Assignment 9

Instructions (the same as for Assignment 1): Make an R Notebook and in it answer the question below. When you are done, hand in on Quercus the *output* from Previewing (or Knitting) your Notebook. Do *not* hand in the Notebook itself. You want to show that you can (i) write code that will answer the questions, (ii) run that code and get some sensible output, (iii) write some words that show you know what is going on and that reflect your conclusions about the data. Your goal is to convince the grader that you *understand* what you are doing: not only doing the right thing, but making it clear that you know *why* it's the right thing.

Do *not* expect to get help on this assignment. The purpose of the assignments is for you to see how much *you* have understood. You will find that you also learn something from grappling with the assignments. The time to get help is after you watch the lectures and work through the problems from PASIAS, via tutorial and the discussion board. The only reason to contact the instructor while working on the assignments is to report something missing like a data file that cannot be read.

You have 4 hours to complete this assignment after you start it.

My solutions to this assignment, with extra discussion, will be available after everyone has handed in their assignment.

1. A university requires all its incoming students to take a standardized test of their verbal and mathematical abilities. Each student receives a verbal and a math score that is a percentile of all the students that took each test: thus a score of 100 means that the student was the highest of everyone that took the test, and a score of 0 means that the student scored the lowest of everybody on that test. The university claims that it needs to know these test results because they do a good job of predicting each student's (cumulative) GPA at the end of first year. The two test scores and the GPA at the end of first year are recorded for a sample of students, and the results are in http://ritsokiguess.site/STAC32/gpa.csv.

   (a) Read in and display (some of) the data. How many students were sampled?

Solution:

No surprises here:

```
my_url <- "http://ritsokiguess.site/STAC32/gpa.csv"
students <- read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   verbal = col_double(),
##   math = col_double(),
##   gpa = col_double()
## )
```

```
students
```

```
## # A tibble: 40 x 3
##    verbal  math   gpa
##     <dbl> <dbl> <dbl>
##  1     81    87  3.49
```

```
##  2     68    99  2.89
##  3     57    86  2.73
##  4    100    49  1.54
##  5     54    83  2.56
##  6     82    86  3.43
##  7     75    74  3.59
##  8     58    98  2.86
##  9     55    54  1.46
## 10     49    81  2.11
## # ... with 30 more rows
```

There are 40 rows and hence 40 students in the sample.

Just for a change, there is no data-organizing story here. The data came from an old textbook, and I did what any sane person would do: I copied the values into a spreadsheet and saved it as a `.csv`.

(b) Make a suitable plot of the response variable against each of the explanatory variables.

Solution:

This means figuring out that `gpa` had better be the response variable, because that comes at the end of the first year, and the other two test scores are known at the beginning. (The first-year GPA really is, or might be, a "response" to the student's test scores, and thus their presumed mathematical and verbal abilities.)

The best way to do this is in one shot, with facets. This means making the two x-columns longer first. I got lazy this time:

```
students %>%
  pivot_longer(-gpa)
```

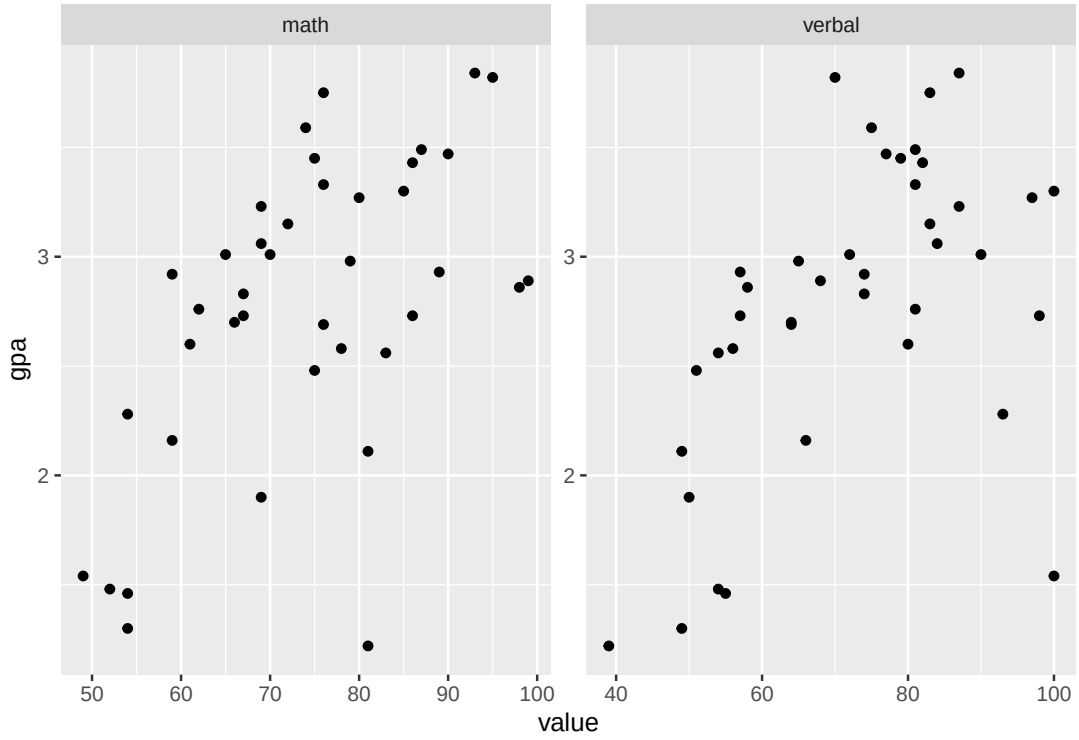
```
## # A tibble: 80 x 3
##      gpa name   value
##    <dbl> <chr>  <dbl>
##  1  3.49 verbal    81
##  2  3.49 math      87
##  3  2.89 verbal    68
##  4  2.89 math      99
##  5  2.73 verbal    57
##  6  2.73 math      86
##  7  1.54 verbal   100
##  8  1.54 math      49
##  9  2.56 verbal    54
## 10  2.56 math      83
## # ... with 70 more rows
```

If you leave out `names_to` and `values_to`, the columns get called `name` and `value`, which actually works perfectly well for our purposes here. Of course, it's also fine to give them more meaningful names.

Then, as we've seen elsewhere, we plot the response variable against whatever we called the values, and then facet by whatever we called the names of these variables. Using `scales="free"` is better because the x-variables might be on different scales, although here they are both

percentiles and so it doesn't matter much. Or, argue that *because* they are both on the same scale, you don't need the `scales="free"` *here* even though you normally would. (This last is, I guess, showing the greatest awareness of what is going on.):

```
students %>%
  pivot_longer(-gpa) %>%
  ggplot(aes(x = value, y = gpa)) + geom_point() +
  facet_wrap(~name, scales = "free")
```



If you cannot make the facets work, get the plots one at a time. It is better to get them both together, but it is better than nothing to get the graphs somehow.

(c) What do you see on your plots? Explain briefly.

Solution:

I see, in both cases, a weak upward trend. I also see an outlier on each plot; on the left one, at the bottom middle with a `math` score around 80 and the lowest `gpa`, and on the right one at the bottom right with a verbal score near 100.

I want you to be clear that these are plots of the *response* against the explanatory variables, and so you are hoping to see *a trend* on these. This is in sharp contrast to residual plots (later) where you are hoping to see *nothing*.

Extra: there is always a grey area between what is part of a weak trend and what is an outlier. My take is that all the other points are part of those weak trends, and only the points I named are clearly off the trends.

Extra extra: the two outliers I named are two different students, because they have different

`gpa` values: the lowest of all, something like 1.3, on the left, and just above 1.5 on the right.

(d) Fit a suitable regression, and display the results.

Solution:

The `gpa` is the response, and the other two variables are explanatory:

```
gpa.1 <- lm(gpa ~ math + verbal, data = students)
summary(gpa.1)
```

```
##
## Call:
## lm(formula = gpa ~ math + verbal, data = students)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10980 -0.20134  0.05034  0.22738  0.74313
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.570537   0.493749  -3.181  0.00297 **
## math         0.033615   0.004928   6.822 4.90e-08 ***
## verbal       0.025732   0.004024   6.395 1.83e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4023 on 37 degrees of freedom
## Multiple R-squared:  0.6811, Adjusted R-squared:  0.6638
## F-statistic: 39.51 on 2 and 37 DF,  p-value: 6.585e-10
```

(e) Does it make practical sense that the Estimates for `math` and `verbal` are positive? Explain briefly.

Solution:

The positive estimates mean that as math score increases (for any verbal score), or as verbal score increases (for any math score), predicted GPA goes up. That is to say, higher test scores are associated with a higher GPA. This makes sense, since you'd expect higher test scores to go with a better ability to learn.

You could be less charitable than this, but be careful: though the relationships on the scatterplots are weak, they are a lot stronger than chance, as evidenced by the small P-values. So you *cannot* say that these Estimates came out positive by chance; they are significantly positive, meaning that if you were to sample another set of students, you would almost certainly see positive estimates again.

(f) Make the usual residual plots for a multiple regression. What do you think is the biggest problem revealed by your plots? Describe briefly how this problem shows up on your graphs.

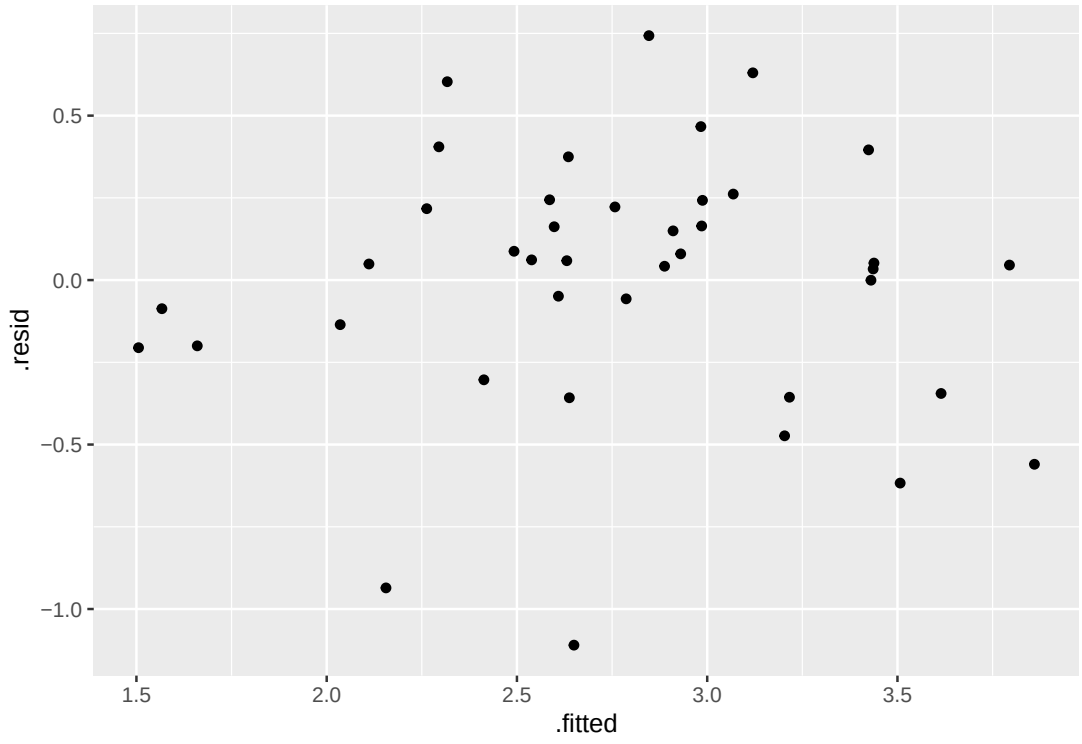
Solution:

The usual plots are three (or four, depending how you count):

- residuals against fitted values
- normal quantile plot of residuals
- residuals against the xs, here `verbal` and `math`.

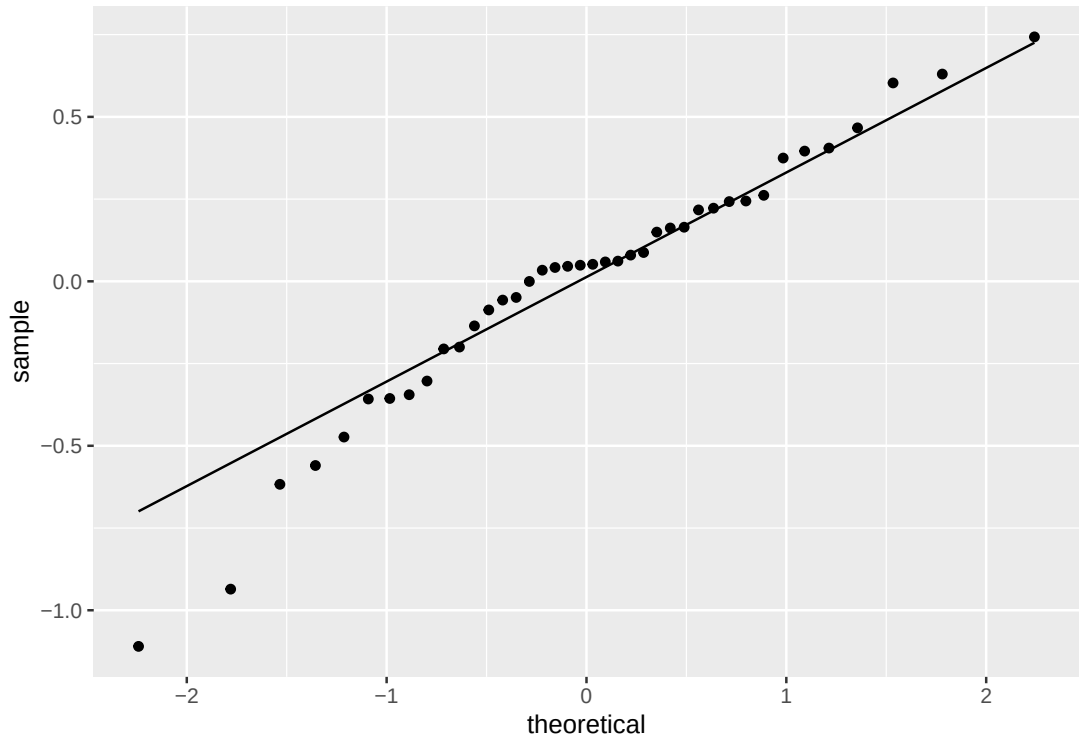
The first two are straightforward:

```
ggplot(gpa.1, aes(x = .fitted, y = .resid)) + geom_point()
```



and

```
ggplot(gpa.1, aes(sample = .resid)) + stat_qq() + stat_qq_line()
```



The plot of residuals against fitted looks pretty random, except for those two points at the bottom, which look *too* low and therefore seem to be outliers. This is supported by the normal quantile plot: all of the residuals look normal enough except for the two most negative ones, which are *too* negative.

With that in mind, you might expect to see those negative residuals on the residuals vs xs as well. Before we get to that, though, we have to make a dataframe with the residuals and the original data in it. This is most easily done using `augment` from the `broom` package; augment the model with the data, rather than the other way around:

```
gpa.1 %>% augment(students)
```

```
## # A tibble: 40 x 9
##    verbal  math   gpa .fitted    .resid .std.resid   .hat .sigma      .cooksd
##     <dbl> <dbl> <dbl>   <dbl>     <dbl>      <dbl>  <dbl>  <dbl>        <dbl>
##  1     81    87  3.49    3.44  0.0517     0.133    0.0613  0.408 0.000384
##  2     68    99  2.89    3.51 -0.617     -1.63     0.118   0.393 0.119
##  3     57    86  2.73    2.79 -0.0571    -0.147    0.0647  0.408 0.000496
##  4    100    49  1.54    2.65 -1.11      -3.04     0.178   0.353 0.670
##  5     54    83  2.56    2.61 -0.0490    -0.126    0.0657  0.408 0.000372
##  6     82    86  3.43    3.43 -0.000376  -0.000965 0.0595  0.408 0.0000000196
##  7     75    74  3.59    2.85  0.743      1.87     0.0258  0.388 0.0310
##  8     58    98  2.86    3.22 -0.356     -0.945    0.122   0.403 0.0415
##  9     55    54  1.46    1.66 -0.200     -0.531    0.123   0.406 0.0132
## 10     49    81  2.11    2.41 -0.303     -0.786    0.0815  0.404 0.0183
## # ... with 30 more rows
```

and then the cleanest way to proceed is to pivot this longer, getting both xs in one column. I

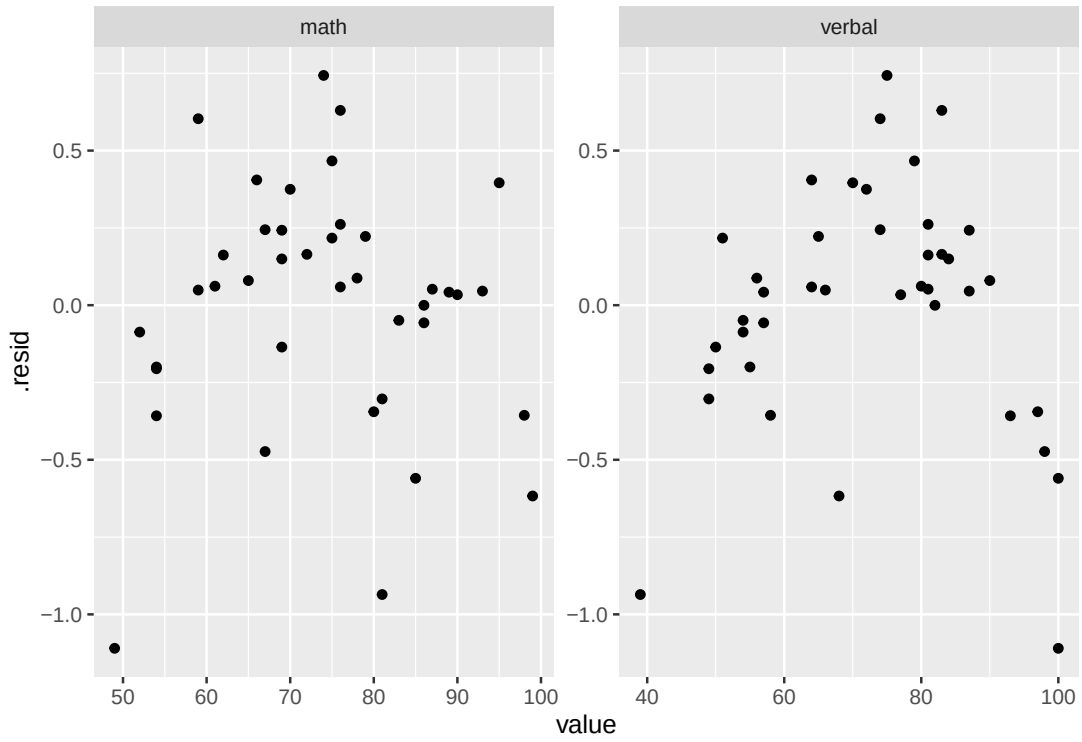
am doing this the lazy way again, getting columns called `name` and `value` by default:

```
gpa.1 %>% augment(students) %>%
  pivot_longer(c(verbal, math))
```

```
## # A tibble: 80 x 9
##      gpa .fitted  .resid .std.resid   .hat .sigma  .cooksd name   value
##    <dbl>   <dbl>   <dbl>      <dbl>  <dbl>  <dbl>    <dbl> <chr>  <dbl>
##  1  3.49    3.44  0.0517      0.133 0.0613  0.408 0.000384 verbal    81
##  2  3.49    3.44  0.0517      0.133 0.0613  0.408 0.000384 math      87
##  3  2.89    3.51 -0.617      -1.63  0.118   0.393 0.119    verbal    68
##  4  2.89    3.51 -0.617      -1.63  0.118   0.393 0.119    math      99
##  5  2.73    2.79 -0.0571     -0.147 0.0647  0.408 0.000496 verbal    57
##  6  2.73    2.79 -0.0571     -0.147 0.0647  0.408 0.000496 math      86
##  7  1.54    2.65 -1.11       -3.04  0.178   0.353 0.670    verbal   100
##  8  1.54    2.65 -1.11       -3.04  0.178   0.353 0.670    math      49
##  9  2.56    2.61 -0.0490     -0.126 0.0657  0.408 0.000372 verbal    54
## 10  2.56    2.61 -0.0490     -0.126 0.0657  0.408 0.000372 math      83
## # ... with 70 more rows
```

Then plot `gpa` against `value` facetted by `name`, as we did for the first plot in this question. Use `scales = "free"` or explain why you don't need to, as before:

```
gpa.1 %>% augment(students) %>%
  pivot_longer(c(verbal, math)) %>%
  ggplot(aes(x = value, y = .resid)) + geom_point() +
  facet_wrap(~name, scales = "free")
```



Once again, this is best, but if you cannot get to this, do the plots one at a time. If you don't

think of `augment`, make a dataframe yourself with all the right stuff in it, eg. like this, where this is the way to pull the residuals out as a column:

```
students %>% mutate(residual = resid(gpa.1))
```

```
## # A tibble: 40 x 4
##    verbal  math   gpa  residual
##     <dbl> <dbl> <dbl>     <dbl>
##  1     81    87  3.49  0.0517
##  2     68    99  2.89 -0.617
##  3     57    86  2.73 -0.0571
##  4    100    49  1.54 -1.11
##  5     54    83  2.56 -0.0490
##  6     82    86  3.43 -0.000376
##  7     75    74  3.59  0.743
##  8     58    98  2.86 -0.356
##  9     55    54  1.46 -0.200
## 10     49    81  2.11 -0.303
## # ... with 30 more rows
```

and then go ahead and make a plot out of this. I haven't done it this way in class, so if you find this idea somewhere, you should say where you got it from.

Anyway, the same two negative residuals show up at the bottom of these plots as well, with otherwise random scatter, supporting the conclusion we had before: those two low outliers are the biggest problem.

(g) Display all the data for the observations with the two most negative residuals.

Solution:

According to all the plots, the two most negative residuals are the only ones below about -0.75, so the strategy is to get the data and the residuals in the same place and `filter`, most easily done by `augment` again:

```
gpa.1 %>% augment(students) %>%
  filter(.resid < -0.75)
```

```
## # A tibble: 2 x 9
##   verbal  math   gpa .fitted .resid .std.resid  .hat .sigma .cooksd
##    <dbl> <dbl> <dbl>   <dbl>  <dbl>      <dbl> <dbl>  <dbl>   <dbl>
## 1    100    49  1.54    2.65 -1.11       -3.04 0.178  0.353   0.670
## 2     39    81  1.22    2.16 -0.936      -2.50 0.136  0.372   0.328
```

This part is a bit of a giveaway that the problem you would find is something to do with residuals!

(h) What does it mean in the context of these data for an observation to have a very negative residual? For the two students with the most negative residuals, what else seems to be unusual about them? Explain briefly.

Solution:

A very negative residual means that the observed value of the response variable is a lot *less*

than the predicted value. Here, that means the student's actual GPA is a lot less than their predicted GPA, based on their test scores. If you did the previous part using `augment`, you have the `.fitted` values right there and you can see this directly: the first student was predicted to get a GPA of 2.65 but only managed 1.54, and the second was predicted to get 2.15 but only achieved 1.22.

(1.54 is not actually the second-lowest GPA; there are, from your first graph, four `gpa` values below 1.5 and one just above, which must be that 1.54.)

What I think is also unusual about these two students is that they have very *unbalanced* test scores: a high one and a very low one (actually the low one is the lowest of all the students[1] in each case).

You might imagine that the student who scored low on `verbal` and pretty high on `math` would study math at university, and maybe found out that university math was a good bit harder than high school math. Or you might imagine that being successful at university requires at least some verbal and mathematical ability both.

This is long enough for you, but I have an Extra that I want to explore:

You might imagine that a student with a high Verbal score might come to university to study something related to language: English, say, or Social Sciences, whereas someone with a high Math score would tend to study something with a lot of math in it: Math itself or Computer Science or Physics, for example. From that point of view, what might determine a student's success is their *higher* score on their two tests,[2] since they will (or might) spend most of their time in courses related to their stronger ability.

The first thing is to make a column with that maximum score in it. This is what you are probably thinking of:

```
students %>% mutate(hi = max(verbal, math))
```

```
## # A tibble: 40 x 4
##    verbal  math   gpa    hi
##     <dbl> <dbl> <dbl> <dbl>
##  1     81    87  3.49   100
##  2     68    99  2.89   100
##  3     57    86  2.73   100
##  4    100    49  1.54   100
##  5     54    83  2.56   100
##  6     82    86  3.43   100
##  7     75    74  3.59   100
##  8     58    98  2.86   100
##  9     55    54  1.46   100
## 10     49    81  2.11   100
## # ... with 30 more rows
```

Oh. Every student got a `hi` of 100, even though only one student got a high of 100 in actual fact.

So what happened? This goes back to how `max` actually works: it is what R calls "vectorized", meaning that it looks at the *whole* column of `verbal` values, and the *whole* column of `math` values, and takes the largest of all of the numbers in both columns. Sometimes this is what you want, but in this case it is not.

This has been an R "gotcha" for a long time, and the approved way around it is to use `pmax`

instead, thus:

```
students %>% mutate(hi = pmax(verbal, math))
```

```
## # A tibble: 40 x 4
##    verbal  math   gpa    hi
##     <dbl> <dbl> <dbl> <dbl>
##  1     81    87  3.49    87
##  2     68    99  2.89    99
##  3     57    86  2.73    86
##  4    100    49  1.54   100
##  5     54    83  2.56    83
##  6     82    86  3.43    86
##  7     75    74  3.59    75
##  8     58    98  2.86    98
##  9     55    54  1.46    55
## 10     49    81  2.11    81
## # ... with 30 more rows
```

If you check down the `hi` column, this is the right thing: the larger of the `verbal` and `math` values *in each row*.

The logic to this is that `max` of two columns is actually a *number*:

```
with(students, max(verbal, math))
```

```
## [1] 100
```

This is 100 because the largest of all the numbers in both columns is 100 (one student scored 100 on `verbal`). The `mutate` hid this rather; it needs an answer that is as long as the number of rows in the dataframe, so it repeated (the R term is "recycled") the value 100 enough times to fill the column. This meant that you got an answer without any messages; however, it was the *wrong* answer!

`pmax`, on the other hand, gives you a vector answer as long as the input vectors (40, because 40 rows, here):

```
with(students, pmax(verbal, math))
```

```
##  [1]  87  99  86 100  83  86  75  98  55  81  76  66  80 100  83  66  83  93  74
## [20]  75  79  81  69  72  54  79  78  98  97  90  54  81  87  95  89  74  93  90
## [39]  81  84
```

and that, which is the right answer as well as the right length, will be used as is. `pmax`, if you are wondering, stands for "parallel max"; it computes the maxima in parallel, that is, by rows.

If you looked at the Extra on the end of the Comp Sci 101 question, you might be wondering whether the `rowwise` idea from there will work here; after all, the idea is to look along each row and find the maximum of `verbal` and `math`, so this ought to work:

```
students %>%
  rowwise() %>%
  mutate(hi = max(verbal, math)) -> students
students
```

```
## # A tibble: 40 x 4
## # Rowwise:
```

```
##    verbal  math   gpa    hi
##     <dbl> <dbl> <dbl> <dbl>
##  1     81    87  3.49    87
##  2     68    99  2.89    99
##  3     57    86  2.73    86
##  4    100    49  1.54   100
##  5     54    83  2.56    83
##  6     82    86  3.43    86
##  7     75    74  3.59    75
##  8     58    98  2.86    98
##  9     55    54  1.46    55
## 10     49    81  2.11    81
## # ... with 30 more rows
```

and indeed it does. I think, once you have seen `rowwise`, that this is the most natural way to go, but it requires a decent understanding of how `max` works. What is happening here is that `rowwise` puts each row in a "box" of its own, so that `max(verbal, math)` now means to consider *only* the `verbal` and `math` in the row we're looking at. That is the same logic as this:

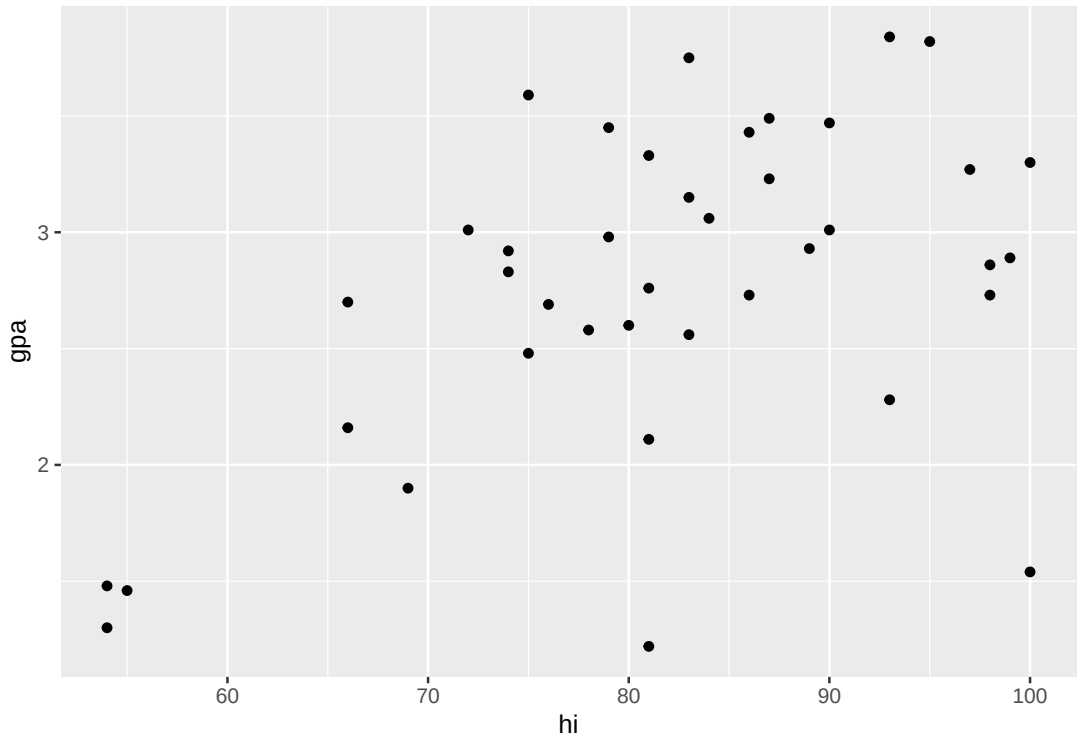
```
students %>%
  mutate(hi2 = map2_dbl(verbal, math, ~max(.x, .y)))
```

```
## # A tibble: 40 x 5
## # Rowwise:
##    verbal  math   gpa    hi   hi2
##     <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     81    87  3.49    87    87
##  2     68    99  2.89    99    99
##  3     57    86  2.73    86    86
##  4    100    49  1.54   100   100
##  5     54    83  2.56    83    83
##  6     82    86  3.43    86    86
##  7     75    74  3.59    75    75
##  8     58    98  2.86    98    98
##  9     55    54  1.46    55    55
## 10     49    81  2.11    81    81
## # ... with 30 more rows
```

where, in English, that reads "for each of the values in `verbal` and `math` (in parallel), work out the max of the two of them, and store it in `hi2`".

All right, does `hi` have any predictive value for `gpa`? A graph first:

```
ggplot(students, aes(x = hi, y = gpa)) + geom_point()
```



Only kinda, and we still have those two outliers bottom right, with `hi` near 80 and at 100, that don't fit the weakish trend. These are the same two outliers as before, and the discussion above would make us suspect that they would fit badly in this model also.

We can try a regression:

```
gpa.2 <- lm(gpa ~ hi, data = students)
summary(gpa.2)
```

```
##
## Call:
## lm(formula = gpa ~ hi, data = students)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.75429 -0.37599  0.00783  0.40025  1.01182
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.429855   0.664330   0.647  0.52149
## hi          0.028644   0.008038   3.564  0.00101 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6085 on 38 degrees of freedom
## Multiple R-squared:  0.2505, Adjusted R-squared:  0.2307
## F-statistic:  12.7 on 1 and 38 DF,  p-value: 0.001006
```

This is a significant upward trend, as before, but the R-squared is not very big. How big was R-squared before?

```
glance(gpa.1)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1     0.681         0.664 0.402      39.5 6.58e-10     2  -18.8  45.5  52.3
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Much bigger. Now, we cannot compare these two models via a test, since neither one of them is contained in the other,[3] but even though the model with only the maximum is simpler, its R-squared is *much* smaller, and so we really seem to need the extra complexity of the larger model: it is "worth having", since the model fits much better. What that actually *means* is that you can predict a student's first-year GPA much more accurately if you know both their verbal and mathematical scores, as compared to just knowing the bigger one.

Having said that, the R-squared of the larger, better model is still only 68%, so that about a third of the variability in GPA is still *not* explained by this model. According to the regression, the unexplained variability is "randomness", but no doubt you can come up with factors that would explain some of it. In your first year, I'm sure you met people who did better or worse than their incoming grades would suggest. Or maybe you were such a person yourself. Either way, I'm sure you wondered about why that was.

Notes

1. In this dataset, I mean; the percentiles were calculated on a larger group of students.

2. Which goes against what I said above about those two outlying students, but let's follow this and see where it goes.

3. The statistical word is "nested", but I didn't want to confuse you with that and the tidyverse "nest", which is a different idea.