Province`, `WealthScore`, `Pregnant` and `Ethnicity`, with an interaction between `Province` and `Wealthscore`.

Script 5 (score 1 149, RMSE 1.833). This student used a Gaussian GLM with inverse link. The chosen covariates were `RecentBirth`, `HHUnder5s`, `TreatedWater`, `Electricity`, `Rural`, `Province`, `Pregnant`, `TotalChildren` and `Ethnicity`. No interactions were included.

My own score and RMSE were 1 137 and 1.827 respectively. I used a quasipoisson GLM with `24-Haemoglobin` as response, and with an identity link function. The chosen covariates were `WealthScore`, `CleanWater`, `Sheep`, `RecentBirth`, `Pregnant`, `Ethnicity` (with Pashto and Turkmen groups merged), `HHEducation` (with 'None' and 'Primary' levels merged), and `Province`. I included interactions between grouped `Province` and both `WealthScore` and `CleanWater`; and between `Ethnicity` and `Pregnant`. For the interactions, five groups of provinces were used.

There is quite a lot of consistency between all of these top-performing models: except for mine, they are all based on a Gaussian distribution with constant variance, and they use similar covariates although they have different link functions (identity, log, inverse). All of the models use `Pregnant`, `RecentBirth`, `Ethnicity`, and `Province`, although they don't all have the same interactions. In view of the conclusions from the 2016 study, it's interesting to note that most of these models do *not* include `Sheep`. Perhaps our collective exercise has cast doubt on the conclusions from that study, therefore.

Richard Chandler
31 May 2019