Assignment 4

Instructions (same as for STAC32): Make an R Notebook and in it answer the question(s) 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, that is *before* you start work on the assignment. 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 5 hours to complete this assignment after you start it.

1. Three brands of popcorn (labelled Gourmet, National, and Generic) and two types of popcorn popper (oil and air) were compared. Three batches of each brand were popped in each type of popper, and the number of cups of popped popcorn produced were measured each time. Our aims are to discover whether one of the types of popper consistently produces more popcorn than the other, whether the brands differ in the amount of popcorn produced, and whether the amount of popcorn produced depends on the combination of popper and brand. If there are any differences, we would like to make recommendations for the best brand. The data are in http://ritsokiguess.site/STAD29/popcorn_long.csv.

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

Solution:

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

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   popper = col_character(),
##   brand = col_character(),
##   cups = col_double()
## )
```

```
popcorn
```

```
## # A tibble: 18 x 3
##    popper brand     cups
##    <chr>  <chr>    <dbl>
##  1 oil    Gourmet    5.5
##  2 oil    National   4.5
##  3 oil    Generic    3.5
##  4 oil    Gourmet    5.5
##   5 oil    National   4.5
##   6 oil    Generic    4
##   7 oil    Gourmet    6
##   8 oil    National   4
##   9 oil    Generic    3
##  10 air    Gourmet    6.5
##  11 air    National   5
##  12 air    Generic    4
##  13 air    Gourmet    7
##  14 air    National   5.5
##  15 air    Generic    5
##  16 air    Gourmet    7
##  17 air    National   5
##  18 air    Generic    4.5
```

Extra: there is rather a long story about how the data came to be this way. In my original plans, I was going to have you work through this, but this got lost in my quest to make this assignment less long,[1] so the story is here.

Here's the original layout of the data, laid out as it would be in a spreadsheet. I stored this in http://ritsokiguess.site/STAD29/popcorn.txt:

```
popper       Gourmet  National   Generic
oil           5.5000    4.5000    3.5000
              5.5000    4.5000    4.0000
              6.0000    4.0000    3.0000
air           6.5000    5.0000    4.0000
              7.0000    5.5000    5.0000
              7.0000    5.0000    4.5000
```

You'll recognize this at once as aligned columns, and so you'll be thinking `read_table`. But there are some extra problems: the blanks in the `popper` column (meant to signify "the same as above"), and also the three short columns of amounts of popcorn instead of one long one (suggesting that there will be a `pivot_longer` sometime).

I was thinking ahead a bit when I dealt with this. Filling in blank values by copying the value above is what `fill` does, but that works with the values-to-be-filled being *missing* rather than empty. This might seem like an odd distinction to be making, but I mean that `fill` requires `NA` values instead of ones that have nothing in them at all. So the first two steps are (i) to read in the data and (ii) to replace the empty values with `NA`. This is a common thing to want to do, so we can do both steps in one `read_table`, thus:

```r
my_other_url <- "http://ritsokiguess.site/STAD29/popcorn.txt"
popcorn0 <- read_table(my_other_url, na="")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   popper = col_character(),
##   Gourmet = col_double(),
##   National = col_double(),
##   Generic = col_double()
## )
```

```
popcorn0
```

```
## # A tibble: 6 x 4
##   popper Gourmet National Generic
##   <chr>    <dbl>    <dbl>   <dbl>
## 1 oil        5.5      4.5     3.5
## 2 <NA>       5.5      4.5     4
## 3 <NA>       6        4       3
## 4 air        6.5      5       4
## 5 <NA>       7        5.5     5
## 6 <NA>       7        5       4.5
```

I'm using a "disposable" name for this dataframe since I have some other work to do with it yet.

If you don't see that, another (more obvious) way is to read the data file in as normal and then redefine the `popper` column:

```
popcorn0a <- read_table(my_other_url)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   popper = col_character(),
##   Gourmet = col_double(),
##   National = col_double(),
##   Generic = col_double()
## )
```

```
popcorn0a
```

```
## # A tibble: 6 x 4
##   popper Gourmet National Generic
##   <chr>    <dbl>    <dbl>   <dbl>
## 1 "oil"      5.5      4.5     3.5
## 2 ""         5.5      4.5     4
## 3 ""         6        4       3
## 4 "air"      6.5      5       4
## 5 ""         7        5.5     5
## 6 ""         7        5       4.5
```

```
popcorn0a %>% mutate(popper=ifelse(popper=="", NA, popper)) -> popcorn0b
popcorn0b
```

```
## # A tibble: 6 x 4
##   popper Gourmet National Generic
##   <chr>    <dbl>    <dbl>   <dbl>
## 1 oil        5.5      4.5     3.5
## 2 <NA>       5.5      4.5     4
## 3 <NA>       6        4       3
## 4 air        6.5      5       4
## 5 <NA>       7        5.5     5
## 6 <NA>       7        5       4.5
```

Comment: people should really not leave blank lines, since there is doubt about what such lines mean, but you see it all the time, especially in spreadsheets. This paper, section 5 talks about that. (The whole paper is very readable and worth your while reading.)

So now we are going to use `fill` to replace those `NA` values with the non-missing values above them. When you look at how `fill` works, you'll see that you can use it to replace missings with the value above them, or the value below, with the former being the default. That means that the filling part is as simple as this:

```
popcorn0 %>% fill(popper)
```

```
## # A tibble: 6 x 4
##   popper Gourmet National Generic
##   <chr>    <dbl>    <dbl>   <dbl>
## 1 oil        5.5      4.5     3.5
## 2 oil        5.5      4.5     4
## 3 oil        6        4       3
## 4 air        6.5      5       4
## 5 air        7        5.5     5
## 6 air        7        5       4.5
```

Check that this works first. As for tidying, the three columns with numbers in them are the different brands of popcorn, with the numbers being the numbers of cups of popcorn produced, and I hope by now you recognize this as a standard `pivot_longer`:

```
popcorn0 %>% fill(popper) %>%
  pivot_longer(-popper, names_to="brand", values_to="cups") -> popcorn_good
popcorn_good
```

```
## # A tibble: 18 x 3
##    popper brand     cups
##    <chr>  <chr>    <dbl>
##  1 oil    Gourmet    5.5
##  2 oil    National   4.5
##  3 oil    Generic    3.5
##  4 oil    Gourmet    5.5
##  5 oil    National   4.5
##  6 oil    Generic    4
##  7 oil    Gourmet    6
##  8 oil    National   4
##  9 oil    Generic    3
## 10 air    Gourmet    6.5
## 11 air    National   5
## 12 air    Generic    4
## 13 air    Gourmet    7
## 14 air    National   5.5
## 15 air    Generic    5
## 16 air    Gourmet    7
## 17 air    National   5
## 18 air    Generic    4.5
```

This is my "good" dataframe, so I give it a "good" name. One column for each of our three variables. The version saved below is the one I had you read:

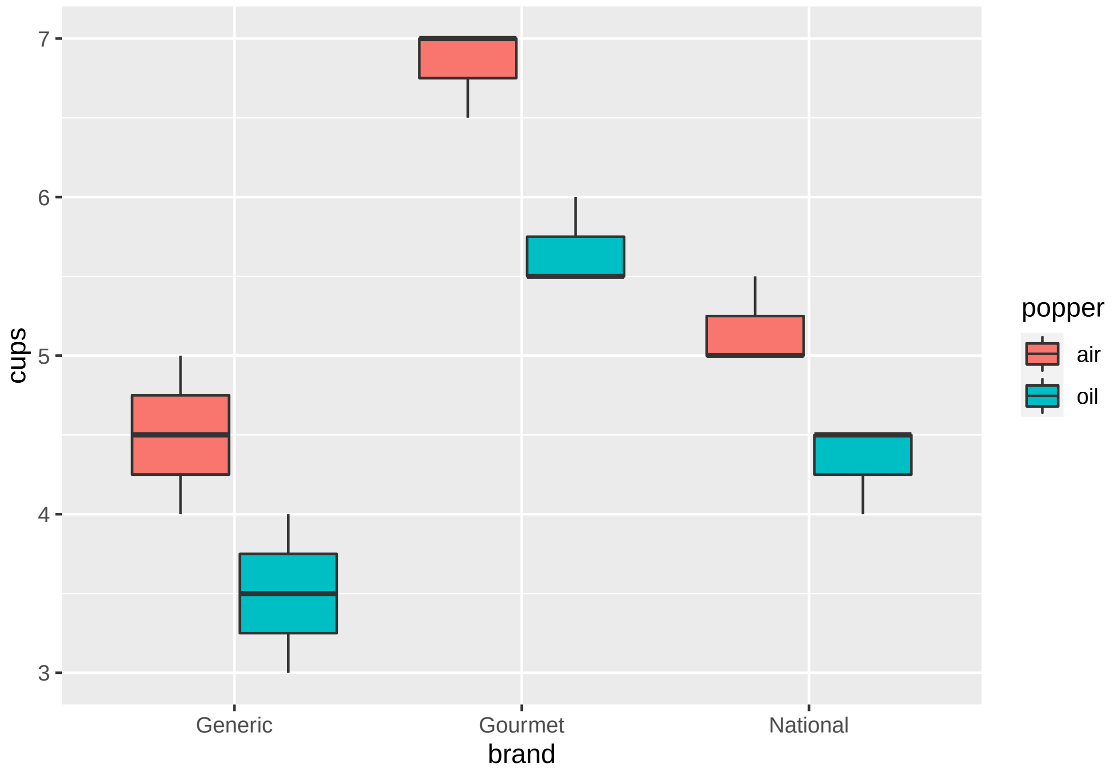
```
write_csv(popcorn_good, "popcorn_long.csv")
```

(b) Make a suitable plot of the data, and comment briefly on what it tells you.

Solution:

I think the best graph here is a grouped boxplot, since there are two categorical variables. (It is a good idea to say this, because it demonstrates your understanding.)

```
ggplot(popcorn, aes(x=brand, fill=popper, y=cups)) + geom_boxplot()
```



I chose to have `brand` as my `x`, since there are more brands (3) than poppers (2).

My personal preference is for filled boxes, but if you prefer the boxes coloured around the outside, go with that.

This plot tells us that for any brand, the air popper produces more popcorn than the oil popper does. The Gourmet brand seems to produce more popcorn than either of the other two brands (regardless of the type of popper used).

Extra: something that often happens in this type of analysis, there are here only 18 observations altogether, so each boxplot is based on only *three* observations. This means that you don't want to be taking any non-normality or outliers *too* seriously here. (That's why I typically look at regression residuals for these.) Some of the boxes here don't even look much like boxes, because the median is the same as one of the quartiles. All it takes for that to happen is that two of the three observations are the same as each other, which can easily happen here since the amount of popcorn was only measured to the nearest half cup. Popped popcorn is hard to measure the volume of.

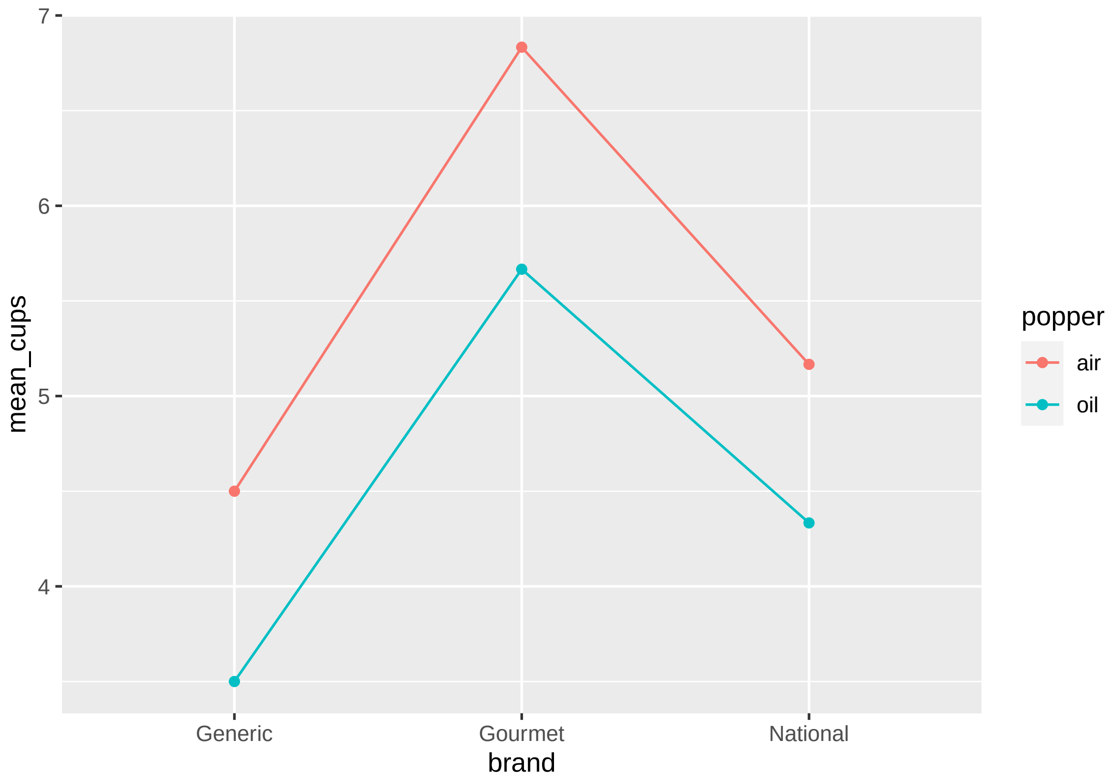
(c) Make an interaction plot. What does this tell you about the data?

Solution:

My habit for these is to get the summaries first and then make the plot (since I can remember how to do that). If you like to have the plot calculate the summaries for you, go right ahead. The first way is this:

```
popcorn %>% group_by(brand, popper) %>%
  summarize(mean_cups = mean(cups)) %>%
  ggplot(aes(x=brand, colour=popper, group=popper, y=mean_cups)) +
  geom_point() + geom_line()
```

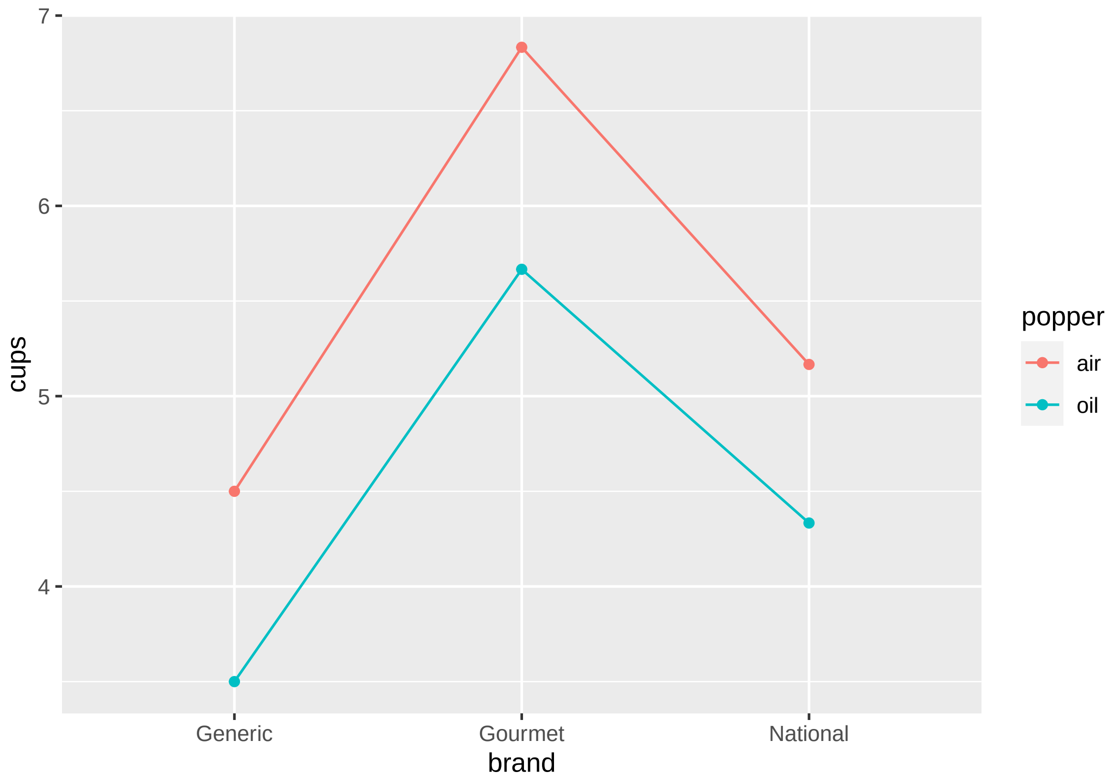
```
## `summarise()` has grouped output by 'brand'. You can override using the `.groups` argument.
```



(or save your summary in something and then plot the something).

And the second way is this:

```
ggplot(popcorn, aes(x=brand, y=cups, colour=popper, group=popper)) +
  stat_summary(fun="mean", geom="point") +
  stat_summary(fun="mean", geom="line")
```



The interaction plot shows very much parallel lines, so we do not expect to see an interaction. That is, there should be an effect of popper and an effect of brand, but not of the combination between them.

Also, the air popper seems better for each brand and the Gourmet brand seems better for both poppers. We ought to be circumspect[2] about our conclusions at this point, since we haven't done any tests yet. Also, the interaction plot tells us nothing about variability or normality, so we will have to assess those some other way, which we will get to later.

(d) Run a suitable two-way analysis of variance, and display the results. You might need to run a second analysis as well, depending on the results of the first. What does your final analysis tell you about the data?

Solution:

The first step is to fit a two-way ANOVA *with* interaction (same principle as for a regression: start with everything, *then* see what you can remove):

```
popcorn.1 <- aov(cups ~ popper*brand, data = popcorn)
summary(popcorn.1)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## popper        1  4.500   4.500    32.4   0.0001 ***
## brand         2 15.750   7.875    56.7 7.68e-07 ***
## popper:brand  2  0.083   0.042     0.3   0.7462
## Residuals    12  1.667   0.139
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction is (rather as we suspected) nowhere near significant, so the next step is to remove it. Either write out the new model (using `+` in place of `*`), or use `update`. This last requires care: `popper*brand` denotes the interaction term *and* the two main effects `popper` and `brand`; just the interaction is `popper:brand`, and *that* is what we want to get rid of:

```
popcorn.2 <- update(popcorn.1, .~.-popper:brand)
summary(popcorn.2)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)    
## popper       1   4.50   4.500      36 3.25e-05 ***
## brand        2  15.75   7.875      63 1.00e-07 ***
## Residuals   14   1.75   0.125                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Also rather as we expected, the two main effects are strongly significant. (Removing the interaction term has decreased their P-values somewhat.) That means that the different poppers do not produce the same amount of popcorn, and the different brands also do not all produce the same amount of popcorn.

(e) If warranted, run Tukey's method. What does this tell you about the data, in particular about which popper(s) and brand(s) of popcorn produce the most popcorn?

Solution:

Having found significant differences between the poppers and among the brands, we now use Tukey to find out where these are. Thinking carefully, we really only need to use Tukey on brands, because there are only two poppers and we already know they are different. But there is no problem running Tukey the way you expect:

```
TukeyHSD(popcorn.2)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = cups ~ popper + brand, data = popcorn)
##
## $popper
##         diff       lwr        upr    p adj
## oil-air   -1 -1.357464 -0.6425356 3.25e-05
##
## $brand
##                   diff        lwr        upr     p adj
## Gourmet-Generic   2.25  1.7157499  2.7842501 0.0000001
## National-Generic  0.75  0.2157499  1.2842501 0.0066142
## National-Gourmet -1.50 -2.0342501 -0.9657499 0.0000102
```

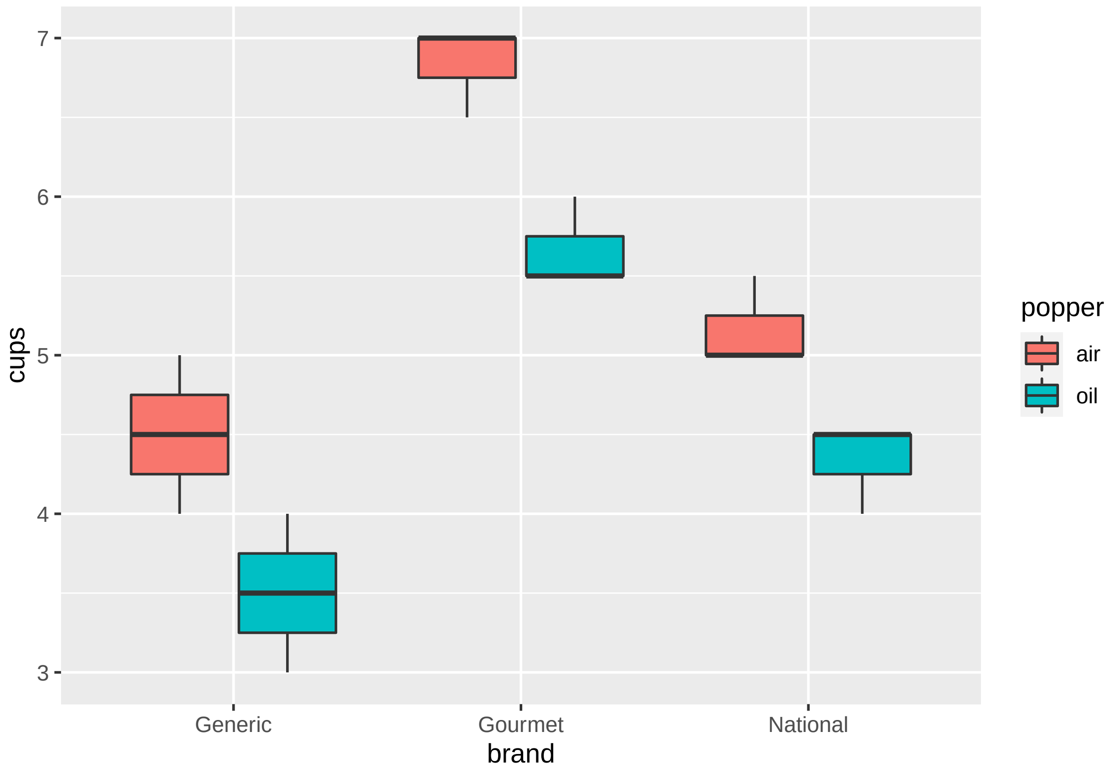
We already said that there was no real value in looking at `popper`, but the `brand` piece is informative: all three brands are significantly different.

One of the brands (and one of the poppers) is therefore best (in terms of producing the most popcorn).

You can work out which ones directly from the Tukey output: `oil` is less than `air`, so `air` is better (produces more popcorn). Also `Gourmet` is better than both `Generic` and `National`, so that's the best brand.

If you don't see that, go back to the grouped boxplot:

```
ggplot(popcorn, aes(x=brand, fill=popper, y=cups)) + geom_boxplot()
```



The Gourmet brand is best, and the air popper is better than the oil one. Since there is no interaction, there is a uniformly best brand (over both poppers) and a uniformly better popper (over all brands). (If there had been a significant interaction, the best brand might depend on which popper you were looking at. Note also that the difference between the two poppers for each brand is about the same all the way across.)

Or, if you like, work out the means by group:

```
popcorn %>% group_by(brand, popper) %>%
  summarize(mean_cups = mean(cups)) %>%
  arrange(desc(mean_cups))
```

```
## `summarise()` has grouped output by 'brand'. You can override using the `.groups` argument.
```

```
## # A tibble: 6 x 3
## # Groups:   brand [3]
##   brand    popper mean_cups
##   <chr>    <chr>      <dbl>
## 1 Gourmet  air         6.83
## 2 Gourmet  oil         5.67
## 3 National air         5.17
## 4 Generic  air         4.5
## 5 National oil         4.33
## 6 Generic  oil         3.5
```

with the same results. Find a way, and talk about what you see. (This last way is really doing predictions, since the fitted value for each group is just the mean of that group.)

Extra: in this case, there is an unambiguously best brand and popper, so your recommendation is a clear one. If two of the brands had not been significantly different, you might have had to recommend *two* brands: the one that came out best here, and the second-best one here that was not significantly worse than the best one. (This issue shows up in the other question, about the recommended medicine for males.) Another point is that we didn't have much data here, so it was perhaps surprising that the conclusions came out so clearly.

(f) Run your preferred analysis as a regression. From this, make a normal quantile plot of the residuals and plot the residuals against the fitted values. Do you see any problems? Explain briefly.

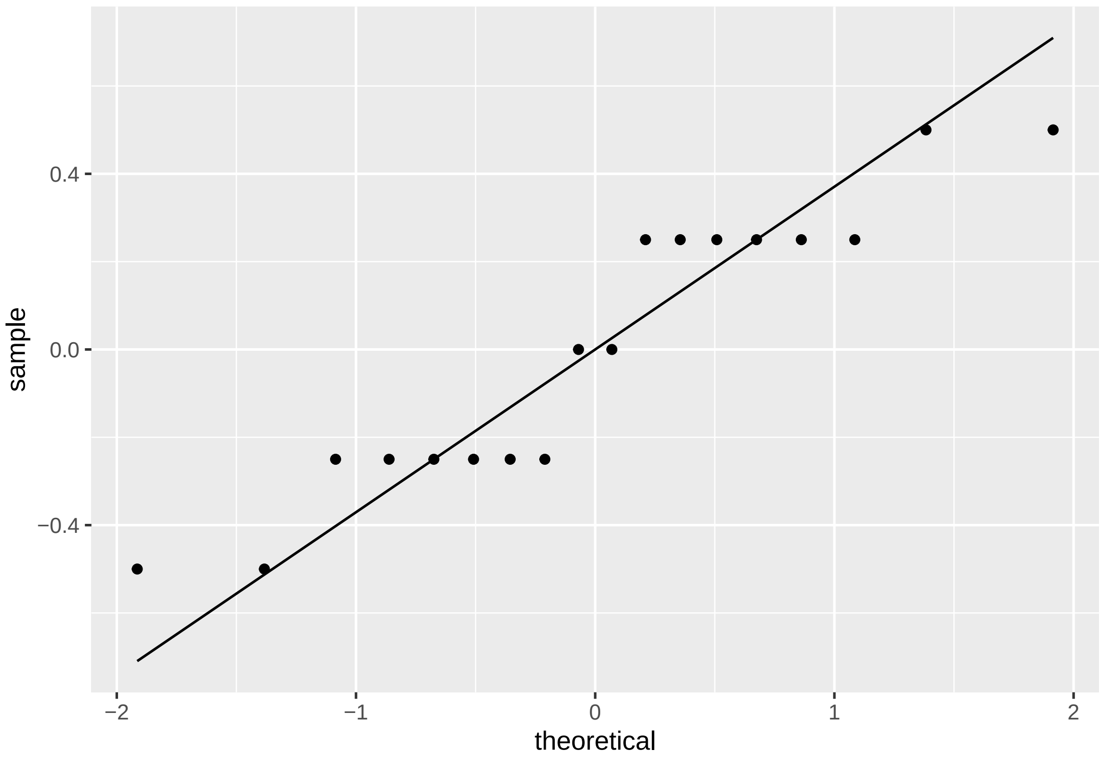
Solution:

Our best model has just the two main effects, and not the non-significant interaction:

```
popcorn.3 <- lm(cups~brand+popper, data=popcorn)
```

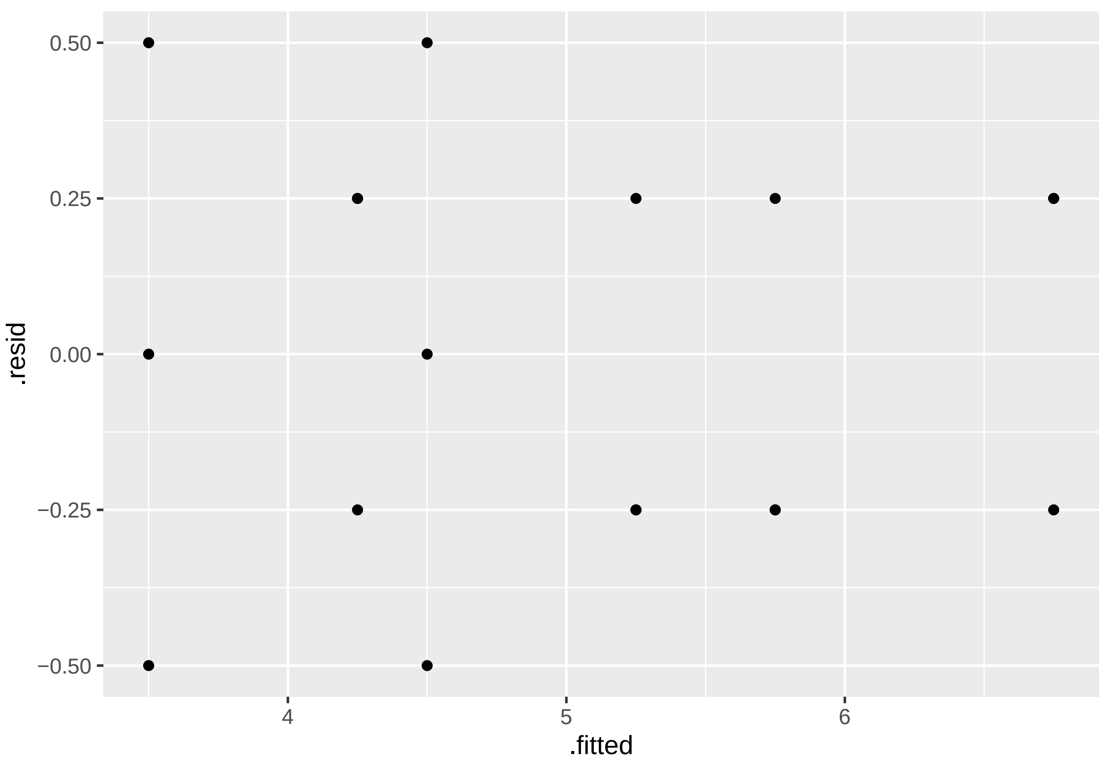
We don't even need to look at the regression, since the output it gives you is equivalent to the aov, but you can if you want. A normal quantile plot of the residuals goes this way:

```
ggplot(popcorn.3, aes(sample=.resid)) + stat_qq() + stat_qq_line()
```



These look pretty good given the discreteness. And residuals against fitted values, which also look pretty good:

```
ggplot(popcorn.3, aes(x=.fitted, y=.resid)) + geom_point()
```

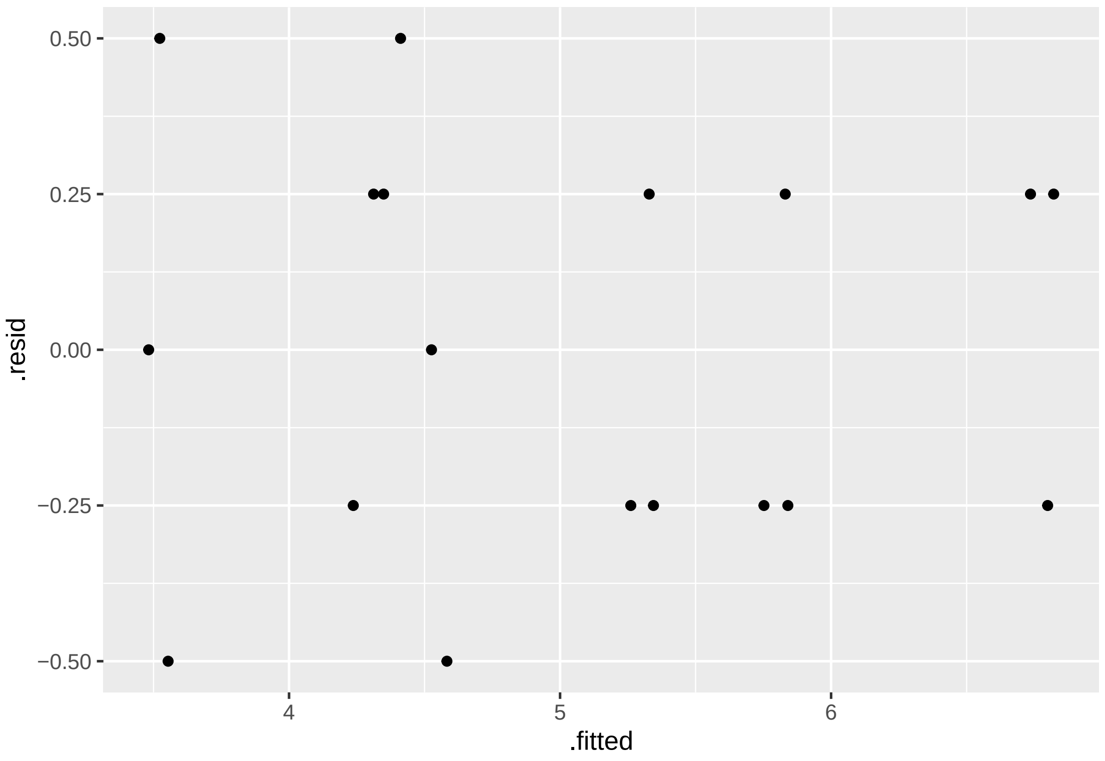


This seems pretty much random, without pattern.

Extra: I wanted to say a bit more about the residuals and fitted values. In an ANOVA, there is one fitted value per group, so in this case there are only six of them ($3 \times 2 = 6$). A residual is the difference between an observation and its fitted value. In this particular example, the observations were all whole numbers or whole numbers plus a half, and the fitted values were roundish numbers, so a lot of the residuals were the same as each other, explaining the discreteness in the normal quantile plot.

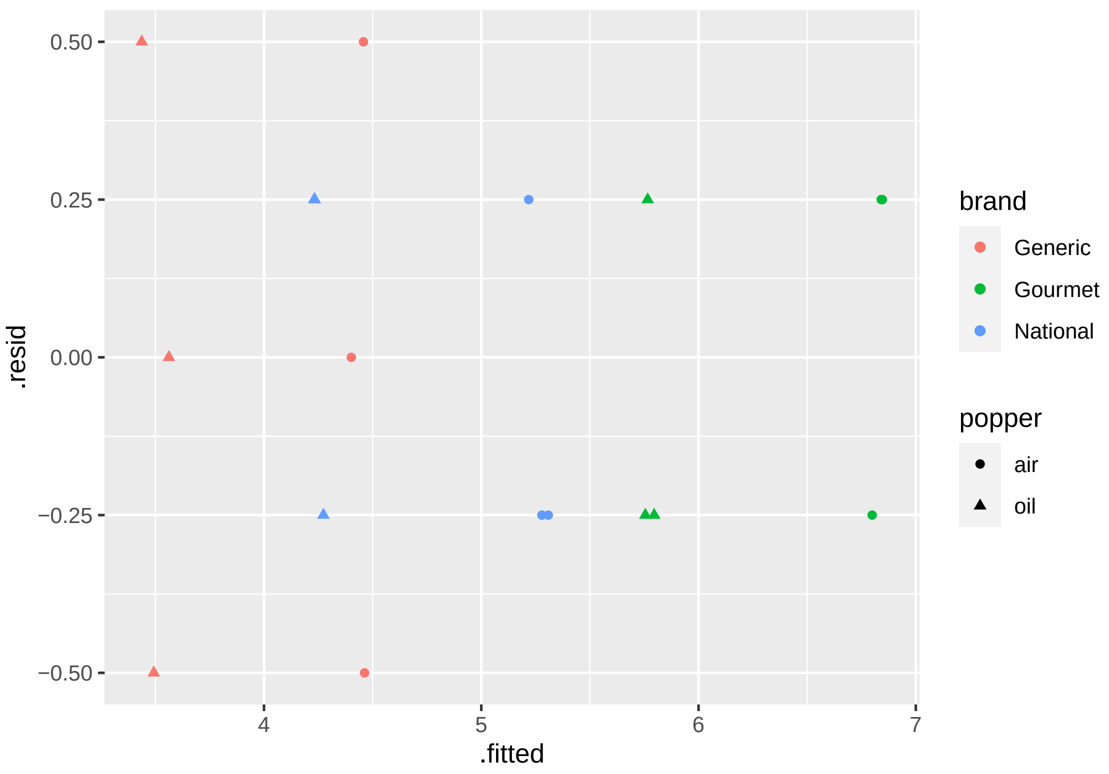
Turning to the residuals vs. fitted values, there are only six different fitted values, and so you see the points on that plot in six vertical columns. Each group has three observations in it, so there should be three dots in each column. Why are there sometimes only two? Let's spread the points out a bit by jittering[3] them:

```
ggplot(popcorn.3, aes(x=.fitted, y=.resid)) + geom_jitter()
```



Some of the points were actually pairs of points that plotted in the same place. I'll take one more step, distinguishing the points by which group they're from, which requires a bit of care since the group is defined by *two* categorical variables:

```
ggplot(popcorn.3, aes(x=.fitted, y=.resid, colour=brand, shape=popper)) + geom_jitter()
```



The trios of more or less vertically-aligned points with the same colour and same shape are in the same group. The residuals within a group are 0 and ±0.5, or ±0.25, with two of one and one of the other. This shows directly that a lot of the residuals will be the same as each other. This happens because the observations are all a whole number or a whole number plus a half, and the fitted values are roundish numbers also:[4]

```
popcorn.3 %>% augment(popcorn)
```

```
## # A tibble: 18 x 9
##    popper brand     cups .fitted    .resid .std.resid  .hat .sigma   .cooksd
##    <chr>  <chr>    <dbl>   <dbl>     <dbl>      <dbl> <dbl>  <dbl>     <dbl>
##  1 oil    Gourmet    5.5    5.75 -2.50e- 1     -0.802 0.222  0.358 4.59e- 2
##  2 oil    National   4.5    4.25  2.50e- 1      0.802 0.222  0.358 4.59e- 2
##  3 oil    Generic    3.5    3.5  -8.88e-16      0     0.222  0.367 4.36e-31
##  4 oil    Gourmet    5.5    5.75 -2.50e- 1     -0.802 0.222  0.358 4.59e- 2
##  5 oil    National   4.5    4.25  2.50e- 1      0.802 0.222  0.358 4.59e- 2
##  6 oil    Generic    4      3.5   5.00e- 1      1.60  0.222  0.331 1.84e- 1
##  7 oil    Gourmet    6      5.75  2.50e- 1      0.802 0.222  0.358 4.59e- 2
##  8 oil    National   4      4.25 -2.50e- 1     -0.802 0.222  0.358 4.59e- 2
##  9 oil    Generic    3      3.5  -5.00e- 1     -1.60  0.222  0.331 1.84e- 1
## 10 air    Gourmet    6.5    6.75 -2.50e- 1     -0.802 0.222  0.358 4.59e- 2
## 11 air    National   5      5.25 -2.50e- 1     -0.802 0.222  0.358 4.59e- 2
## 12 air    Generic    4      4.5  -5.00e- 1     -1.60  0.222  0.331 1.84e- 1
## 13 air    Gourmet    7      6.75  2.50e- 1      0.802 0.222  0.358 4.59e- 2
## 14 air    National   5.5    5.25  2.50e- 1      0.802 0.222  0.358 4.59e- 2
## 15 air    Generic    5      4.5   5.00e- 1      1.60  0.222  0.331 1.84e- 1
## 16 air    Gourmet    7      6.75  2.50e- 1      0.802 0.222  0.358 4.59e- 2
## 17 air    National   5       5.25 -2.50e- 1      -0.802 0.222  0.358 4.59e- 2
## 18 air    Generic    4.5     4.5  -8.88e-16       0     0.222  0.367 4.33e-32
```

I like the idea of using the residuals mainly because there are more of them than observations in each group, so you have more to go on in assessing normality (and the tools are familiar ones from regression). Looking at the normal quantile plot gets directly at whether the residuals, and hence the data within each group, are normal enough. Unequal spreads will show up as some of the vertical columns of points going further up and down than others (not happening here), or even as fanning-out, which would have the same fix (transformation) as in regression.

You could, if you wanted to, go on from here and plot residuals against `brand` and then `popper`, as boxplots because the two explanatory variables are both discrete. You would be looking for residuals that average out to zero within each group and have about the same spread, with a symmetric shape.

2. A number of people with depression were randomly assigned to receive one of four different medicines to treat the depression. These were: none (no medicine), a placebo (that looked like a real medicine), a homeopathic medicine, and a pharmaceutical (an actual drug, the kind of thing a doctor would prescribe for you). At the end of the four-week study, each person completed the Beck Depression Inventory (BDI), which is a standard questionnaire designed to assess how depressed a person is. A higher score on this questionnaire indicates greater depression, and so a lower score is better. The data are in http://ritsokiguess.site/STAD29/depression2.csv. Our aim is to see how the `bdi` depression score depends on `medicine` and `gender` (the gender each person identified as), and to recommend a medicine or medicines, possibly different for each gender.

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

Solution:

Exactly as usual:

```
my_url <- "http://ritsokiguess.site/STAD29/depression2.csv"
depression <- read_csv(my_url)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_double(),
##   gender = col_character(),
##   bdate = col_date(format = ""),
##   bdi = col_double(),
##   bdigroup = col_double(),
##   medicine = col_character()
## )
```

```
depression
```

```
## # A tibble: 100 x 6
##          id gender bdate        bdi bdigroup medicine
##       <dbl> <chr>  <date>     <dbl>    <dbl> <chr>
##   1 2016003 female 1959-01-01    34        4 placebo
##   2 2016005 male   1951-01-01    35        4 homeopathic
##   3 2016007 female 1952-01-01    19        3 pharmaceutical
##   4 2016009 male   1962-01-01    38        4 none
##  5 2016011 female 1981-01-01    38        4 placebo
##  6 2016013 female 1967-01-01    43        4 homeopathic
##  7 2016015 female 1949-01-01    20        3 pharmaceutical
##  8 2016017 female 1991-01-01    51        4 none
##  9 2016019 male   1955-01-01    33        4 placebo
## 10 2016021 male   1967-01-01    37        4 homeopathic
## # ... with 90 more rows
```

There are 100 people altogether.

Extra: there is a data story here, of course. The data came from an example using SPSS, and the standard way of saving an SPSS dataset is in a `.sav` file. R can read these directly. The easiest way I know is via a package called `rio`. This is very easy to use: you pass the filename into `import`, it figures out what it is, and reads it in. `rio` is thus very complicated behind the scenes; it is a front end to a whole bunch of other packages that do the actual work of reading in data files of different types, so that when you install it, you also install a bunch of other things at the same time. Here's how it goes:

```r
library(rio)
depression0 <- import("~/Downloads/depression.sav")
head(depression0, 10)
```

```
##         id gender      bdate bdi bdigroup medicine
## 1  2016003      1 1959-01-01  34        4        2
## 2  2016005      0 1951-01-01  35        4        3
## 3  2016007      1 1952-01-01  19        3        4
## 4  2016009      0 1962-01-01  38        4        1
## 5  2016011      1 1981-01-01  38        4        2
## 6  2016013      1 1967-01-01  43        4        3
## 7  2016015      1 1949-01-01  20        3        4
## 8  2016017      1 1991-01-01  51        4        1
## 9  2016019      0 1955-01-01  33        4        2
## 10 2016021      0 1967-01-01  37        4        3
```

I am using a temporary dataframe name, since I am expecting to have to do some more work yet. Something else to note: `import` makes a `data.frame` rather than a `tibble` as we are used to, so that if you are not careful, you'll see all of it rather than the first ten lines. If that happens to you (as it probably will if you are getting PDF output), you'll need to use `head()` or something similar to display a smaller number of rows.

SPSS allows categorical variables to have a "label" which is what displays when you display the dataset, but when they make it to R, you don't see them, instead seeing numeric codes. They are there, though, just rather hidden. Here's the "structure" of our dataframe:

```r
str(depression0)
```

```
## 'data.frame':    100 obs. of  6 variables:
##  $ id      : num  2016003 2016005 2016007 2016009 2016011 ...
##   ..- attr(*, "format.spss")= chr "F1.0"
##   ..- attr(*, "display_width")= int 10
##  $ gender  : num  1 0 1 0 1 1 1 1 0 0 ...
##   ..- attr(*, "format.spss")= chr "F1.0"
##   ..- attr(*, "display_width")= int 10
##   ..- attr(*, "labels")= Named num [1:2] 0 1
##   .. ..- attr(*, "names")= chr [1:2] "Male" "Female"
##  $ bdate   : Date, format: "1959-01-01" "1951-01-01" ...
##  $ bdi     : num  34 35 19 38 38 43 20 51 33 37 ...
##   ..- attr(*, "label")= chr "Score on Beck's Depression Inventory"
##   ..- attr(*, "format.spss")= chr "F1.0"
##   ..- attr(*, "display_width")= int 10
##  $ bdigroup: num  4 4 3 4 4 4 3 4 4 4 ...
##   ..- attr(*, "label")= chr "Diagnosis based on Beck's Depression Inventory score"
##   ..- attr(*, "format.spss")= chr "F1.0"
##   ..- attr(*, "display_width")= int 10
##   ..- attr(*, "labels")= Named num [1:4] 1 2 3 4
##   .. ..- attr(*, "names")= chr [1:4] "Minimal depression" "Mild depression" "Moderate depre
##  $ medicine: num  2 3 4 1 2 3 4 1 2 3 ...
##   ..- attr(*, "label")= chr "Medicine administered"
##   ..- attr(*, "format.spss")= chr "F1.0"
##   ..- attr(*, "display_width")= int 10
##   ..- attr(*, "labels")= Named num [1:4] 1 2 3 4
##   .. ..- attr(*, "names")= chr [1:4] "None" "Placebo" "Homeopathic" "Pharmaceutical"
##  - attr(*, "notes")= chr [1:5] "document These data are the property of www.spss-
tutorials.com." "   (Entered 02-Jun-2016)" "document May be freely used and shared as long as
```

OK, so I guess I should say that these data came from here, which is a bit more of a hint than I wanted to give you while you were still doing this one.

It is possible to extract the labels from here, like this:

```
attr(depression0$gender, "labels")
```

```
##   Male Female
##      0      1
```

This is a "named vector": the numbers at the bottom are the values and the words above them are their names. To make this easier to work with, let's put it into a dataframe, which is what `enframe` does:

```
attr(depression0$gender, "labels") %>%
  enframe() -> genders
genders
```

```
## # A tibble: 2 x 2
##   name   value
##   <chr>  <dbl>
## 1 Male       0
## 2 Female     1
```

This is going to be convenient for looking things up later. Now, we can do `medicine` the same way:

```
attr(depression0$medicine, "labels") %>%
  enframe() -> medicines
medicines
```

```
## # A tibble: 4 x 2
##   name           value
##   <chr>          <dbl>
## 1 None               1
## 2 Placebo            2
## 3 Homeopathic        3
## 4 Pharmaceutical     4
```

I am using plural names for these, to avoid confusing them with columns of our dataframe.

Now to look these up in our original dataframe, which is going to use `left_join` (twice, eventually). Let's do `gender` first and make sure our procedure is right:

```
depression0 %>% left_join(genders, by = c("gender"="value")) %>% head(10)
```

```
##          id gender      bdate bdi bdigroup medicine   name
## 1  2016003      1 1959-01-01  34        4        2 Female
## 2  2016005      0 1951-01-01  35        4        3   Male
## 3  2016007      1 1952-01-01  19        3        4 Female
## 4  2016009      0 1962-01-01  38        4        1   Male
## 5  2016011      1 1981-01-01  38        4        2 Female
## 6  2016013      1 1967-01-01  43        4        3 Female
## 7  2016015      1 1949-01-01  20        3        4 Female
## 8  2016017      1 1991-01-01  51        4        1 Female
## 9  2016019      0 1955-01-01  33        4        2   Male
## 10 2016021      0 1967-01-01  37        4        3   Male
```

That seems to have worked: every 1 in the `gender` column goes with a `Female` in the column which is now called `name` (which we'll fix up later). Note that I had to specify the names of the columns in the original dataframe and the lookup table that have to match, since their names are different. The `by` means that the column called `gender` in the first dataframe (`depression0`) has to match the column called `value` in the second one (`genders`).

So let's do `medicine` now, the same way:

```
depression0 %>%
  left_join(genders, by = c("gender"="value")) %>%
  left_join(medicines, by = c("medicine"="value")) %>% head(10)
```

```
##          id gender      bdate bdi bdigroup medicine name.x         name.y
## 1  2016003      1 1959-01-01  34        4        2 Female        Placebo
## 2  2016005      0 1951-01-01  35        4        3   Male    Homeopathic
## 3  2016007      1 1952-01-01  19        3        4 Female Pharmaceutical
## 4  2016009      0 1962-01-01  38        4        1   Male           None
## 5  2016011      1 1981-01-01  38        4        2 Female        Placebo
## 6  2016013      1 1967-01-01  43        4        3 Female    Homeopathic
## 7  2016015      1 1949-01-01  20        3        4 Female Pharmaceutical
## 8  2016017      1 1991-01-01  51        4        1 Female           None
## 9  2016019      0 1955-01-01  33        4        2   Male        Placebo
## 10 2016021      0 1967-01-01  37        4        3   Male    Homeopathic
```

That's as far as I need to go for this question, so let's grab the columns I want and give them good names, all of which I can do with `select`:

```
depression0 %>%
  left_join(genders, by = c("gender"="value")) %>%
  left_join(medicines, by = c("medicine"="value")) %>%
  select(id, gender=name.x, bdate, bdi, bdigroup, medicine=name.y) %>% head(20)
```

```
##          id gender      bdate bdi bdigroup       medicine
## 1  2016003 Female 1959-01-01  34         4       Placebo
## 2  2016005   Male 1951-01-01  35         4    Homeopathic
## 3  2016007 Female 1952-01-01  19         3 Pharmaceutical
## 4  2016009   Male 1962-01-01  38         4           None
## 5  2016011 Female 1981-01-01  38         4       Placebo
## 6  2016013 Female 1967-01-01  43         4    Homeopathic
## 7  2016015 Female 1949-01-01  20         3 Pharmaceutical
## 8  2016017 Female 1991-01-01  51         4           None
## 9  2016019   Male 1955-01-01  33         4       Placebo
## 10 2016021   Male 1967-01-01  37         4    Homeopathic
## 11 2016023 Female 1953-01-01  21         3 Pharmaceutical
## 12 2016024   Male 1986-01-01  45         4           None
## 13 2016025   Male 1960-01-01  35         4       Placebo
## 14 2016026 Female 1959-01-01  35         4    Homeopathic
## 15 2016028   Male 1964-01-01  30         4 Pharmaceutical
## 16 2016030 Female 1954-01-01  42         4           None
## 17 2016031 Female 1955-01-01  31         4       Placebo
## 18 2016032   Male 1955-01-01  19         3    Homeopathic
## 19 2016034   Male 1957-01-01  27         3 Pharmaceutical
## 20 2016038   Male 1961-01-01  32         4           None
```

which is (20 rows of) the dataframe I saved for you. If you wanted, you could handle `bdigroup` the same way; this is a category of depression based on the BDI score.

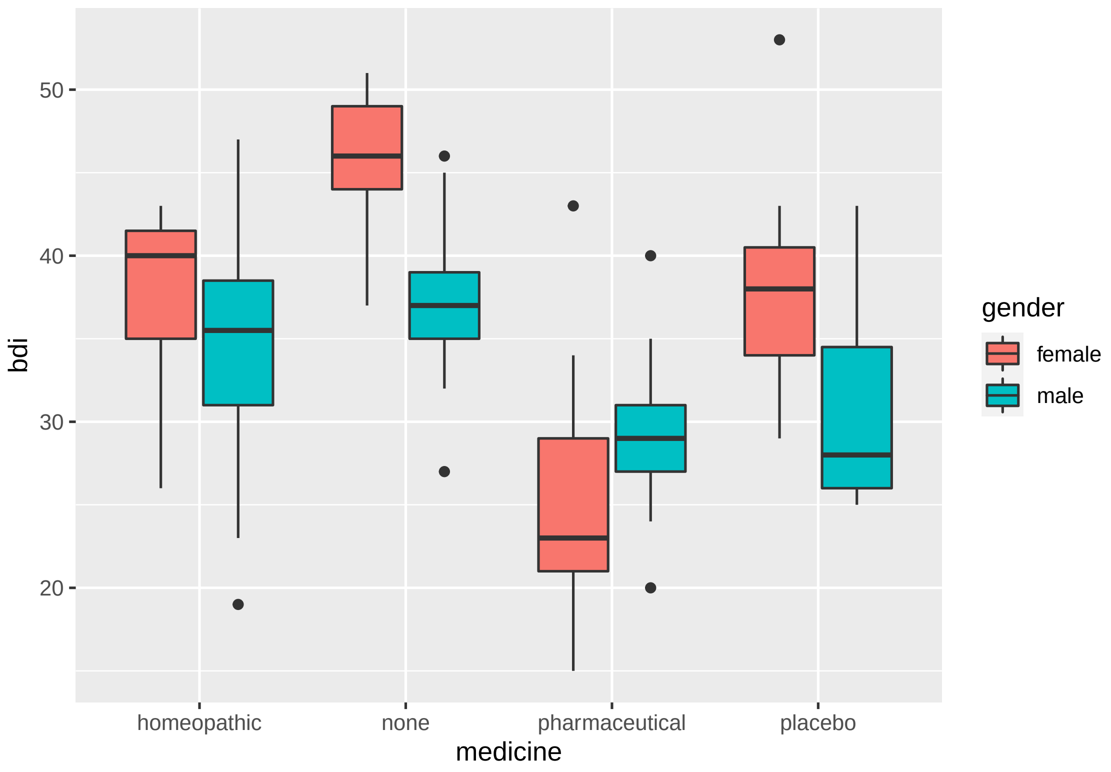
I discovered I already had a file called `depression.csv`, of different data, so this one had to be called `depression2`.

(b) Make a suitable plot, using only the variables that are of interest to us.

Solution:

We care about the `bdi` score (quantitative), `gender` (categorical) and `medicine` (also categorical), so a grouped boxplot is the thing. We have to make a choice about which categorical variable is `x` and which is `fill` (or `colour`). My usual procedure is to have the one with more categories (here `medicine`) be `x`. Another indication is that we will later want to compare medicines by gender, which is easier to do if the latter is the "traces" (try it the other way around and see what you think):

```r
ggplot(depression, aes(x = medicine, y = bdi, fill = gender)) + geom_boxplot()
```



There are a lot of outliers, which we are going to ignore for this question, because I wanted you to get the mechanics of a standard two-way anova. The suggestion from here, though, is that the best results come from the pharmaceutical for the females, and from either that or the placebo for the males. (The fact that these two are similar for the males and very different for the females suggests that there may be a significant interaction coming up.)

The choice between `colour` and `fill` is entirely yours. `fill` colours in the whole box, while `colour` just colours around the outside.

(c) Make an interaction plot for these data. What does it tell you? Explain briefly.

Solution:

The easiest way is to make a summary table of means first, giving your summary a meaningful name because it is going on the y-axis of your plot:

```
depression %>%
  group_by(medicine, gender) %>%
  summarize(mean_bdi = mean(bdi)) -> d
```

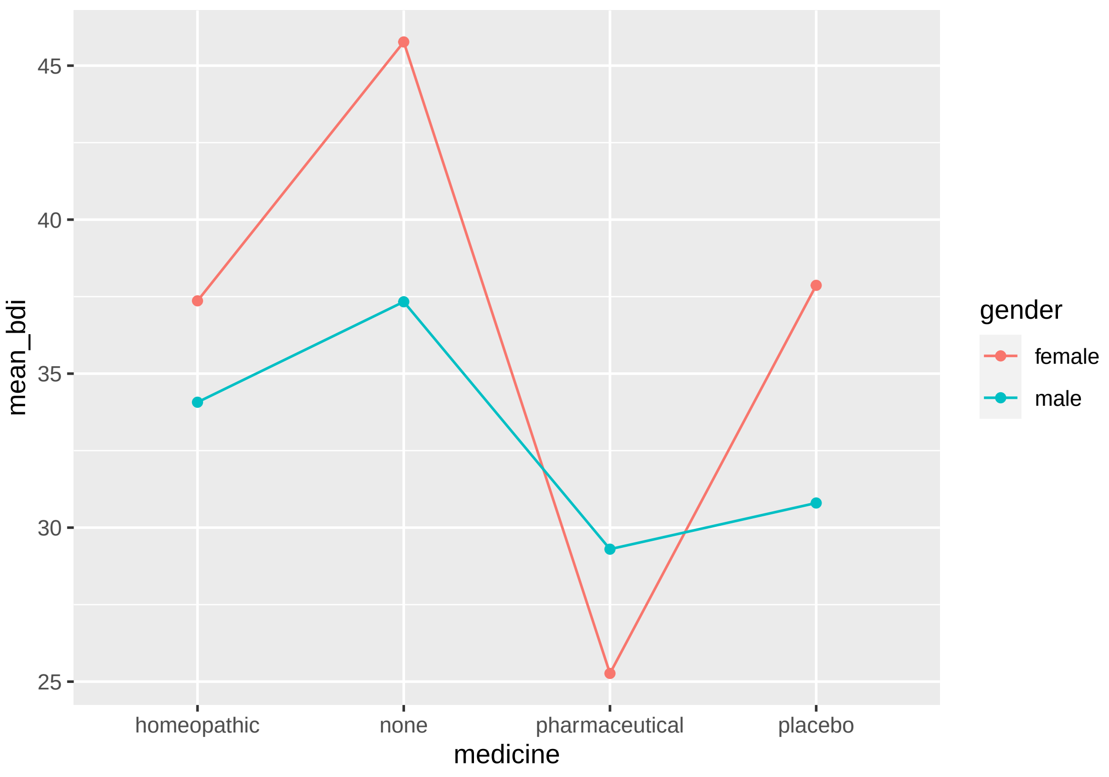
```
## `summarise()` has grouped output by 'medicine'. You can override using the `.groups` argume
```

```
d
```

```
## # A tibble: 8 x 3
## # Groups:   medicine [4]
##   medicine       gender mean_bdi
##   <chr>          <chr>     <dbl>
## 1 homeopathic    female     37.4
## 2 homeopathic    male       34.1
## 3 none           female     45.8
## 4 none           male       37.3
## 5 pharmaceutical female     25.3
## 6 pharmaceutical male       29.3
## 7 placebo        female     37.9
## 8 placebo        male       30.8
```

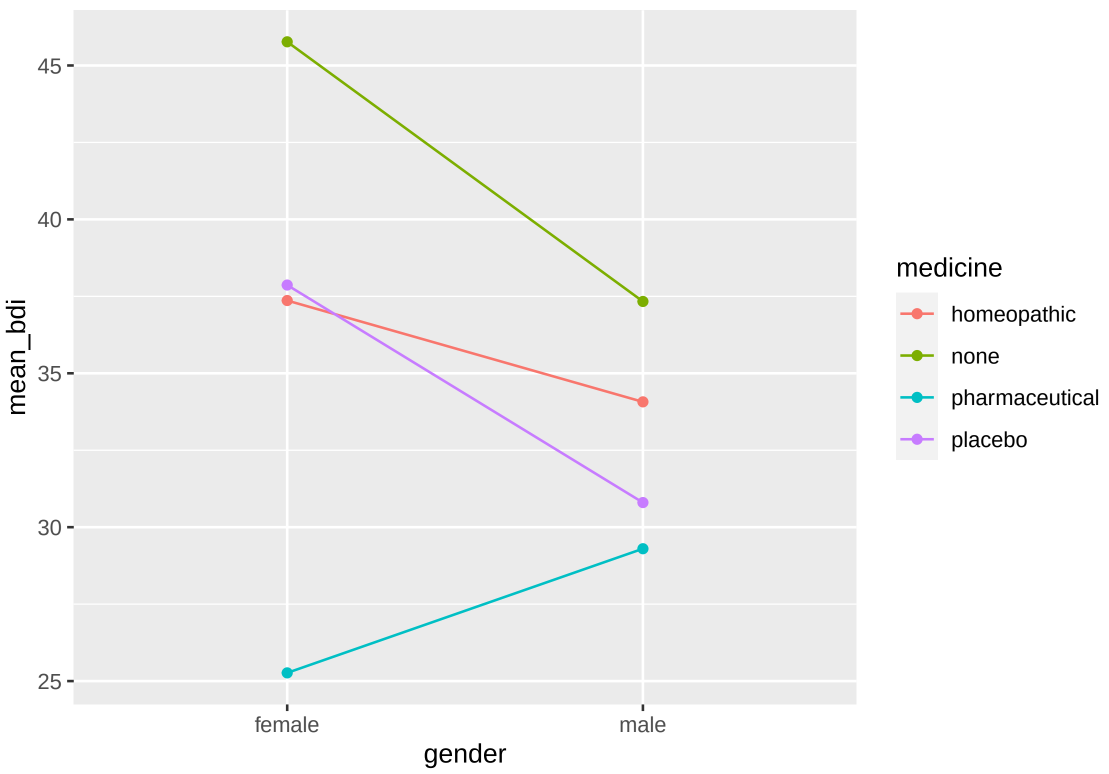
and then use that to make your plot, either with medicine going across (which I prefer, for the same reasons as on the boxplot):

```
ggplot(d, aes(x = medicine, y = mean_bdi, colour = gender, group = gender)) + geom_point() + g
```



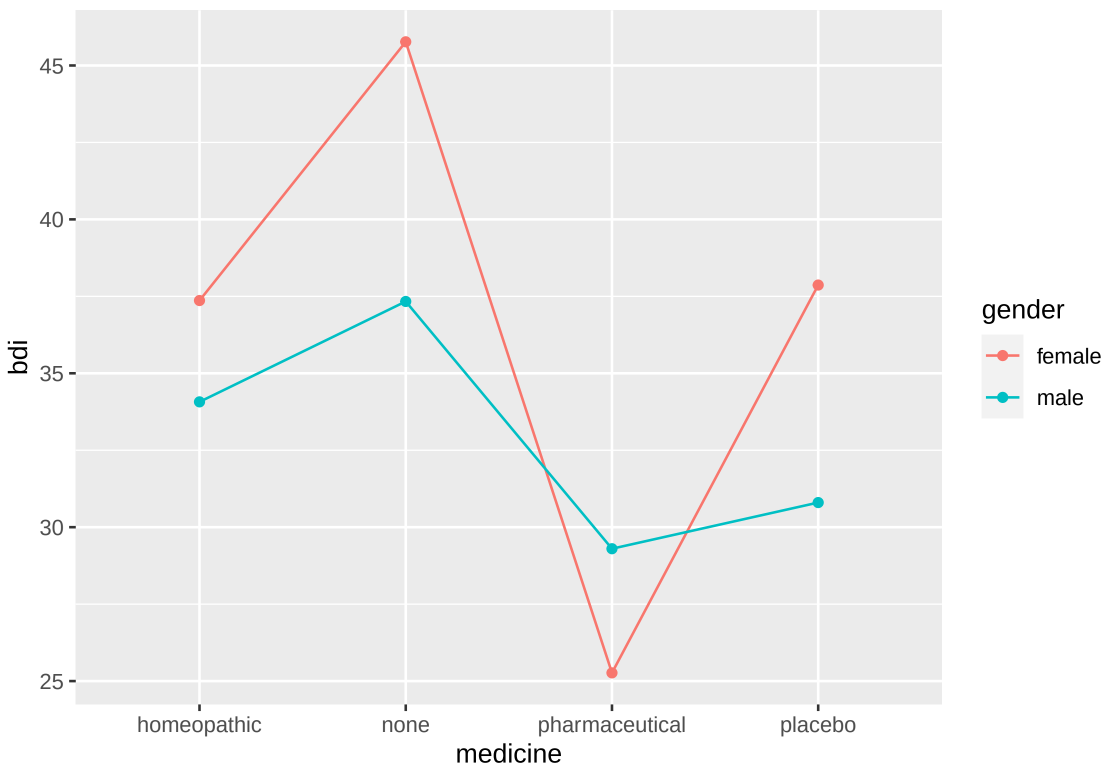
or with **gender** going across:

```
ggplot(d, aes(x = gender, y = mean_bdi, colour = medicine, group = medicine)) + geom_point() +
```



or, do it in one shot with `stat_summary`:

```
ggplot(depression, aes(x=medicine, y = bdi, colour=gender, group=gender)) +
  stat_summary(fun="mean", geom="point") +
  stat_summary(fun="mean", geom="line")
```



The disadvantage of this, for me, is that I can remember the other one, but this one I have to find an example of to copy.

Whichever of these you come up with (or the fourth variant with the four medicine traces and `stat_summary`), I think you'd have to say that these lines are *not* parallel and that therefore there ought to be an interaction: females react much better to the pharmaceutical and much worse to everything else than males do, at least on average.

(The caveat would be that nothing on this plot says anything about normality or spreads or outliers. In particular, there is a disturbing number of outliers on the boxplot, which in the grand scheme of things ought to concern us. But I wanted to give you a chance to practice one with interaction, so we won't worry about that until later.)

(d) Run a suitable two-way analysis of variance. What do you conclude from it? (Don't run any followups yet, because we have to look at the output and decide what a suitable followup is.)

Solution:

An `aov` with interaction:

```
depression.1 <- aov(bdi ~ medicine * gender, data = depression)
summary(depression.1)
```

```
##                 Df Sum Sq Mean Sq F value   Pr(>F)    
## medicine         3   2780   926.6  25.199 5.47e-12 ***
## gender           1    340   340.3   9.254  0.00306 ** 
## medicine:gender  3    568   189.3   5.147  0.00246 ** 
## Residuals       92   3383    36.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction is significant, meaning that the effect of medicine is different for males and females.

At this point you *stop*. Even though both main effects are significant, it makes no sense to talk about "a medicine effect" (regardless of gender), or about "a gender effect" (regardless of medicine), because the significant interaction means that the medicine effects are different for males and females, and so we should look at the two genders separately. (This might suggest right away that simple effects of medicine for each gender would be the thing.)

(e) The researchers wanted to make a recommendation for best medicine(s) for each of males and females separately. Would it be appropriate to use simple effects for this? Explain briefly.

Solution:

The place to use simple effects is when we have a significant interaction, as we do here. The researchers are hoping to do a comparison of medicines for "each level of the other factor", that is, for each gender. So simple effects are not only appropriate, but they will do exactly what we are looking for.

(f) Run a simple effects analysis for each gender, including Tukey if appropriate. What are your recommendations for medicines?

Solution:

Because there are four medicines to compare each time, this one goes a lot more smoothly using the simple `filter`ing strategy: make a dataframe containing just the females, analyze, then make another containing just the males and analyze that.

First the females, then:

```
depression %>%
  filter(gender == "female") -> females
females
```

```
## # A tibble: 54 x 6
##          id gender bdate        bdi bdigroup medicine
##       <dbl> <chr>  <date>     <dbl>    <dbl> <chr>
##  1 2016003 female 1959-01-01    34        4 placebo
##  2 2016007 female 1952-01-01    19        3 pharmaceutical
##  3 2016011 female 1981-01-01    38        4 placebo
##  4 2016013 female 1967-01-01    43        4 homeopathic
##  5 2016015 female 1949-01-01    20        3 pharmaceutical
##  6 2016017 female 1991-01-01    51        4 none
##  7 2016023 female 1953-01-01    21        3 pharmaceutical
##  8 2016026 female 1959-01-01    35        4 homeopathic
##  9 2016030 female 1954-01-01    42        4 none
## 10 2016031 female 1955-01-01    31        4 placebo
## # ... with 44 more rows
```

A one-way analysis of `bdi` by `medicine`:

```
females.1 <- aov(bdi ~ medicine, data = females)
summary(females.1)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)    
## medicine     3   3040  1013.3   28.73 6.08e-11 ***
## Residuals   50   1764    35.3                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is definitely a difference among the medicines for the females. Which ones differ from which?

```
TukeyHSD(females.1)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
## 
## Fit: aov(formula = bdi ~ medicine, data = females)
## 
## $medicine
##                                   diff        lwr        upr     p adj
## none-homeopathic             8.4055944   1.939677  14.871512 0.0060343
## pharmaceutical-homeopathic -12.0969697 -18.362203  -5.831736 0.0000274
## placebo-homeopathic          0.5030303  -5.762203   6.768264 0.9965220
## pharmaceutical-none        -20.5025641 -26.483295 -14.521833 0.0000000
## placebo-none                -7.9025641 -13.883295  -1.921833 0.0051180
## placebo-pharmaceutical      12.6000000   6.836823  18.363177 0.0000025
```

They all differ, except for placebo and homeopathic. If you like, work out from here that `pharmaceutical` is lower on `bdi` than all the others for the females, or go back and look at the interaction plot, or work out which mean was lowest:

```
females %>% group_by(medicine) %>%
  summarize(mean_bdi = mean(bdi)) %>%
  arrange(mean_bdi)
```

```
## # A tibble: 4 x 2
##   medicine       mean_bdi
##   <chr>             <dbl>
## 1 pharmaceutical     25.3
## 2 homeopathic        37.4
## 3 placebo            37.9
## 4 none               45.8
```

The pharmaceutical was significantly lower than all the others, so that is the one to recommend for females.

Now we repeat the whole thing for the males. You can do a lot of copying and pasting here:

```
depression %>%
  filter(gender == "male") -> males
males
```

```
## # A tibble: 46 x 6
##          id gender bdate        bdi bdigroup medicine
##      <dbl> <chr>  <date>     <dbl>    <dbl> <chr>
##  1 2016005 male   1951-01-01    35        4 homeopathic
##  2 2016009 male   1962-01-01    38        4 none
##  3 2016019 male   1955-01-01    33        4 placebo
##  4 2016021 male   1967-01-01    37        4 homeopathic
##  5 2016024 male   1986-01-01    45        4 none
##  6 2016025 male   1960-01-01    35        4 placebo
##  7 2016028 male   1964-01-01    30        4 pharmaceutical
##  8 2016032 male   1955-01-01    19        3 homeopathic
##  9 2016034 male   1957-01-01    27        3 pharmaceutical
## 10 2016038 male   1961-01-01    32        4 none
## # ... with 36 more rows
```

```
males.1 <- aov(bdi ~ medicine, data = males)
summary(males.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## medicine     3  425.3  141.77   3.677 0.0194 *
## Residuals   42 1619.3   38.55
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Once again, significant differences among the medicines, though not as strongly as for the females.

```
TukeyHSD(males.1)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = bdi ~ medicine, data = males)
##
## $medicine
##                                 diff        lwr        upr     p adj
## none-homeopathic            3.261905  -3.272217  9.7960269 0.5461322
## pharmaceutical-homeopathic -4.771429 -11.648394  2.1055365 0.2623911
## placebo-homeopathic        -3.271429 -10.148394  3.6055365 0.5851828
## pharmaceutical-none        -8.033333 -15.145074 -0.9215923 0.0213706
## placebo-none               -6.533333 -13.645074  0.5784077 0.0818172
## placebo-pharmaceutical      1.500000  -5.927971  8.9279713 0.9486628
```

At $\alpha = 0.05$, there is only *one* significant difference, between pharmaceutical and none, and:

```
males %>% group_by(medicine) %>%
  summarize(mean_bdi = mean(bdi)) %>%
  arrange(mean_bdi)
```

```
## # A tibble: 4 x 2
##   medicine       mean_bdi
##   <chr>             <dbl>
## 1 pharmaceutical     29.3
## 2 placebo            30.8
## 3 homeopathic        34.1
## 4 none               37.3
```

none of the top three there are significantly different, so for males you would be justified in recommending anything except `none`: that is, statistically, you cannot choose between `pharmaceutical`, `placebo`, and `homeopathic`.

This is a different result than for the females, and this ought not to be a surprise because of the significant interaction: we would *expect* the results to be different for males and females.

Extra 1: there is an issue of what to make of the result for the males. One reaction is that we need to get more data, and that will make it easier to get statistically significant results. On the other hand, the difference between `pharmaceutical` and `placebo` is only 1.5 points on this scale, and that might not be a medically meaningful difference. If that is the case, getting more data won't help much; for males, it really may be true that it doesn't make any medically meaningful difference whether you give them a real drug or a placebo when treating depression.

Extra 2: I suppose I ought to try to get these results all at once. It is a bit more involved than the other times we've done it, though.

We start by making separate mini-dataframes for each of the males and females:

```
depression %>%
  group_by(gender) %>%
  nest()
```

```
## # A tibble: 2 x 2
## # Groups:   gender [2]
##   gender data
##   <chr>  <list>
## 1 female <tibble [54 x 5]>
## 2 male   <tibble [46 x 5]>
```

The column `data` is a list-column; it contains the two data frames that we called `females` and `males` above. For each of those we can run an ANOVA predicting `bdi` from `medicine`:

```
depression %>%
  group_by(gender) %>%
  nest() %>%
  mutate(aovs = map(data, ~aov(bdi ~ medicine, data = .)))
```

```
## # A tibble: 2 x 3
## # Groups:   gender [2]
##   gender data              aovs
##   <chr>  <list>            <list>
## 1 female <tibble [54 x 5]> <aov>
## 2 male   <tibble [46 x 5]> <aov>
```

The column called `aovs` contains all the output from the analyses of variance. For the moment, though, we just want the P-values: that is, tidy it and pull out the thing called `p.value`:

```
depression %>%
  group_by(gender) %>%
  nest() %>%
  mutate(aovs = map(data, ~aov(bdi ~ medicine, data = .))) %>%
  mutate(aov_tidy = map(aovs, ~tidy(.))) %>%
  mutate(pval = map(aov_tidy, "p.value")) %>%
  unnest(pval)
```

```
## # A tibble: 4 x 5
## # Groups:   gender [2]
##   gender data              aovs   aov_tidy              pval
##   <chr>  <list>            <list> <list>               <dbl>
## 1 female <tibble [54 x 5]> <aov>  <tibble [2 x 6]>  6.08e-11
## 2 female <tibble [54 x 5]> <aov>  <tibble [2 x 6]> NA
## 3 male   <tibble [46 x 5]> <aov>  <tibble [2 x 6]>  1.94e- 2
## 4 male   <tibble [46 x 5]> <aov>  <tibble [2 x 6]> NA
```

There are actually two P-values in the tidy output from `aov` and the second one is missing, but we finally got there. The female P-value is tiny, and the male one is about 0.0194, which matches with what we had before.

To get the Tukeys, let's back up a bit, since we only need `aovs` so far:

```
depression %>%
  group_by(gender) %>%
  nest() %>%
  mutate(aovs = map(data, ~aov(bdi ~ medicine, data = .))) %>%
  mutate(tukeys = map(aovs, ~TukeyHSD(.)))
```

```
## # A tibble: 2 x 4
## # Groups:   gender [2]
##   gender data              aovs   tukeys
##   <chr>  <list>            <list> <list>
## 1 female <tibble [54 x 5]> <aov>  <TukeyHSD>
## 2 male   <tibble [46 x 5]> <aov>  <TukeyHSD>
```

The bit of each Tukey we need is the piece of it called `medicine`:

```
depression %>%
  group_by(gender) %>%
  nest() %>%
  mutate(aovs = map(data, ~aov(bdi ~ medicine, data = .))) %>%
  mutate(tukeys = map(aovs, ~TukeyHSD(.))) %>%
  mutate(med = map(tukeys, "medicine"))
```

```
## # A tibble: 2 x 5
## # Groups:   gender [2]
##   gender data              aovs   tukeys     med
##   <chr>  <list>            <list> <list>     <list>
## 1 female <tibble [54 x 5]> <aov>  <TukeyHSD> <dbl[,4] [6 x 4]>
## 2 male   <tibble [46 x 5]> <aov>  <TukeyHSD> <dbl[,4] [6 x 4]>
```

The Tukey output each time has 6 rows (6 comparisons between 4 medicines) and 4 columns of numbers, so this is encouraging. However, we have a problem in that the output from `TukeyHSD` is a `matrix` rather than a dataframe, and it's not clear how to bash this into shape. Also, the matrix actually has *five* columns; the one saying which groups are being compared is a not-really-column called "row names", which we will also have to handle.

```
depression %>%
  group_by(gender) %>%
  nest() %>%
  mutate(aovs = map(data, ~aov(bdi ~ medicine, data = .))) %>%
  mutate(tukeys = map(aovs, ~TukeyHSD(.))) %>%
  mutate(med = map(tukeys, "medicine")) %>%
  mutate(med2 = map(med, ~as_tibble(., rownames = "comp")))
```

```
## # A tibble: 2 x 6
## # Groups:   gender [2]
##   gender data              aovs   tukeys     med                med2
##   <chr>  <list>            <list> <list>     <list>             <list>
## 1 female <tibble [54 x 5]> <aov>  <TukeyHSD> <dbl[,4] [6 x 4]> <tibble [6 x 5]>
## 2 male   <tibble [46 x 5]> <aov>  <TukeyHSD> <dbl[,4] [6 x 4]> <tibble [6 x 5]>
```

The function `as_tibble` takes something that looks like a dataframe but isn't (our matrix here), and turns it into a dataframe. The `rownames` input to `as_tibble` says to keep the row names and put them in a column called `comp` (for "comparison"; you could call the column whatever you like.). From here, we take a look at `med2` by `unnest`ing it, and get rid of the other columns that have served their purpose:

```
depression %>%
  group_by(gender) %>%
  nest() %>%
  mutate(aovs = map(data, ~aov(bdi ~ medicine, data = .))) %>%
  mutate(tukeys = map(aovs, ~TukeyHSD(.))) %>%
  mutate(med = map(tukeys, "medicine")) %>%
  mutate(med2 = map(med, ~as_tibble(., rownames = "comp"))) %>%
  unnest(med2) %>%
  select(-data, -aovs, -tukeys, -med)
```

```
## # A tibble: 12 x 6
## # Groups:   gender [2]
##    gender comp                           diff    lwr     upr   `p adj`
##    <chr>  <chr>                         <dbl>  <dbl>   <dbl>     <dbl>
##  1 female none-homeopathic              8.41    1.94  14.9    6.03e- 3
##  2 female pharmaceutical-homeopathic -12.1    -18.4   -5.83   2.74e- 5
##  3 female placebo-homeopathic           0.503  -5.76   6.77   9.97e- 1
##  4 female pharmaceutical-none        -20.5    -26.5  -14.5    1.56e-11
##  5 female placebo-none                 -7.90  -13.9   -1.92   5.12e- 3
##  6 female placebo-pharmaceutical      12.6      6.84  18.4    2.54e- 6
##  7 male   none-homeopathic              3.26   -3.27   9.80   5.46e- 1
##  8 male   pharmaceutical-homeopathic  -4.77  -11.6    2.11   2.62e- 1
##  9 male   placebo-homeopathic          -3.27  -10.1    3.61   5.85e- 1
## 10 male   pharmaceutical-none          -8.03  -15.1   -0.922 2.14e- 2
## 11 male   placebo-none                 -6.53  -13.6    0.578 8.18e- 2
## 12 male   placebo-pharmaceutical        1.5    -5.93   8.93   9.49e- 1
```

Finally, from here we can see which comparisons are significant for each of males and females, and see which medicine(s) are best for each of them. The results are the same as before.

You are entirely justified in wondering whether this was really worth all the trouble. (The business about converting the matrices to dataframes was something *I* had to look up.) If the filter way works for you, I say go with that.

Extra 3: I didn't ask you to assess the assumptions, normality, equal spreads and such. This goes the same way as in the other question where the interaction was not significant: run the model as a regression, and look at the residuals from that:

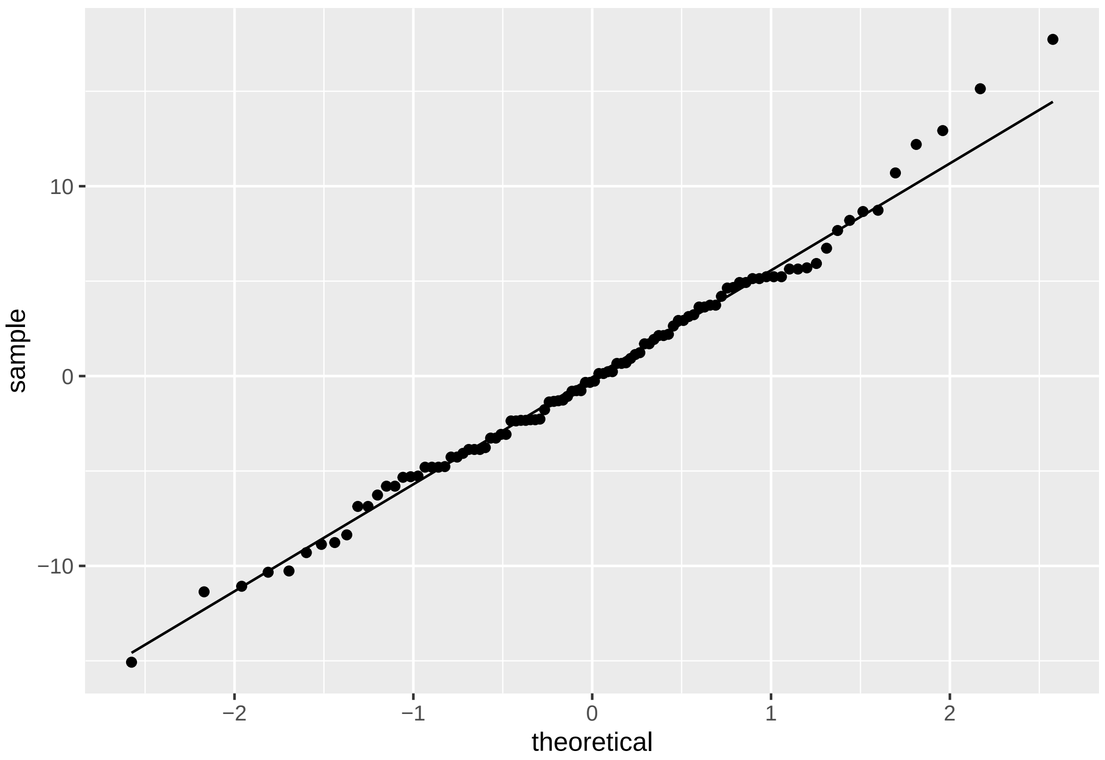
```
depression.2 <- lm(bdi ~ medicine * gender, data = depression)
summary(depression.2)
```

```
##
## Call:
## lm(formula = bdi ~ medicine * gender, data = depression)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.071  -3.867  -0.300   3.733  17.733
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          37.364      1.828  20.436  < 2e-16 ***
## medicinenone                          8.406      2.484   3.384  0.00105 **
## medicinepharmaceutical              -12.097      2.407  -5.026 2.46e-06 ***
## medicineplacebo                       0.503      2.407   0.209  0.83493
## gendermale                           -3.292      2.443  -1.348  0.18112
## medicinenone:gendermale              -5.144      3.444  -1.493  0.13873
## medicinepharmaceutical:gendermale     7.325      3.478   2.106  0.03791 *
## medicineplacebo:gendermale           -3.775      3.478  -1.085  0.28067
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.064 on 92 degrees of freedom
## Multiple R-squared:  0.5216, Adjusted R-squared:  0.4852
## F-statistic: 14.33 on 7 and 92 DF,  p-value: 1.829e-12
```

This is rather hard to interpret because we now have a significant interaction; the last three lines tell you how the effect of each medicine is different for males compared to the baseline females. For example, males score lower than females on None and Placebo, but higher on Pharmaceutical. (This is strictly speaking compared to the male-female difference on the baseline medicine Homeopathic. See, I said this was hard to interpret.)

I would rather not struggle with that, though, preferring to make some graphs, such as a normal quantile plot of the residuals:

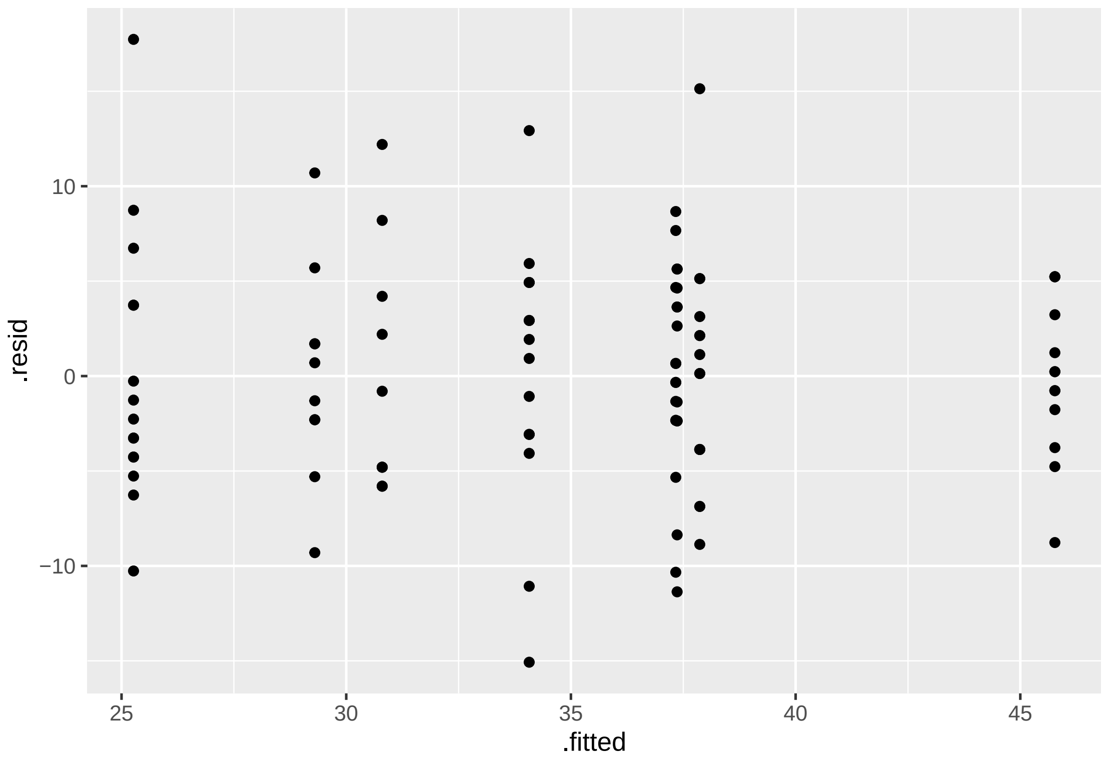
```
ggplot(depression.2, aes(sample = .resid)) + stat_qq() + stat_qq_line()
```



Despite the outliers we saw before, there is nothing wrong with the normality here.

Residuals against fitted values:

```
ggplot(depression.2, aes(x = .fitted, y = .resid)) + geom_point()
```



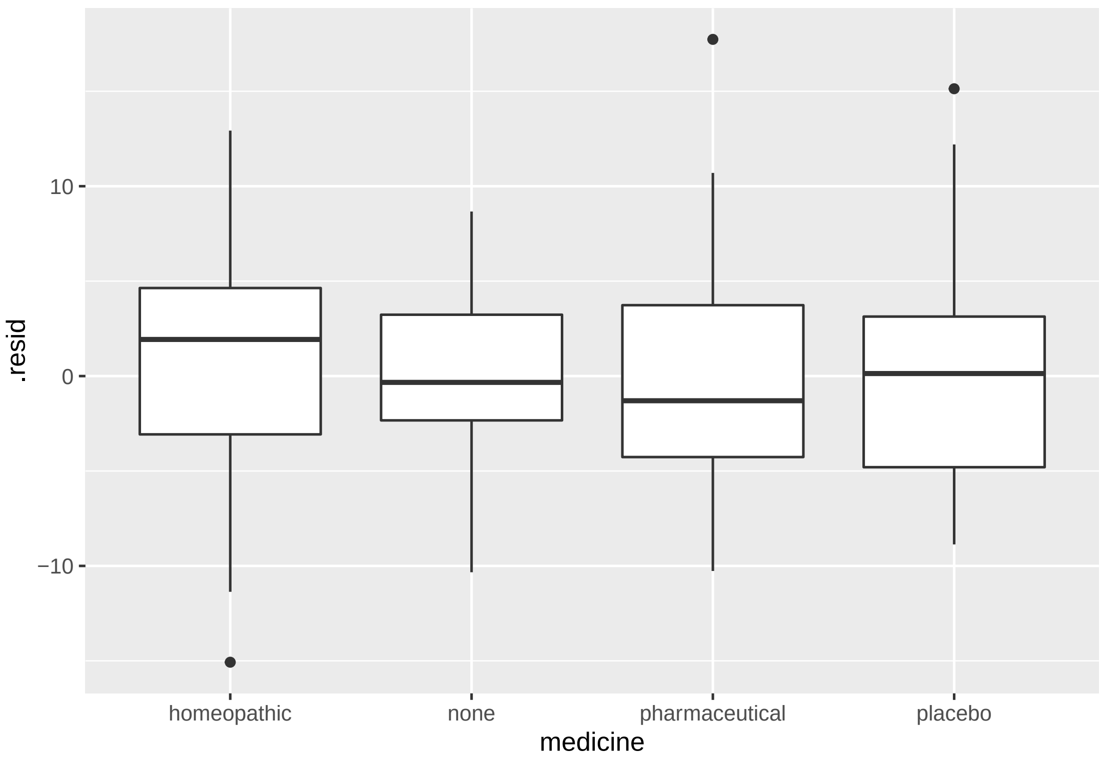
This seems pretty random, except perhaps for the residuals on the right with the largest fitted values, which seem less variable than the others.

Also, residuals against medicines and gender, which, you'll recall, have to be boxplots because the explanatory variables are categorical:

```
depression.2 %>% augment(depression) -> depression_aug
depression_aug
```

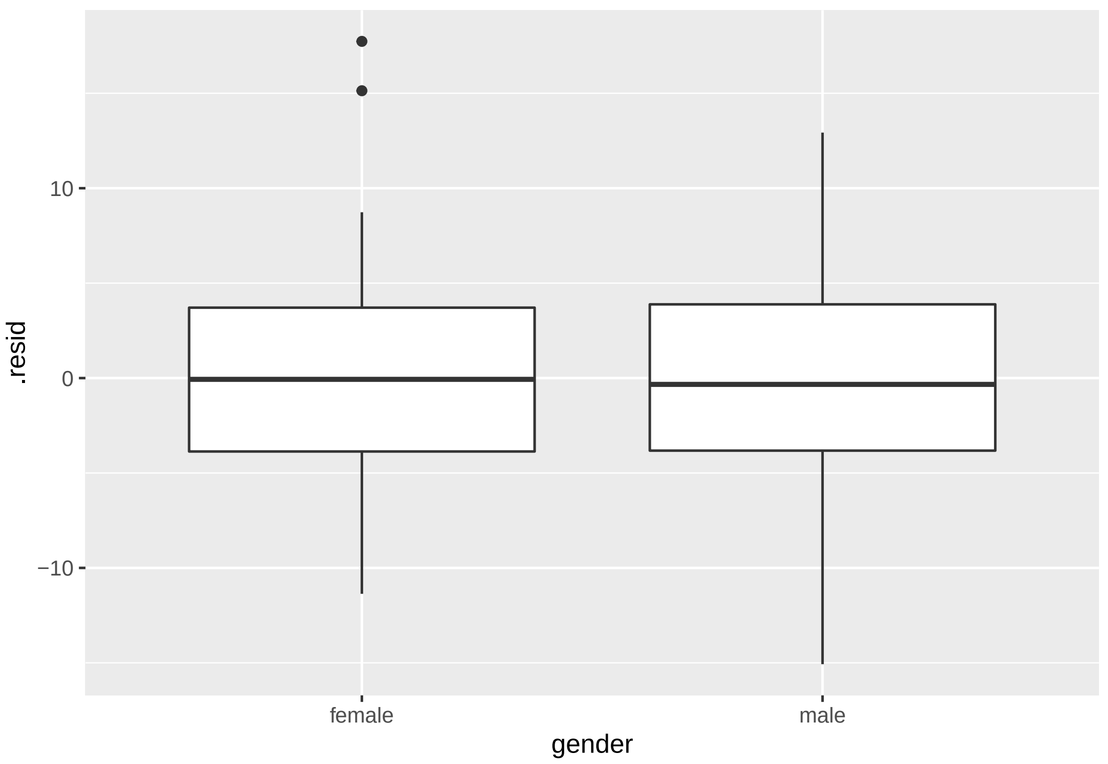
```
## # A tibble: 100 x 12
##         id gender bdate        bdi bdigroup medicine .fitted .resid .std.resid
##      <dbl> <chr>  <date>     <dbl>    <dbl> <chr>      <dbl>  <dbl>      <dbl>
##  1 2.02e6 female 1959-01-01    34        4 placebo     37.9 -3.87      -
0.660
##  2 2.02e6 male   1951-01-01    35        4 homeopa~    34.1  0.929     0.159
##  3 2.02e6 female 1952-01-01    19        3 pharmac~    25.3 -6.27     -1.07
##  4 2.02e6 male   1962-01-01    38        4 none        37.3  0.667     0.115
##  5 2.02e6 female 1981-01-01    38        4 placebo     37.9  0.133     0.0228
##  6 2.02e6 female 1967-01-01    43        4 homeopa~    37.4  5.64      0.975
##  7 2.02e6 female 1949-01-01    20        3 pharmac~    25.3 -5.27      -
0.899
##  8 2.02e6 female 1991-01-01    51        4 none        45.8  5.23      0.898
##  9 2.02e6 male   1955-01-01    33        4 placebo     30.8  2.20      0.382
## 10 2.02e6 male   1967-01-01    37        4 homeopa~    34.1  2.93      0.501
## # ... with 90 more rows, and 3 more variables: .hat <dbl>, .sigma <dbl>,
## #   .cooksd <dbl>
```

```
ggplot(depression_aug, aes(x = medicine, y = .resid)) + geom_boxplot()
```



These look more or less symmetric and centred at zero, but with a few outliers. Enough to worry about? Up to you.

```
ggplot(depression_aug, aes(x = gender, y = .resid)) + geom_boxplot()
```



Centred at zero, more or less symmetric and with equal spread except for those two outliers.

The problem is, if we don't like the ANOVA, we don't have anything to do instead. There is no two-way Welch ANOVA, and there is no equivalent to Mood's median test. All we can do is to run the ANOVA anyway, look at these plots, and express our doubts.

I have idly thought about generalizing Mood's median test to this situation. The idea I had is to work out the grand median of `bdi`, and then count up the number of data values above and below that for each combination of `gender` and `medicine`. It looks as if that would produce something like a three-way table of frequencies, and that might be amenable to a log-linear analysis (which we learn about right at the end of this course). I need to explore that more carefully, and also to find out whether anybody has done that before (there are a lot of smart people in this world, and my experience is if I can work it out, then somebody else has already done so, probably back in the 1950s).

Notes

1. You'll notice I am not using the word "shorter".

2. Cautious, "wary and unwilling to take risks".

3. This means moving the points slightly away from where they really should plot so that you can see them all.

4. *augment* is from *broom*.