STAD29 / STA 1007 assignment 4

Due Tuesday Feb 4 at 11:59pm on Quercus

Hand in the indicated questions. In preparation for the questions you hand in, it is worth your while to work through (or at least read through) the other questions as well.

Hand in your work on Quercus. If you did STAC32 last fall, it's the same procedure. A reminder is here: `https://www.utsc.utoronto.ca/~butler/c32/quercus1.nb.html`

You are reminded that work handed in with your name on it must be *entirely your own work*. It is as if you have signed your name under it. If it was done wholly or partly by someone else, *you have committed an academic offence*, and you can expect to be asked to explain yourself. The same applies if you allow someone else to copy your work. The grader will be watching out for assignments that look suspiciously similar to each other (or to my solutions). Besides which, if you do not do your own assignments, you *will* do badly on the exams, because the struggle to figure things out for yourself is an important part of the learning process.

You will need this first:

```
library(tidyverse)
```

Hand in questions 2 and 4.

1. Work through, or look at the problems in Chapter 20 of PASIAS that relate to nominal responses: 20.4, 20.5, 20.9, 20.11. (The other questions in Chapter 20 are about the organization of some of the data sets used in the chapter; feel free to go through those too.)

2. A survey called High School and Beyond was given to a large number of American high school seniors (grade 12) by the National Center of Education Statistics. The data set at `http://ritsokiguess.site/STAD29/hsb.txt` is a random sample of 200 of those students.

   The variables collected are:

   - `gender`: student's gender, female or male.
   - `ses`: Socio-economic status of student's family (low, middle, or high)
   - `schtyp`: School type, public or private.
   - `prog`: Student's program, general, academic, or vocational.
   - `read`: Score on standardized reading test.
   - `write`: Score on standardized writing test.
   - `math`: Score on standardized math test.
   - `science`: Score on standardized science test.
   - `socst`: Score on standardized social studies test.

   Some of these variables are quantitative and some are categorical.

   You will recognize this as one of the data sets from PASIAS. This time, we take a different angle: we try to predict which program a student went into based on the values of the other variables.

   (a) (2 marks) Read in and display (some of) the data.

Solution: The usual thing:

```
my_url <- "http://ritsokiguess.site/STAD29/hsb.txt"
ml <- read_delim(my_url," ")
## Parsed with column specification:
## cols(
##   `row,` = col_double(),
##   id = col_double(),
##   female = col_character(),
##   ses = col_character(),
##   schtyp = col_character(),
##   prog = col_character(),
##   read = col_double(),
##   write = col_double(),
##   math = col_double(),
##   science = col_double(),
##   socst = col_double(),
##   honors = col_character(),
##   awards = col_double(),
##   cid = col_double()
## )
ml
## # A tibble: 200 x 14
##      `row,`    id female ses   schtyp prog   read write  math science socst honors
##       <dbl> <dbl> <chr>  <chr> <chr>  <chr> <dbl> <dbl> <dbl>   <dbl> <dbl> <chr>
##  1        1    45 female low   public voca~    34    35    41      29    26 not e~
##  2        2   108 male   midd~ public gene~    34    33    41      36    36 not e~
##  3        3    15 male   high  public voca~    39    39    44      26    42 not e~
##  4        4    67 male   low   public voca~    37    37    42      33    32 not e~
##  5        5   153 male   midd~ public voca~    39    31    40      39    51 not e~
##  6        6    51 female high  public gene~    42    36    42      31    39 not e~
##  7        7   164 male   midd~ public voca~    31    36    46      39    46 not e~
##  8        8   133 male   midd~ public voca~    50    31    40      34    31 not e~
##  9        9     2 female midd~ public voca~    39    41    33      42    41 not e~
## 10       10    53 male   midd~ public voca~    34    37    46      39    31 not e~
## # ... with 190 more rows, and 2 more variables: awards <dbl>, cid <dbl>
```

My data frame is called `ml`. Yours probably isn't. It looks as if I have the right thing: 200 rows and 13 columns plus a row number.

(b) (2 marks) Use `multinom` to fit a model predicting program from everything else that makes sense as an explanatory variable. Exclude everything after `socst`. No need to display the output.

Solution: Load package `nnet` first, since that's where `multinom` lives. You might need to install it first, but don't include the installation in what you hand in:

```
library(nnet)
ml.1 <- multinom(prog~female+ses+schtyp+read+write+math+science+socst, data=ml)
## # weights:  33 (20 variable)
## initial  value 219.722458
## iter  10 value 172.925326
## iter  20 value 156.065379
## final  value 155.776076
```

```
## converged
```

I think those are all the variables it makes sense to include. I said to exclude everything after `socst`, and `row` and `id` are identifiers.

Extra: `summary` of one of these models looks like this:

```
summary(ml.1)
```

```
## Call:
## multinom(formula = prog ~ female + ses + schtyp + read + write +
##     math + science + socst, data = ml)
##
## Coefficients:
##          (Intercept) femalemale     seslow sesmiddle schtyppublic        read
## general     3.161817 -0.1400384 0.91165962 0.6737407    0.6021655 -0.04355380
## vocation    7.327585 -0.3526068 0.01878735 1.1740961    1.9403196 -0.03402511
##                write        math    science       socst
## general  -0.02672221 -0.09961354 0.10264775 -0.02550275
## vocation -0.03292013 -0.11559404 0.06001292 -0.07677874
##
## Std. Errors:
##          (Intercept) femalemale    seslow sesmiddle schtyppublic       read
## general     1.766449  0.4485411 0.5852751 0.4990211    0.5549095 0.03049076
## vocation    2.036698  0.4946766 0.6806716 0.5638307    0.8203955 0.03398077
##               write       math    science      socst
## general  0.03295135 0.03423444 0.03132909 0.02646662
## vocation 0.03491891 0.03821745 0.03227780 0.02893924
##
## Residual Deviance: 311.5522
## AIC: 351.5522
```

There is one row for each response category apart from the baseline `academic`, and one column for each explanatory variable (plus the intercept). The numbers are not helpful (yet), but they're log-odds as for a regular logistic regression.

(c) (3 marks) There are a lot of explanatory variables. To see whether any of them can be removed, we can use `step` on one of these models.[1] Run `step` with `direction="backward"` on your model from the previous part, saving the result and then displaying the `summary` of it. Which explanatory variables seem to be still in the model? Hint: `step` produces a lot of output; the input `trace` controls how much output there is. See if you can minimize the amount of output.

Solution: This. If your model in the previous part was different from mine, your answer here might be a little different from mine, but the point here is the procedure. The default value of `trace` is 1 (you can tell by reading the help for `trace`), with a higher value indicating more output, so let's try 0:

```
ml.2 <- step(ml.1, direction="backward", trace=0)
```

```
## trying - female
## trying - ses
## trying - schtyp
## trying - read
## trying - write
## trying - math
## trying - science
## trying - socst
```

```
## # weights:  30 (18 variable)
## initial  value 219.722458
## iter  10 value 172.662548
## iter  20 value 156.063823
## final  value 156.032828
## converged
## trying - ses
## trying - schtyp
## trying - read
## trying - write
## trying - math
## trying - science
## trying - socst
## # weights:  27 (16 variable)
## initial  value 219.722458
## iter  10 value 176.827677
## iter  20 value 156.410686
## final  value 156.406678
## converged
## trying - ses
## trying - schtyp
## trying - read
## trying - math
## trying - science
## trying - socst
## # weights:  24 (14 variable)
## initial  value 219.722458
## iter  10 value 171.169761
## iter  20 value 157.775586
## final  value 157.775540
## converged
## trying - ses
## trying - schtyp
## trying - math
## trying - science
## trying - socst
```

I think that's pretty minimal. The implication here is that it tried to remove `ses`, `schtyp`, `math`, `science` and `socst` and failed to do so, so these are the ones that are left:

```
summary(ml.2)
```

```
## Call:
## multinom(formula = prog ~ ses + schtyp + math + science + socst,
##     data = ml)
##
## Coefficients:
##          (Intercept)      seslow sesmiddle schtyppublic       math    science
## general     2.587029  0.87607389 0.6978995    0.6468812 -0.1212242 0.08209791
## vocation    6.687272 -0.01569301 1.2065000    1.9955504 -0.1369641 0.03941237
##                socst
## general  -0.04441228
## vocation -0.09363417
##
```

```
## Std. Errors:
##          (Intercept)    seslow sesmiddle schtyppublic       math    science
## general     1.686492 0.5758781 0.4930330     0.545598 0.03213345 0.02787694
## vocation    1.945363 0.6690861 0.5571202     0.812881 0.03591701 0.02864929
##               socst
## general  0.02344856
## vocation 0.02586717
##
## Residual Deviance: 315.5511
## AIC: 343.5511
```

This is indeed the case. Careful that `ses` is the name of the categorical variable, and the values in `seslow` and `sesmiddle` are the estimates for two of its *levels* (the other one `high` being the baseline). The `call` line at the top of the output also shows the model that was fitted, which has the same five things in it.

(d) (3 marks) The `summary` output is very hard to understand. Let's set up to do some predictions. First, for each of the explanatory variables in your final model from `step`, find out all its levels if it is categorical, and find its (first and third) quartiles if it is quantitative. Do this how you like. You don't need to be clever.

Solution: The lazy way to do this is to look at the `summary` of the entire data frame:

```
summary(ml)
##       row,              id           female              ses
##  Min.   :  1.00   Min.   :  1.00   Length:200         Length:200
##  1st Qu.: 50.75   1st Qu.: 50.75   Class :character   Class :character
##  Median :100.50   Median :100.50   Mode  :character   Mode  :character
##  Mean   :100.50   Mean   :100.50
##  3rd Qu.:150.25   3rd Qu.:150.25
##  Max.   :200.00   Max.   :200.00
##     schtyp              prog                read           write
##  Length:200         Length:200         Min.   :28.00   Min.   :31.00
##  Class :character   Class :character   1st Qu.:44.00   1st Qu.:45.75
##  Mode  :character   Mode  :character   Median :50.00   Median :54.00
##                                        Mean   :52.23   Mean   :52.77
##                                        3rd Qu.:60.00   3rd Qu.:60.00
##                                        Max.   :76.00   Max.   :67.00
##       math          science          socst          honors
##  Min.   :33.00   Min.   :26.00   Min.   :26.00   Length:200
##  1st Qu.:45.00   1st Qu.:44.00   1st Qu.:46.00   Class :character
##  Median :52.00   Median :53.00   Median :52.00   Mode  :character
##  Mean   :52.65   Mean   :51.85   Mean   :52.41
##  3rd Qu.:59.00   3rd Qu.:58.00   3rd Qu.:61.00
##  Max.   :75.00   Max.   :74.00   Max.   :71.00
##      awards          cid
##  Min.   :0.00   Min.   : 1.00
##  1st Qu.:0.00   1st Qu.: 5.00
##  Median :1.00   Median :10.50
##  Mean   :1.67   Mean   :10.43
##  3rd Qu.:2.00   3rd Qu.:15.00
##  Max.   :7.00   Max.   :20.00
```

This will give you the quartiles for all the quantitative variables. To deal with the categorical variables `ses` and `schtyp`, you can `count` up how many times each one appears:

```
ml %>% count(ses)
```

```
## # A tibble: 3 x 2
##   ses        n
##   <chr>  <int>
## 1 high      58
## 2 low       47
## 3 middle    95
```

```
ml %>% count(schtyp)
```

```
## # A tibble: 2 x 2
##   schtyp      n
##   <chr>   <int>
## 1 private    32
## 2 public    168
```

Or get the `distinct` ones:

```
ml %>% distinct(ses)
```

```
## # A tibble: 3 x 1
##   ses
##   <chr>
## 1 low
## 2 middle
## 3 high
```

```
ml %>% distinct(schtyp)
```

```
## # A tibble: 2 x 1
##   schtyp
##   <chr>
## 1 public
## 2 private
```

Or turn it into a `factor` (it's currently text) and get its `levels`:

```
levels(factor(ml$ses))
```

```
## [1] "high"   "low"    "middle"
```

```
levels(factor(ml$schtyp))
```

```
## [1] "private" "public"
```

Going back to the quantitative variables, to get the Q1 and Q3 of everything you want, perhaps the most obvious way is to construct a great big `summarize`:

```
ml %>% summarize(
           mathq1=quantile(math, 0.25),
           mathq3=quantile(math, 0.75),
           scienceq1=quantile(science, 0.25),
           scienceq3=quantile(science, 0.75),
           socstq1=quantile(socst, 0.25),
           socstq3=quantile(socst, 0.75)
       )
```

```
## # A tibble: 1 x 6
##   mathq1 mathq3 scienceq1 scienceq3 socstq1 socstq3
##    <dbl>  <dbl>     <dbl>     <dbl>   <dbl>   <dbl>
## 1     45     59        44        58      46      61
```

Or, you can use `summarize` to get Q1 and Q3 of everything quantitative, and then pick out what you want. With more than one summary statistic, you have to put them in a `list`. These variants of `summarize` use the squiggle and dot thing like `map` does: "for each variable that is numeric, work out the first and third quartiles of it":

```
ml %>% summarize_if(is.numeric, list(q1=~quantile(., 0.25), q3=~quantile(., 0.75)))
## # A tibble: 1 x 18
##   `row,_q1` id_q1 read_q1 write_q1 math_q1 science_q1 socst_q1 awards_q1 cid_q1
##       <dbl> <dbl>   <dbl>    <dbl>   <dbl>      <dbl>    <dbl>     <dbl>  <dbl>
## 1      50.8  50.8      44     45.8      45         44       46         0      5
## # ... with 9 more variables: `row,_q3` <dbl>, id_q3 <dbl>, read_q3 <dbl>,
## #   write_q3 <dbl>, math_q3 <dbl>, science_q3 <dbl>, socst_q3 <dbl>,
## #   awards_q3 <dbl>, cid_q3 <dbl>
```

There are more columns than will display in my output, but you'll be able to scroll across to get the ones you want.

Or `select` the columns first, and then summarize all of them:

```
ml %>% select(math, science, socst) %>%
    summarize_all(list(q1=~quantile(., 0.25), q3=~quantile(., 0.75)))
## # A tibble: 1 x 6
##   math_q1 science_q1 socst_q1 math_q3 science_q3 socst_q3
##     <dbl>      <dbl>    <dbl>   <dbl>      <dbl>    <dbl>
## 1      45         44       46      59         58       61
```

Find a way that works. `ses` has levels `low, middle, high`; `schtyp` has levels `private` and `public`; the quartiles of `math` are 45 and 59; of `science` are 44 and 58, and of `socst` are 46 and 61.

(e) (3 marks) Make a data frame that includes all combinations of values you obtained in the previous part. (I have five variables, two of which are categorical, and one of those has three levels, so I have 48 rows.)

Solution: `crossing`, as ever for these. I like to set up the pairs or trios of values for each variable first, using plural names. But if you can do it all in one go without getting confused, go for it:

```
sess <- c("low", "middle", "high")
schtyps  <- c("private", "public")
sciences <- c(44, 58)
socsts <- c(46, 61)
maths <- c(45, 59)
new <- crossing(ses=sess, schtyp=schtyps, science=sciences, socst=socsts, math=maths)
new
## # A tibble: 48 x 5
##    ses   schtyp  science socst  math
##    <chr> <chr>     <dbl> <dbl> <dbl>
##  1 high  private      44    46    45
##  2 high  private      44    46    59
##  3 high  private      44    61    45
##  4 high  private      44    61    59
##  5 high  private      58    46    45
##  6 high  private      58    46    59
##  7 high  private      58    61    45
##  8 high  private      58    61    59
##  9 high  public       44    46    45
## 10 high  public       44    46    59
```

```
## # ... with 38 more rows
```

48 rows.

(f) (3 marks) Obtain predictions, using your model that came from `step` and the data frame of values to predict for that you just created. Be sure to obtain predicted *probabilities* for each response category. Display (some of) your predictions side by side with the values they are predictions for. Save your final data frame.

Solution: Now that we've done the setup, that is this:

```
p <- predict(ml.2, new, type="probs")
preds <- cbind(new, p)
preds
##       ses  schtyp science socst math  academic    general    vocation
## 1    high private      44    46   45 0.7134056 0.19467533 0.091919090
## 2    high private      44    46   59 0.9355143 0.04676994 0.017715808
## 3    high private      44    61   45 0.8533882 0.11961922 0.026992544
## 4    high private      44    61   59 0.9705639 0.02492415 0.004511933
## 5    high private      58    46   45 0.4796188 0.41308175 0.107299437
## 6    high private      58    46   59 0.8398621 0.13252254 0.027615351
## 7    high private      58    61   45 0.6678579 0.29546346 0.036678643
## 8    high private      58    61   59 0.9181696 0.07441913 0.007411284
## 9    high  public      44    46   45 0.4050374 0.21106025 0.383902398
## 10   high  public      44    46   59 0.8098660 0.07731549 0.112818486
## 11   high  public      44    61   45 0.6665147 0.17840254 0.155082736
## 12   high  public      44    61   59 0.9231602 0.04526998 0.031569821
## 13   high  public      58    46   45 0.2330791 0.38333639 0.383584523
## 14   high  public      58    46   59 0.6480070 0.19525333 0.156739636
## 15   high  public      58    61   45 0.4446798 0.37566783 0.179652424
## 16   high  public      58    61   59 0.8236198 0.12747511 0.048905067
## 17    low private      44    46   45 0.5611194 0.36770859 0.071171992
## 18    low private      44    46   59 0.8781945 0.10543411 0.016371402
## 19    low private      44    61   45 0.7311286 0.24610598 0.022765414
## 20    low private      44    61   59 0.9378700 0.05783796 0.004292060
## 21    low private      58    46   45 0.3040867 0.62894290 0.066970449
## 22    low private      58    46   59 0.7085685 0.26849592 0.022935544
## 23    low private      58    61   45 0.4724829 0.50197244 0.025544648
## 24    low private      58    61   59 0.8315402 0.16185233 0.006607518
## 25    low  public      44    46   45 0.3140279 0.39296476 0.293007372
## 26    low  public      44    46   59 0.7318525 0.16778415 0.100363373
## 27    low  public      44    61   45 0.5342341 0.34339734 0.122368578
## 28    low  public      44    61   59 0.8684870 0.10227527 0.029237686
## 29    low  public      58    46   45 0.1522144 0.60118276 0.246602854
## 30    low  public      58    46   59 0.5097610 0.36885813 0.121380889
## 31    low  public      58    61   45 0.2918452 0.59208398 0.116070791
## 32    low  public      58    61   59 0.6992339 0.25989333 0.040872777
## 33 middle private      44    46   45 0.5053223 0.27710018 0.217577504
## 34 middle private      44    46   59 0.8592933 0.08632801 0.054378635
## 35 middle private      44    61   45 0.7207859 0.20302733 0.076186800
## 36 middle private      44    61   59 0.9370842 0.04835806 0.014557722
## 37 middle private      58    46   45 0.2874917 0.49757548 0.214932851
## 38 middle private      58    46   59 0.7007880 0.22220936 0.077002627
```

```
## 39 middle private      58    61    45 0.4824966 0.42895116 0.088552275
## 40 middle private      58    61    59 0.8404424 0.13688742 0.022670171
## 41 middle  public      44    46    45 0.1917717 0.20081201 0.607416331
## 42 middle  public      44    46    59 0.6033659 0.11575188 0.280882177
## 43 middle  public      44    61    45 0.4318847 0.23230203 0.335813256
## 44 middle  public      44    61    59 0.8245222 0.08125115 0.094226640
## 45 middle  public      58    46    45 0.1019926 0.33708496 0.560922421
## 46 middle  public      58    46    59 0.4142840 0.25084807 0.334867934
## 47 middle  public      58    61    45 0.2470510 0.41940845 0.333540553
## 48 middle  public      58    61    59 0.6624924 0.20605039 0.131457174
```

Mine displays all 48 rows. Yours will probably display the first 10, with a Next button to click on to see more.

I saved my final output to use again in a moment. (That's why I asked you to save yours.)

(g) (3 marks) Assess the effect of `ses` on the probabilities of a student being in each of the three programs by displaying appropriate rows of your saved data frame of predictions (using `slice` or `filter`). Which programs do students of each socioeconomic status mostly end up in?

Solution: Find three rows of your data frame of predictions that have (i) the *same* values for `schtyp, science, socst, math` and (ii) *different* values for `ses`. This is assessing what happens when you change `ses` but leave everything else the same.

I looked through my data frame and saw that rows 1, 17, 33 would do this:

```
preds %>% slice(1, 17, 33)
```

```
##      ses  schtyp science socst math  academic   general   vocation
## 1   high private      44    46   45 0.7134056 0.1946753 0.09191909
## 2    low private      44    46   45 0.5611194 0.3677086 0.07117199
## 3 middle private      44    46   45 0.5053223 0.2771002 0.21757750
```

The precise rows depend on the order you used in the `crossing` earlier. Or you could use `filter` to pick out the rows where you want the values to stay the same (`schtyp, science, socst, math`). These are actually different probabilities (just fine) but the comparison is the same:

```
preds %>% filter(
              schtyp=="public",
              science==44,
              socst==46,
              math==45
          )
```

```
##      ses schtyp science socst math  academic   general  vocation
## 1   high public      44    46   45 0.4050374 0.2110602 0.3839024
## 2    low public      44    46   45 0.3140279 0.3929648 0.2930074
## 3 middle public      44    46   45 0.1917717 0.2008120 0.6074163
```

Use either value for each of those; it doesn't matter.

So, what are those predictions telling us? High-SES students are most likely to go into the academic program, while middle-SES students are most likely to go into the vocational program.

What you say about low-SES students will depend on precisely which values you pick for the other variables, but those seem most likely to end up in the `general` program, at least for my values.

Comment, if you want to add any: the high-SES students may mainly have in mind to go to university later, so it would make sense that most of them would go into the academic program.

Maybe most of the middle-SES students reckon they're *not* going to university, so they would tend to go into a program that would more immediately help them get a job.

(h) (3 marks) By looking at an appropriate choice of rows from your data frame of predictions, assess the effect of an increase in `math` score on a student's choice of program.

Solution: Now pick two rows from your data frame of predictions that differ on the value of `math` but are the same on everything else. In mine, rows 1 and 2 will do that:

```
preds %>% slice(1,2)
```

```
##    ses  schtyp science socst math  academic    general   vocation
## 1 high private      44    46   45 0.7134056 0.19467533 0.09191909
## 2 high private      44    46   59 0.9355143 0.04676994 0.01771581
```

Or, use `filter` to grab *any* rows of `math`, but pick a value of the others to be consistent with, for example:

```
preds %>% filter(
          ses=="middle",
          schtyp=="public",
          science==58,
          socst==46
          )
```

```
##      ses schtyp science socst math  academic   general  vocation
## 1 middle public      58    46   45 0.1019926 0.3370850 0.5609224
## 2 middle public      58    46   59 0.4142840 0.2508481 0.3348679
```

This gives a different pair of rows, but the picture should be the same whichever ones you pick.[2]

The first time, I gave `socst` a value that I didn't predict for at all, and wondered why the result had no rows!

The effect, then: whichever values you look at, a student with a higher math score is *much* more likely to go into the academic program, and much *less* likely to go into either of the others.

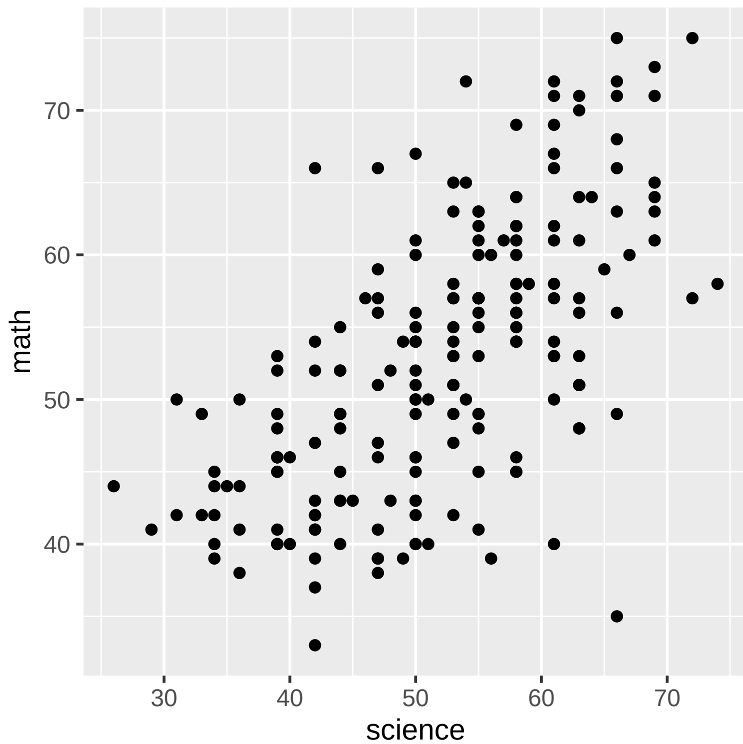
Extra: the effect seems to be biggest for math. I tried the others too. Science:

```
preds %>% slice(1,5)
```

```
##    ses  schtyp science socst math  academic   general   vocation
## 1 high private      44    46   45 0.7134056 0.1946753 0.09191909
## 2 high private      58    46   45 0.4796188 0.4130818 0.10729944
```

This effect looks backwards to me, but remember it's the effect of a change in science *all else remaining equal*, which may not be the case; a student with a high science score may tend to have a high math score as well:

```
ggplot(ml, aes(x=science, y=math)) + geom_point()
```



At least kinda. So if you know about math, you kinda know about science too.

The effect of social studies is like the one for math:

```
preds %>% slice(1, 3)
```

```
##    ses  schtyp science socst math  academic   general   vocation
## 1 high private      44    46   45 0.7134056 0.1946753 0.09191909
## 2 high private      44    61   45 0.8533882 0.1196192 0.02699254
```

though evidently a student who scores high on social studies will tend to take *different* academic courses than one who scores high on math.

3. Work through, or at least read through, chapter 21 of PASIAS.

4. A small clinical trial is run to compare two combination treatments in patients with advanced gastric cancer. Twenty participants with stage IV gastric cancer who consent to participate in the trial are randomly assigned to receive chemotherapy before surgery or chemotherapy after surgery. The primary outcome is death and participants are followed for up to 48 months (4 years) following enrollment into the trial. (The foregoing is directly taken from the website where I got the data; it's not how I would have written it, but I think it's good for you to see how this kind of thing is written by others.)

   The data are shown in `http://ritsokiguess.site/STAD29/chemo.csv`. There are three columns: whether each patient had chemotherapy before or after surgery, how many months they were observed for, and whether or not they were observed to have died.

   (a) (2 marks) Read in and display (some of) the data.

Solution: Nothing new here:

```
my_url <- "http://ritsokiguess.site/STAD29/chemo.csv"
trial <- read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   chemo = col_character(),
##   months = col_double(),
##   died = col_character()
## )
```

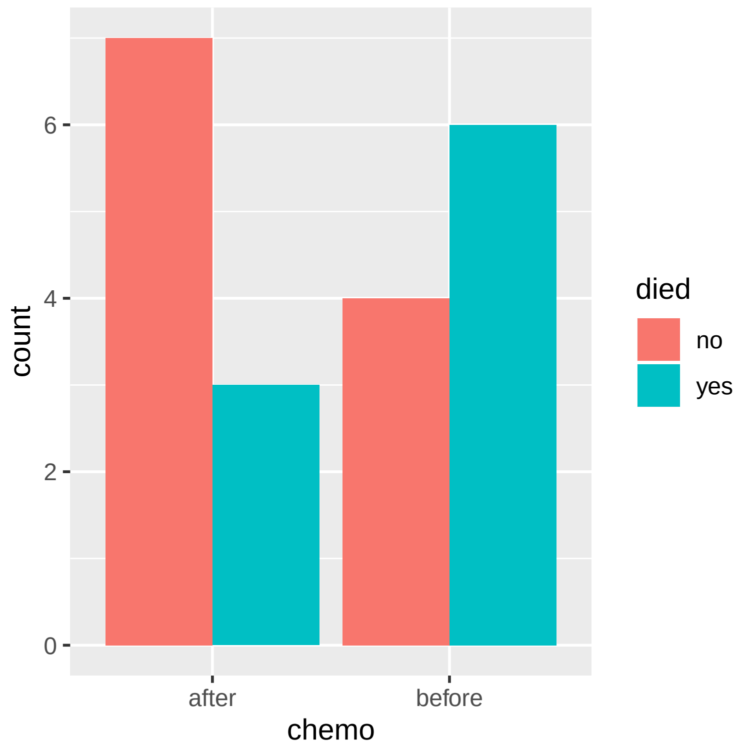
```
trial
```

```
## # A tibble: 20 x 3
##    chemo  months died
##    <chr>   <dbl> <chr>
##  1 before      8 yes
##  2 before     12 yes
##  3 before     26 yes
##  4 before     14 yes
##  5 before     21 yes
##  6 before     27 yes
##  7 before      8 no
##  8 before     32 no
##  9 before     20 no
## 10 before     40 no
## 11 after      33 yes
## 12 after      28 yes
## 13 after      41 yes
## 14 after      48 no
## 15 after      48 no
## 16 after      25 no
## 17 after      37 no
## 18 after      48 no
## 19 after      25 no
## 20 after      43 no
```

I got all 20 rows, but you will probably see only the first ten, and you can scroll down to see that there were also patients who had chemotherapy after surgery.

Extra: if you ignore the `months` column, you could make a graph of `chemo` and `died`; these are both categorical so a grouped bar chart would be the thing:

```
ggplot(trial, aes(x=chemo, fill=died)) + geom_bar(position="dodge")
```



The majority of the patients who had chemo before surgery died, and the majority of the patients who had chemo after surgery were still alive when last observed. This suggests that having chemo after surgery is better, but it fails to consider how long the patients were observed for. We *will* consider this in the modelling we do shortly, but for now consider the possibility that the patients who had chemo after surgery might not have been observed for very long, and that would make this treatment look better than it actually was.

(b) (2 marks) In the context of *this* data set, what would a censored observation look like? Give an example of one from the data set.

Solution: A censored observation is one where the event of interest (what the description calls the "primary endpoint"), that is, death, was never observed to occur. For example, patient 7 was only observed for 8 weeks and was never observed to die. (Or patients 8, 9 or 10.)

Extra: patients can be censored for a variety of reasons: maybe they decide to withdraw from the study, or maybe they stopped coming to doctor's appointments (I think, though, this study was in a hospital). Or they moved house or hospital. Or, maybe they *died of something else* and not of the gastric cancer directly.

(c) (3 marks) Create a response variable suitable for a Cox proportional-hazards regression. Do this outside the data frame, and display at least some of its values. How are the censored observations distinguished?

Solution: At some point you'll need this:

```
library(survival)
```

and then (I used my customary name `y` for a response):

```
y <- with(trial, Surv(months, died=="yes"))
```

```
y
## [1]  8  12  26  14  21  27   8+ 32+ 20+ 40+ 33  28  41  48+ 48+ 25+ 37+ 48+ 25+
## [20] 43+
```

The censored observations are displayed with a plus.

Extra: though `y` looks like an ordinary vector and you would expect to be able to create it with `mutate`, it actually has *two* columns, and it just displays by default with special formatting. This is how it really looks:

```
print.default(y)
##       time status
##  [1,]    8      1
##  [2,]   12      1
##  [3,]   26      1
##  [4,]   14      1
##  [5,]   21      1
##  [6,]   27      1
##  [7,]    8      0
##  [8,]   32      0
##  [9,]   20      0
## [10,]   40      0
## [11,]   33      1
## [12,]   28      1
## [13,]   41      1
## [14,]   48      0
## [15,]   48      0
## [16,]   25      0
## [17,]   37      0
## [18,]   48      0
## [19,]   25      0
## [20,]   43      0
## attr(,"type")
## [1] "right"
## attr(,"class")
## [1] "Surv"
```

The censoring information is obtained from the `status` column in there.

(d) (3 marks) Fit a Cox proportional-hazards model to predict survival time from the treatment `chemo`. Display the results.

Solution: This is very like a regression, only the computation behind the scenes properly takes care of the censoring:

```
trial.1 <- coxph(y~chemo, data=trial)
summary(trial.1)
## Call:
## coxph(formula = y ~ chemo, data = trial)
##
##   n= 20, number of events= 9
##
##               coef exp(coef) se(coef)     z Pr(>|z|)
## chemobefore 1.9673    7.1515   0.8474 2.321   0.0203 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##             exp(coef) exp(-coef) lower .95 upper .95
## chemobefore     7.151     0.1398     1.358     37.65
##
## Concordance= 0.748  (se = 0.047 )
## Likelihood ratio test= 6.46  on 1 df,   p=0.01
## Wald test            = 5.39  on 1 df,   p=0.02
## Score (logrank) test = 6.92  on 1 df,   p=0.009
```

(e) (2 marks) Is there any evidence that one of the treatments is better than the other? If so, which treatment is better? Explain briefly. (Hint: you will get this backwards unless you are careful.)

Solution: The P-value of 0.0203 on the `chemobefore` line says that there is a significant difference in survival between having chemo before surgery and having it after (the baseline).

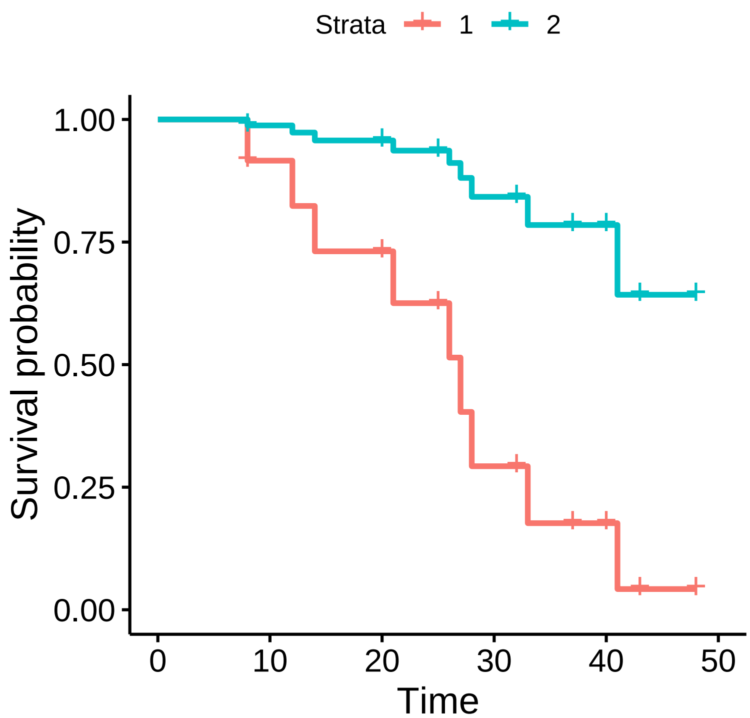
To see which way around it goes, look at the estimate (coefficient) 1.9673 for `chemobefore`. This is positive, and larger than the 0 for the baseline `chemoafter`. A more positive coefficient goes with a *higher hazard of event*: that is, the event (death) is more likely to happen sooner for `before` than for `after`. Thus, having chemotherapy *after* surgery is better for survival.

Extra: this is easy to get backwards, but remember that a higher hazard of death is *bad*. (Usually the event of interest is something bad like death, but not always; I had an example once where the event of interest was passing a driving test, and in that case a higher hazard of event was a good thing in that people were more likely to pass the driving test sooner.)

(f) (3 marks) Make a suitable plot that shows which treatment is better, and explain briefly why it shows that.

Solution: This is `ggsurvplot`. That goes like this:

```r
library(survminer)
## Loading required package: ggpubr
## Loading required package: magrittr
##
## Attaching package: 'magrittr'
## The following object is masked from 'package:purrr':
##
##     set_names
## The following object is masked from 'package:tidyr':
##
##     extract
trial.new <- tribble(
    ~chemo,
    "before",
    "after"
)
s1 <- do.call(survfit, list(formula=trial.1, newdata=trial.new, data=trial))
ggsurvplot(s1, conf.int = F)
```



The first step is to create a `new` data frame with values to predict for (`survfit` is like `predict`). Our one explanatory variable is `chemo` and its values are `before` and `after` (either order is good). Then that rather arcane thing with `do.call` in it (copy the example from lecture notes and modify), and finally the plot. The strata in the plot are in the same order as in `trial.new`.

By way of comment: the best survival goes with the upper right survival curve, because that is associated with surviving longest. In mine, that is stratum 2, which is `after`. (If you had `after` first in your data frame, it'll be your stratum 1, and your red and blue survival curves will be switched around from mine.)

In short, if the event is undesirable, the upper right survival curve goes with avoiding it for as long as possible, and is thus best.

Notes

[1]Oddly, `step` works but `drop1` doesn't. I have yet to figure out why that is.

[2]The principle with `filter` is that you omit the explanatory variable whose effect you want to see, and supply values, any ones, for the others.