STAD29 / STA 1007 assignment 3

Due Tuesday Jan 28 at 11:59pm on Blackboard

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.

Some packages that you will probably need too:

```
library(MASS)
library(tidyverse)
library(nnet)
```

I am also using `conflicted` which requires (for this assignment) this code:

```
library(conflicted)
conflict_prefer("select", "dplyr")
```

```
## [conflicted] Will prefer dplyr::select over any other package
```

```
conflict_prefer("filter", "dplyr")
```

```
## [conflicted] Will prefer dplyr::filter over any other package
```

I used `conflicted` as well because both `tidyverse` and `MASS` have a `select` and I want to make sure I get the right one. There is also a `filter` in base R that I make sure I don't get by mistake.

1. Review the questions in Chapter 19 of PASIAS that relate to multiple logistic regression (where you have several explanatory variables but a yes/no response): 19.4, 19.5, 19.6, 19.8, 19.10.

   Hand the next one in.

2. A study was made on the effects of analgesics (painkillers) on neuralgia patients. Two different treatments were used, labelled A and B, along with a placebo, labelled P. The researchers also recorded the age and sex of each patient, and the duration of pain before the treatment began. The response variable was whether or not the patient reported pain after the treatment.

   (a) (2 marks) What feature of this study makes logistic regression a plausible method of analysis?

   > **Solution:** The response variable is a yes/no: pain or no pain. This is what makes a logistic regression a plausible method of analysis. A rather rapid two points.

   (b) (2 marks) The data are in `http://ritsokiguess.site/STAD29/neuralgia.txt`. Read the data into R, and check that you have all the promised variables. (Hint: check the layout of the data.)

Solution: When you looked at the data in your browser, which of course you did, you'll have seen that the values are aligned in columns, and thus `read_table` is called for:

```
my_url="http://ritsokiguess.site/STAD29/neuralgia.txt"
neur <- read_table(my_url)
```

```
## Parsed with column specification:
## cols(
##   treatment = col_character(),
##   sex = col_character(),
##   age = col_double(),
##   duration = col_double(),
##   pain = col_character()
## )
```

```
neur
```

```
## # A tibble: 60 x 5
##    treatment sex     age duration pain
##    <chr>     <chr> <dbl>    <dbl> <chr>
##  1 P         F        68        1 No
##  2 P         M        66       26 Yes
##  3 A         F        71       12 No
##  4 A         M        71       17 Yes
##  5 B         F        66       12 No
##  6 A         F        64       17 No
##  7 P         M        70        1 Yes
##  8 A         F        64       30 No
##  9 B         F        78        1 No
## 10 B         M        75       30 Yes
## # ... with 50 more rows
```

Treatment, sex, age, duration, and a yes/no for pain, and all numbers or text where you would expect them. Check.

Use whatever name you like for the data frame, as usual.

(c) (2 marks) Fit a logistic regression predicting `pain` from everything else. Display the output. Hint: what kind of thing does your response variable need to be?

Solution:

`pain` is text, and needs to be an explicit `factor`. Make it one, by creating a new column, or as here:

```
neur.1=glm(factor(pain)~treatment+sex+age+duration, data=neur, family=binomial)
summary(neur.1)
```

```
##
## Call:
## glm(formula = factor(pain) ~ treatment + sex + age + duration,
##     family = binomial, data = neur)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7638  -0.5904  -0.1952   0.6151   2.3153
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -20.588282   7.102883  -2.899  0.00375 **
## treatmentB   -0.526853   0.937025  -0.562  0.57394
## treatmentP    3.181690   1.016021   3.132  0.00174 **
## sexM          1.832202   0.796206   2.301  0.02138 *
## age           0.262093   0.097012   2.702  0.00690 **
## duration     -0.005859   0.032992  -0.178  0.85905
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 81.503  on 59  degrees of freedom
## Residual deviance: 48.736  on 54  degrees of freedom
## AIC: 60.736
##
## Number of Fisher Scoring iterations: 5
```

(d) (3 marks) When you have a mixture of quantitative and categorical explanatory variables, it's hard to tell from the `summary` output which ones should be kept. Pass your model into `drop1` with `test="Chisq"`. Which explanatory variable(s) are candidates to be removed? Explain briefly.

Solution: This:

```
drop1(neur.1, test="Chisq")
## Single term deletions
##
## Model:
## factor(pain) ~ treatment + sex + age + duration
##           Df Deviance    AIC     LRT  Pr(>Chi)
## <none>         48.736 60.736
## treatment  2   68.424 76.424 19.6882 5.306e-05 ***
## sex        1   55.036 65.036  6.3000  0.012074 *
## age        1   59.213 69.213 10.4769  0.001209 **
## duration   1   48.767 58.767  0.0317  0.858663
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The P-values in the last column are for each explanatory variable (as a whole, if it's categorical). They are all significant except for `duration`, with a (very) large P-value of 0.858, and so `duration` is the only thing that could be removed.

Extra: the small P-value for `treatment` says that there is *some* effect of treatment, but not what it is.[1] We'll come back to this shortly.

(e) (2 marks) Remove any non-significant explanatory variables and fit again, and display the results.

Solution: Copy, paste and edit is your friend:

```
neur.2=glm(factor(pain)~treatment+sex+age, data=neur, family=binomial)
summary(neur.2)
##
## Call:
## glm(formula = factor(pain) ~ treatment + sex + age, family = binomial,
##     data = neur)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7436  -0.5904  -0.1982   0.6020   2.3064
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.86939    6.94218  -3.006  0.00265 **
## treatmentB   -0.54741    0.93123  -0.588  0.55664
## treatmentP    3.17896    1.01348   3.137  0.00171 **
## sexM          1.82353    0.79195   2.303  0.02130 *
## age           0.26496    0.09591   2.763  0.00573 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 81.503  on 59  degrees of freedom
## Residual deviance: 48.767  on 55  degrees of freedom
## AIC: 58.767
##
## Number of Fisher Scoring iterations: 5
```

Extra: a way to avoid the copy-paste is via `update`, which looks like this:

```
neur.2=update(neur.1,.~.-duration)
summary(neur.2)
```

```
##
## Call:
## glm(formula = factor(pain) ~ treatment + sex + age, family = binomial,
##     data = neur)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7436  -0.5904  -0.1982   0.6020   2.3064
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.86939    6.94218  -3.006  0.00265 **
## treatmentB   -0.54741    0.93123  -0.588  0.55664
## treatmentP    3.17896    1.01348   3.137  0.00171 **
## sexM          1.82353    0.79195   2.303  0.02130 *
## age           0.26496    0.09591   2.763  0.00573 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 81.503  on 59  degrees of freedom
## Residual deviance: 48.767  on 55  degrees of freedom
## AIC: 58.767
##
## Number of Fisher Scoring iterations: 5
```

Same results. The syntax of `update` is this: the first thing is a model you already fitted (here `neur.1`), and the second thing is a kind of a model formula with a lot of dots in it. A dot means "whatever was there before": that is, the same response variable (the dot to the left of the squiggle) and all the same explanatory variables (the one to the right) *except* take away (the minus) the variable `duration`. You can add variables by putting a plus in front of them. `update` works with any regression-like thing where you have a response variable depending in some way upon explanatory variables, such as regular regression, logistic regression, and even (I think) survival analysis, which we see later.

(f) (1 mark) What is the thing that we are predicting the probability of? Explain (very) briefly.

Solution: `pain` is a categorical variable with two levels `Yes` and `No`, which are taken in alphabetical order. So `No` is the baseline and we are predicting the probability of `Yes`: that is, that the patient *does* report pain after the treatment.

This is the usual thing, but I wanted to make sure you got it right here, so that you have the right interpretations below.

(g) (2 marks) By looking at the numbers in the Estimate column of the output from your most recent model, describe how the treatments differ, and thus which one(s) are best or worst.

Solution: Treatment A is the baseline, so its Estimate is zero. Treatment B's estimate is -0.55 and the placebo's is 3.18. This means that treatment B is slightly *better* than treatment A (that is, patients on treatment B, all else equal, are a little *less* likely to report pain), and the placebo P is a *lot* worse than both the others.

Extra: the P-values in the last column of the `summary` are for comparison with treatment A, the baseline. Treatment B is not significantly different from A, but the Placebo is definitely significantly different from A. What we don't get is a comparison of B with Placebo, but the (strong) implication is that both A and B are a lot better than Placebo, and A and B are not different from each other. So you would recommend "either A or B" as a treatment, since there is no basis for preferring one over the other.[2]

(h) (3 marks) Use your most recent model to do some predictions, as below. We will use all possible combinations of the three treatments, the two sexes, and the two ages 65 and 75 (which are about the 1st and 3rd quartiles of the ages, so they are more or less "typical" of the ages we have). We're going to use `predict`. Use the following line to get the "new" data to predict from:

```
new <- crossing(treatment=c("A","B","P"),sex=c("F","M"),age=c(65,75))
```

and check that this did indeed give you all possible combinations. (Whenever you want "all possible combinations", `crossing` is probably what you want.) Obtain the predicted probabilities of pain for each of these, using `predict` with a suitable `type` to make sure that you get probabilities. Put the predicted probabilities side by side with the things they're predictions for.

Solution:

The most recent and best model is the one without `duration` (not significant) but with `treatment` (significant), along with `age` and `sex`. This is the one I called `neur.2`.

Following the steps gives this. There are three treatments, two sexes and two ages, so there should be (and are) $3 \times 2 \times 2 = 12$ rows in `new`. I printed out `p` just to show you that the predicted probabilities are just a "vector" of values, so you really want to put them beside the values they are predictions for using `cbind`.

```
new <- crossing(treatment=c("A","B","P"),sex=c("F","M"),age=c(65,75))
new
```

```
## # A tibble: 12 x 3
##    treatment sex     age
##    <chr>     <chr> <dbl>
##  1 A         F        65
##  2 A         F        75
##  3 A         M        65
##  4 A         M        75
##  5 B         F        65
##  6 B         F        75
##  7 B         M        65
##  8 B         M        75
##  9 P         F        65
## 10 P         F        75
## 11 P         M        65
## 12 P         M        75
```

```
p <- predict(neur.2,new,type="response")
p
```

```
##          1          2          3          4          5          6          7
## 0.02540669 0.26945088 0.13901671 0.69553312 0.01485544 0.17583494 0.08541914
##          8          9         10         11         12
## 0.56922787 0.38507957 0.89858082 0.79502472 0.98210332
```

```
cbind(new,p)
```

```
##    treatment sex age          p
## 1          A   F  65 0.02540669
## 2          A   F  75 0.26945088
## 3          A   M  65 0.13901671
## 4          A   M  75 0.69553312
## 5          B   F  65 0.01485544
## 6          B   F  75 0.17583494
## 7          B   M  65 0.08541914
## 8          B   M  75 0.56922787
## 9          P   F  65 0.38507957
## 10         P   F  75 0.89858082
## 11         P   M  65 0.79502472
## 12         P   M  75 0.98210332
```

(i) (2 marks) By looking at the predicted probabilities from the last part, and by comparing predictions for the same sex and age, what is the effect of treatment on pain? Specifically, are treatments A and B better than the placebo? Do they differ much from each other?

Solution: For example, compare the females of age 65 (lines 1, 5, and 9). The probability of pain from the placebo is *much* higher (0.385) than from either of the real treatments (0.025 and 0.015). Since a lower probability of pain is better, treatment B is a little better than treatment A. This pattern is[3] the same no matter which age and sex you look at.[4] So looking, say, at older males would tell the same story.

This, you note, is the same picture as you got earlier by looking at the Estimates.

(j) (3 marks) Again looking at the predicted probabilities, how do you describe the effect of age? Of sex?

Solution: Likewise compare the effect of these variables when everything else is constant. For example, you can assess the effect of age by comparing rows 1 and 2 of the predictions, since this is for females under treatment A for the two different ages. Since the predicted pain probabilities are 0.025 and 0.269, the probability of pain goes up substantially with age.[5] You could have compared, say, lines 11 and 12 (males on the placebo of different ages) to get the same result. The probabilities are very different, but the comparison of them is the same.

To assess the effect of sex, you want two lines that are identical apart from sex, say rows 1 and 3 (treatment A, age 65). The probability of pain is (quite a lot) higher for the males: 0.025 for the females, 0.139 for the males. The same pattern persists for the other comparisons.

Men are such wimps!

3. Work through, or look at the problems in Chapter 20 of PASIAS that relate to ordinal responses: 20.1, 20.2, 20.8, 20.10. You don't need to worry about nominal responses yet.

   Hand the next one in.

4. Breast cancer affects a lot of women, but it can be treated if caught early enough. One way this can be done is to ask women over 40 to have a procedure called a "mammogram" every year. This is an x-ray that uses a low dose of radiation to help find any lumps or unusual areas of breast tissue, which can then be investigated further to rule out cancer or anything else malignant. There is a lot more information at `https://www.cancer.ca/en/cancer-information/diagnosis-and-treatment/tests-and-procedures/mammography/?region=on`. The kind of procedure we are concerned with here is called "screening mammography" there.

   Encouraging otherwise healthy women over 40 to get a mammogram is a public health benefit because treating breast cancer is easier and cheaper if it is caught earlier. It is, of course, also better for the women who undergo a mammogram, at least in the long run.

   A study was carried out on attitudes towards mammography in women over 40. Several things were recorded, mostly categorical:

   - `obs`: a code identifying each respondent (ignore)
   - `me`: "mammograph experience", values `less1` (last mammogram was less than 1 year ago), `more1` (last one was more than 1 year ago), `never`. This is the response variable.
   - `sympt`: response to the statement "you do not need a mammogram unless you develop symptoms", on a 4-point scale from Strongly Disagree to Strongly Agree. (Later, you will need to think carefully about what a Strongly Disagree response to this statement actually *means*.)
   - `pb`: Perceived Benefit of mammography. This is the sum of five responses on a 1–4 scale (with, therefore, a minimum score of 5 and a maximum of 20). A *low* score indicates strong agreement with the benefits of mammography.
   - `hist`: family history of breast cancer (meaning, mother or a sister), yes or no.
   - `bse`: "has anyone taught you how to examine your own breasts" (such as a doctor or a nurse), yes or no.
   - `detc`: "how likely do you think it is that a mammogram could find a new case of breast cancer?", not likely, somewhat likely, very likely.

   I had to do a bit of data reorganization (which I talk about in an Extra in my solutions), so the data is in a (to you) unusual format, which I will lead you through reading in.

   (a) (2 marks) Read in and display (some of) the data. The data frame is at `http://ritsokiguess.site/STAD29/meexp.rds`. (You won't see anything if you try to look at this.) Use `readRDS` to read it in. First store the URL in a variable called (for example) `my_url`, and then do something like this to put it in a data frame called `meexp`:

   ```
   meexp <- readRDS(url(my_url))
   ```

The extra `url()` seems to be necessary.[6]

Solution: I pretty much told you what to do:

```
my_url <- "http://ritsokiguess.site/STAD29/meexp.rds"
meexp <- readRDS(url(my_url))
meexp
```

```
## # A tibble: 412 x 7
##      obs    pb me    sympt             hist  bse   detc
##    <dbl> <dbl> <fct> <fct>             <fct> <fct> <fct>
##  1     1     7 never disagree          no    yes   somewhat
##  2     2    11 never agree             no    yes   very
##  3     3     8 never disagree          yes   yes   very
##  4     4    11 less1 disagree          no    yes   very
##  5     5     7 more1 strongly_disagree no    yes   very
##  6     6     7 never disagree          no    yes   very
##  7     7     6 more1 strongly_disagree no    yes   somewhat
##  8     8     6 never strongly_disagree no    yes   very
##  9     9     6 never agree             no    yes   very
## 10    10     6 less1 strongly_disagree no    yes   very
## # ... with 402 more rows
```

Solution:

Extra 1: `saveRDS` and `readRDS` are a way of saving an actual R object (such as a data frame) to a file. The advantage they have is that the R object that is read is *exactly* the same as the one that was saved. In this case, you'll notice that a lot of the columns of the input data frame are `fctr` or factors; if you had used `read_csv` or similar, they would have been text when you read them back in (even if they had been factors when I wrote them to a `.csv` file). Normally, this is not a problem, but here a lot of the survey responses were ordered categories (such as `sympt` and the response `me`) and I wanted to make sure the ordering was preserved for you, rather than being turned into a nonsensical alphabetical order. Hence, with a little care, you can read the data frame in exactly as I had it, which ensures that if you do things right you'll get exactly the same results I did.

Extra 2: the actual way data in these kinds of surveys are collected is to use numerical codes for the responses. These codes are then described in a "data dictionary" or "code sheet", which here looks like this:

Code Sheet for the Variables in the Mammography Experience Study
Described in Section 8.1.2 page 264.

Variable	Description	Codes/Values	Name
1	Identification Code	1-412	OBS
2	Mammograph Experience	0 = Never	ME
		1 = Within One Yea	
		2 = Over One Year Ago	
3	"You do not need a mamogram unless	1 = Strongly Agree	SYMPT
	you develop symptoms"	2 = Agree	
		3 = Disagree	
		4 = Stongly Disagree	
4	Perveived benefit of mammography*	5 - 20	PB
5	Mother or Sister with a history	0 = No, 1 = Yes	HIST
	of breast cancer		
6	"Has anyone taught you how to	0 = No, 1 = Yes	BSE
	examine your own breasts: that is BSE"		
7	"How likelyis it that a mamogram	1= Not likely	DETC
	could find a new case of	2 = Somewhat likely	
	breast cancer"	3 = Very likely	

*The variable PB is the sum of five scaled responses, each on a four point scale.
A low value is indicative of a woman with strong agreement with the benefits of mammography.

I didn't want you to have to decode those codes, so I put the categories directly into the data file. Here's what I started with:

```
meexp0 <- read_table("meexp.dat", col_names=F)
## Parsed with column specification:
## cols(
##  X1 = col_double(),
##  X2 = col_double(),
##  X3 = col_double(),
##  X4 = col_double(),
##  X5 = col_double(),
##  X6 = col_double(),
##  X7 = col_double()
## )
meexp0
## # A tibble: 412 x 7
##       X1    X2    X3    X4    X5    X6    X7
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1     0     3     7     0     1     2
##  2     2     0     2    11     0     1     3
##  3     3     0     3     8     1     1     3
##  4     4     1     3    11     0     1     3
##  5     5     2     4     7     0     1     3
##  6     6     0     3     7     0     1     3
##  7     7     2     4     6     0     1     2
##  8     8     0     4     6     0     1     3
##  9     9     0     2     6     0     1     3
## 10    10     1     4     6     0     1     3
## # ... with 402 more rows
```

The file has no row of variable names, but the numeric values are aligned in columns.[7] Saying that there are no column names gives the columns names like `X1, X2,...`. So now I have to give the columns meaningful names (the ones in my description and the code sheet) as well as meaningful values. This is rather a lot of repetitive tidyversery, thus (which I explain below):

```
meexp0 %>%
  rename(obs=X1) %>%
  mutate(X2=ifelse(X2==0, 3, X2)) %>%
  mutate(me=fct_recode(factor(X2), less1="1", more1="2", never="3")) %>%
  mutate(sympt=fct_recode(factor(X3),
                          strongly_agree="1",
                          agree="2",
                          disagree="3",
                          strongly_disagree="4")) %>%
  rename(pb=X4) %>%
  mutate(hist=fct_recode(factor(X5), no="0", yes="1")) %>%
  mutate(bse=fct_recode(factor(X6), no="0", yes="1")) %>%
  mutate(detc=fct_recode(factor(X7), not="1", somewhat="2", very="3")) %>%
  select(-X2, -X3, -X5, -X6, -X7) -> meexp
meexp
## # A tibble: 412 x 7
##      obs    pb me    sympt             hist  bse   detc
##    <dbl> <dbl> <fct> <fct>             <fct> <fct> <fct>
##  1     1     7 never disagree          no    yes   somewhat
```

```
##  2     2    11 never agree             no    yes   very
##  3     3     8 never disagree          yes   yes   very
##  4     4    11 less1 disagree          no    yes   very
##  5     5     7 more1 strongly_disagree no    yes   very
##  6     6     7 never disagree          no    yes   very
##  7     7     6 more1 strongly_disagree no    yes   somewhat
##  8     8     6 never strongly_disagree no    yes   very
##  9     9     6 never agree             no    yes   very
## 10    10     6 less1 strongly_disagree no    yes   very
## # ... with 402 more rows
```

There are two kinds of things I have to do: if the column really should be a number, I just have to give it its proper name. (`obs` and `pb`). Otherwise, it really should be categorical, and I want to replace the number each time by some text saying what the number stands for. The package `forcats` (part of the `tidyverse`) is for doing things with categorical variables, and we use `fct_recode` to translate each number into some text that represents that number. So I start with `mutate` to define a new column with the name I want, and then call on `fct_recode`. The first input of this is a categorical variable whose values we want to recode. For us, though, things like `X2` and `X3` are *numbers*, so we have to make them categorical first, hence the `factor(X2)`. After that come some name equals value pairs: new name (not in quotes), an equals sign, and the old value we want to replace (in quotes). Anything that is not listed (for example, if some of the values of `X2` were 4 or 5) is not changed. I wanted to change everything, though.

I did one other surreptitious thing. The code sheet says that in `X2`, which will become `me`, 0 means "never". `fct_recode` doesn't reorder anything, but I wanted `never` to be at the end rather than the beginning, so I first turned all the zeroes in `X2` into 3's, and then translated the 3's into `never` so that they would come out in the right order.[8]

The last step is to get rid of the columns I recoded and no longer need (`rename` gets rid of the old name but `fct_recode` does not) and save the result. I used `saveRDS` to save this for you.

A bonus of having categorical variables as R `factor`s rather than text is that they play nicely with `summary`:

```
summary(meexp)
```

```
##       obs              pb             me                    sympt      hist
##  Min.   :  1.0   Min.   : 5.000   less1:104   strongly_agree   : 40   no :368
##  1st Qu.:103.8   1st Qu.: 6.000   more1: 74   agree            : 73   yes: 44
##  Median :206.5   Median : 7.000   never:234   disagree         :160
##  Mean   :206.5   Mean   : 7.561               strongly_disagree:139
##  3rd Qu.:309.2   3rd Qu.: 9.000
##  Max.   :412.0   Max.   :17.000
##   bse            detc
##  no : 54   not     : 18
##  yes:358   somewhat:105
##            very    :289
##
##
##
```

Quantitative variables have the five-number summary as usual, but now the categorical ones display a count of how many observations there are in each category, which is very pleasant.

(b) (2 marks) What order are the categories of the response variable `me` in? Does this order make sense? Explain briefly. (Hint: investigate `levels` or `distinct` or `summary` of a data frame.)

Solution: First, find a way of displaying what categories `me` has. This one appeals to my base-R upbringing. `levels` takes a vector that is an actual `factor`, so use the dollar sign (or `with`) to get it:

```
levels(meexp$me)
```

```
## [1] "less1" "more1" "never"
```

```
with(meexp, levels(me))
```

```
## [1] "less1" "more1" "never"
```

Summarizing the data frame will give you the levels of a factor *in the order that they are*:

```
summary(meexp)
```

```
##       obs              pb             me                      sympt      hist
##  Min.   :  1.0   Min.   : 5.000   less1:104   strongly_agree   : 40   no :368
##  1st Qu.:103.8   1st Qu.: 6.000   more1: 74   agree            : 73   yes: 44
##  Median :206.5   Median : 7.000   never:234   disagree         :160
##  Mean   :206.5   Mean   : 7.561               strongly_disagree:139
##  3rd Qu.:309.2   3rd Qu.: 9.000
##  Max.   :412.0   Max.   :17.000
##   bse            detc
##  no : 54   not     : 18
##  yes:358   somewhat:105
##            very    :289
##
##
##
```

Or you can do something more Tidyverse-like by displaying the distinct ones. `distinct` is part of the Tidyverse, so you can use a pipe:

```
meexp %>% distinct(me)
```

```
## # A tibble: 3 x 1
##   me
##   <fct>
## 1 never
## 2 less1
## 3 more1
```

This, however, does not work:

```
meexp %>% count(me)
```

```
## # A tibble: 3 x 2
##   me        n
##   <fct> <int>
## 1 less1   104
## 2 more1    74
## 3 never   234
```

What `count` does is *always* to sort the categories into alphabetical order, whether they actually are that way or not.

These pieces of code show that the categories are "less than 1 year", "more than 1 year", "never" *in that order*. These go from most recent mammogram to least recent[9], so are in a sensible order.

(c) (2 marks) Why is it that a logistic regression with `polr` would make more sense than one with `multinom`? Explain briefly.

Solution: The consideration here is the kind of response variable you have. `me` has three categories (so we cannot use regular logistic regression) but they belong in order (from most recent to least, or vice versa). If the categories were *unordered* (that is, just labels like "red", "green", "blue") then `multinom` would have been the thing, but they are ordered, so we should use `polr`.

(d) (2 marks) Fit a suitable model with `polr`. You don't need to display the output. Hint: in this data frame, each row represents *one* woman, which is different from the example in lecture.

Solution: `polr` is like `glm` except that you don't need a `family`. So you need a model formula and a data frame. We know that `me` is the response, so everything else, apart from `obs`, which is just an identifier for the individual women, should be explanatory (at least for now). It doesn't matter whether those are quantitative or categorical. Also, you don't need a `weights` because each row represents only one woman. (The clue for this is whether you have a column of frequencies. In the miners example in lecture, I did; here, I don't.)

`polr` comes from `MASS` so you need to load that sometime. You probably already installed it for `boxcox` or something like that:

```
library(MASS)
meexp.1 <- polr(me~pb+sympt+hist+bse+detc, data=meexp)
```

The `summary` output from this is not very illuminating:

```
summary(meexp.1)
```

```
##
## Re-fitting to get Hessian
## Call:
## polr(formula = me ~ pb + sympt + hist + bse + detc, data = meexp)
##
## Coefficients:
##                            Value Std. Error t value
## pb                       0.18049    0.05777  3.1242
## symptagree               0.13432    0.54941  0.2445
## symptdisagree           -1.28833    0.46149 -2.7917
## symptstrongly_disagree -1.73513    0.46273 -3.7498
## histyes                 -1.01286    0.32066 -3.1586
## bseyes                  -1.07722    0.39193 -2.7485
## detcsomewhat             0.55879    0.64326  0.8687
## detcvery                -0.08519    0.61791 -0.1379
##
## Intercepts:
##              Value   Std. Error t value
## less1|more1 -2.0628  0.9422    -2.1893
## more1|never -1.0608  0.9375    -1.1315
##
## Residual Deviance: 698.9543
## AIC: 718.9543
```

Some of those are actually consistent with the `drop1` output below, but not so clearly.

(e) (3 marks) Use `drop1` with `test="Chisq"` to demonstrate that none of the explanatory variables should be removed.

Solution: Feed `drop1` a fitted model and a `test=`:

```
drop1(meexp.1, test="Chisq")
```

```
## Single term deletions
##
## Model:
## me ~ pb + sympt + hist + bse + detc
##        Df    AIC    LRT  Pr(>Chi)
## <none>    718.95
## pb      1 727.04 10.081  0.001498 **
## sympt   3 750.86 37.902 2.965e-08 ***
## hist    1 727.05 10.091  0.001490 **
## bse     1 725.47  8.516  0.003520 **
## detc    2 721.10  6.145  0.046310 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All the P-values are less than 0.05, so all the explanatory variables have some impact on "mammogram experience", and none of them should be removed.

Note that `drop1` does the right thing with categorical *explanatory* variables. To test something like `sympt` with four categories, we want to see whether *any* of the categories have a different impact on the recency of mammogram than the others, so we want to test `sympt` *as a whole*, with $4 - 1 = 3$ degrees of freedom. `drop1` does this properly.

Extra: to come back to the `summary` output from above:

```
summary(meexp.1)
```

```
##
## Re-fitting to get Hessian
## Call:
## polr(formula = me ~ pb + sympt + hist + bse + detc, data = meexp)
##
## Coefficients:
##                            Value Std. Error t value
## pb                       0.18049    0.05777  3.1242
## symptagree               0.13432    0.54941  0.2445
## symptdisagree           -1.28833    0.46149 -2.7917
## symptstrongly_disagree -1.73513    0.46273 -3.7498
## histyes                 -1.01286    0.32066 -3.1586
## bseyes                  -1.07722    0.39193 -2.7485
## detcsomewhat             0.55879    0.64326  0.8687
## detcvery                -0.08519    0.61791 -0.1379
##
## Intercepts:
##             Value   Std. Error t value
## less1|more1 -2.0628  0.9422    -2.1893
## more1|never -1.0608  0.9375    -1.1315
##
## Residual Deviance: 698.9543
## AIC: 718.9543
```

The last column of the table of coefficients is the estimate divided by its standard error, a z or t or something like that. There isn't actually a test (here), but as a guideline, anything bigger in size than about 2 is probably meaningful. Thus the `pb` score has an impact on getting a mammogram, and you can also assess the two-level categorical variables `hist` and `bse`; it looks

as if those are significant also.

The other two categorical variables `sympt` and `detc` require more work to assess. As usual, the missing category is the baseline, and the last column compares the level shown to the baseline. For `sympt` the baseline is `strongly_agree`; `agree` is very close to this, but the other two categories are very different. This suggests (as we find later) that women who agree and who disagree with the statement in `sympt` will be very different on how recently they've had a mammogram. (The strength of agreement or disagreement, though, doesn't have much impact.)

The baseline category for `detc` is `not`. `very` is close to `not`, while it is `somewhat` that is different. None of those t-values is very big, which makes it a bit surprising that `detc` came out significant.

(f) (3 marks) The usual way of understanding a model of this kind is to do some predictions. This is made harder here by the large number of significant explanatory variables. One place to start is to find some "typical" values to work from. To set this up, find the median of each of the quantitative variables, and find the combination of categories of all the categorical variables that has the most observations. (Hint: for the categorical variables, count them all, and put the frequencies in order.)

Solution: The only quantitative variable is `pb`, which was a numerical score on a questionnaire:

```
meexp %>% summarize(med=median(pb))
```

```
## # A tibble: 1 x 1
##     med
##   <dbl>
## 1     7
```

The other four explanatory variables are all categorical. To count combinations of them, put them *all* inside a `count` (in any order):

```
meexp %>% count(sympt, hist, bse, detc)
```

```
## # A tibble: 36 x 5
##    sympt          hist  bse   detc         n
##    <fct>          <fct> <fct> <fct>    <int>
##  1 strongly_agree no    no    not          2
##  2 strongly_agree no    no    somewhat     3
##  3 strongly_agree no    no    very         3
##  4 strongly_agree no    yes   not          2
##  5 strongly_agree no    yes   somewhat    13
##  6 strongly_agree no    yes   very        16
##  7 strongly_agree yes   yes   very         1
##  8 agree          no    no    not          4
##  9 agree          no    no    somewhat     6
## 10 agree          no    no    very         4
## # ... with 26 more rows
```

The column `n` has the counts in it. But I want to find the biggest one, so if I arrange the counts in descending order, that will do it:

```
meexp %>% count(sympt, hist, bse, detc) %>%
  arrange(desc(n))
```

```
## # A tibble: 36 x 5
##    sympt             hist  bse   detc         n
##    <fct>             <fct> <fct> <fct>    <int>
##  1 disagree          no    yes   very        93
##  2 strongly_disagree no    yes   very        92
##  3 disagree          no    yes   somewhat    27
##  4 agree             no    yes   very        26
```

```
##  5 agree             no    yes   somewhat    23
##  6 strongly_disagree no    yes   somewhat    20
##  7 strongly_agree    no    yes   very        16
##  8 disagree          yes   yes   very        15
##  9 strongly_agree    no    yes   somewhat    13
## 10 disagree          no    no    very        11
## # ... with 26 more rows
```

The most common combination was:

- disagreeing that a mammogram should only be obtained when there are cancer symptoms
- having no family history of breast cancer
- having been shown how to do a breast self-exam
- thinking it was very likely that a mammogram can reveal a new case of breast cancer

The top two combinations (with the second one being "strongly disagree" rather than "disagree" with everything else being the same) are a long way ahead of the rest.

This says nothing about the recency of mammograms, though. That we're coming to.

(g) (4 marks) Investigate the effect of family history of breast cancer on the recency of a mammogram. To do this:

- create a new data frame with two rows: the first row contains what you got in the previous part, and the second row is the same as the first, except that the family history variable takes the other value. The names of the variables in your new data frame must be the same as the ones in the original data frame.
- Then obtain predicted probabilities for each response category for each of those two rows, displaying the predictions next to the values they are predictions for.
- Finally, explain whether the predictions are what you'd expect, in language that a hospital *administrator* or other public health official would be able to understand.

Solution: All right, that's a rather beefy question.

The first thing is to create a data frame of values to predict for. The first row comes from the previous part: 7 for `pb`, `disagree` for `sympt`, `no` for `hist`, `yes` for `bse`, and `very` for `detc`. The second row is a repeat of the first row except that `hist` is now `yes` rather than `no`.

I think that `tribble` is the easiest way to organize this, along with some judicious copying, pasting and editing:

```
new <- tribble(
  ~sympt,     ~hist, ~bse,  ~detc, ~pb,
  "disagree", "yes", "yes", "very", 7,
  "disagree",  "no", "yes", "very", 7,
)
new
```

```
## # A tibble: 2 x 5
##   sympt    hist  bse   detc     pb
##   <chr>    <chr> <chr> <chr> <dbl>
## 1 disagree yes   yes   very      7
## 2 disagree no    yes   very      7
```

`tribble` will let you add extra space to line things up.

Next, feed this into `predict`:

```
p <- predict(meexp.1, new, type="probs")
cbind(new, p)
##      sympt hist bse detc pb     less1     more1     never
## 1 disagree  yes yes very  7 0.5343043 0.2232586 0.2424371
## 2 disagree   no yes very  7 0.2941258 0.2374575 0.4684167
```

So, what does this mean? Note that what we're doing is just like regression: to find the effect of something on the response, we leave everything else the same and change just the something. So here we assess the effect of family history by seeing what happens when it changes in terms of the recency of mammogram, all else (literally) equal.

Here, a woman with family breast cancer history is much more likely to have had a mammogram in the last year (or, flipping it around, a woman with *no* family breast cancer history is much more likely *never* to have had a mammogram).

This makes logical sense because if your mother or sister had breast cancer, you would want to take extra care not to get it yourself, and a mammogram is a way at least of catching it before it gets too serious.

(h) (4 marks) Repeat the previous part, but this time assessing the effect of answer to the statement "you do not need a mammogram unless you develop symptoms" on the recency of mammogram. Again, do the results make practical sense?

Solution: The same principle as the previous part. Start with the same first line of your data frame `new`, then copy and paste it, changing `sympt` to one of its other values. There are four of them, so your `new` will have four rows, thus:

```
new <- tribble(
  ~sympt, ~hist, ~bse, ~detc, ~pb,
  "disagree", "yes", "yes", "very", 7,
  "strongly_disagree", "yes", "yes", "very", 7,
  "agree", "yes", "yes", "very", 7,
  "strongly_agree", "yes", "yes", "very", 7
)
new
## # A tibble: 4 x 5
##   sympt             hist  bse   detc     pb
##   <chr>             <chr> <chr> <chr> <dbl>
## 1 disagree          yes   yes   very      7
## 2 strongly_disagree yes   yes   very      7
## 3 agree             yes   yes   very      7
## 4 strongly_agree    yes   yes   very      7
```

Then do the exact same prediction again and display the results in the same way:

```
p <- predict(meexp.1, new, type="probs")
cbind(new, p)
##               sympt hist bse detc pb     less1     more1     never
## 1          disagree  yes yes very  7 0.5343043 0.2232586 0.2424371
## 2 strongly_disagree  yes yes very  7 0.6420410 0.1880348 0.1699242
## 3             agree  yes yes very  7 0.2166640 0.2129851 0.5703510
## 4    strongly_agree  yes yes very  7 0.2403254 0.2225020 0.5371727
```

The more a woman disagrees with the statement, the more likely she is to have had a recent mammogram. Conversely, the more a woman agrees with the statement, the more likely she is *never* to have had a mammogram.

The statement is logically kind of backwards, so you will need to think carefully to disentangle

it. Disagreeing with the statement means that there are perceived to be other good reasons to get a mammogram, such as to catch anything early even if you don't have symptoms, which in turn makes it *more* likely that such a woman would have had one recently. Agreeing with the statement is the opinion that a woman without symptoms doesn't need a mammogram, and thus such a woman probably wouldn't get one. All makes sense.

Extra: how about `detc`? Same again:

```
new <- tribble(
  ~sympt,      ~hist, ~bse,  ~detc, ~pb,
  "disagree", "yes", "yes", "very", 7,
  "disagree", "yes", "yes", "somewhat", 7,
  "disagree", "yes", "yes", "not", 7,
)
new
```

```
## # A tibble: 3 x 5
##   sympt    hist  bse   detc        pb
##   <chr>    <chr> <chr> <chr>    <dbl>
## 1 disagree yes   yes   very         7
## 2 disagree yes   yes   somewhat     7
## 3 disagree yes   yes   not          7
```

The answer to this was how likely the respondent thought that a mammogram could uncover a new case of breast cancer. Our suspicion before is that `somewhat` might be different from the others:

```
p <- predict(meexp.1, new, type="probs")
cbind(new, p)
```

```
##      sympt hist bse     detc pb     less1     more1     never
## 1 disagree  yes yes     very  7 0.5343043 0.2232586 0.2424371
## 2 disagree  yes yes somewhat  7 0.3760045 0.2453706 0.3786249
## 3 disagree  yes yes      not  7 0.5130587 0.2285176 0.2584237
```

This is indeed how it plays out. A woman who thinks that a mammogram is "very likely" or "not likely" to uncover a new case is more likely to have a recent mammogram than one who thinks it's "somewhat likely". I don't understand this, so I didn't ask you about it. I would have guessed that the more likely a woman thought a mammogram would reveal a new case, the more likely she would have had one recently. But apparently not.

Extra: from a public health point of view, you would want to encourage women aged over 40 to get annual mammograms, because the cost of doing the mammograms is dwarfed by the cost of otherwise having to treat (possibly advanced) cases of cancer down the road. The point of a study like this one is to see what is linked with women being more likely to have actually *had* a recent mammogram. As we've seen, family history is one thing, but attitudes towards "you should only get a mammogram if you have symptoms" also had a big effect. You can't (as a public health agency) do anything about family history, but you can do a lot to educate women on the value of getting frequent mammograms even without symptoms, perhaps by starting with family doctors. My doctor is often telling me about things that someone "of my age" should be worrying about.

Extra extra: none of this is in any way related to whether mammograms actually do do any good. That would be another study. You would have to compare women who have mammograms with women who don't, and compare the incidence and severity of breast cancer between the two groups (adjusting for other differences between the groups). This would have to be an observational study, since it is unethical to make some women get mammograms while preventing others from doing so. The point of the study in this question is to assume that getting frequent mammograms is a good thing, and then to ask how this might be made to happen.

Notes

[1]Like ANOVA.
[2]The slight difference between A and B was just chance.
[3]As the model dictates.
[4]If it were not, you would have an ANOVA-style *interaction* between the variables concerned.
[5]This is why age was significant.
[6]This is because the input to `readRDS` has to be what R knows of as a "connection". If you just give it the URL, it'll think it's some kind of file which it then won't be able to find, but if you wrap it in `url()` it knows that it's really a URL and should be treated as such.
[7]I wasn't sure whether they were this or tabs, but I tried this and it worked.
[8]Think about what will happen if I leave the `never` as 0: will the three categories then be in a sensible order?
[9]You can't get less recent than "never"!