Assignment 1

Instructions (same as for STAC32): Make an R Notebook and in it answer the question below. When you are done, hand in on Quercus the *output* from Previewing (or Knitting) your Notebook. Do *not* hand in the Notebook itself. You want to show that you can (i) write code that will answer the questions, (ii) run that code and get some sensible output, (iii) write some words that show you know what is going on and that reflect your conclusions about the data. Your goal is to convince the grader that you *understand* what you are doing: not only doing the right thing, but making it clear that you know *why* it's the right thing.

Do *not* expect to get help on this assignment. The purpose of the assignments is for you to see how much *you* have understood. You will find that you also learn something from grappling with the assignments. The time to get help is after you watch the lectures and work through the problems from PASIAS, via tutorial and the discussion board, 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 4 hours to complete this assignment after you start it.

1. How does the stock market react to changes in interest rates and unemployment rates? An economist recorded the interest rate and unemployment rate each month, and also the "stock index price", a measure of how well the stock market is doing. The data are in http://ritsokiguess.site/STAD29/stocks.txt.

   (a) Take a look at the data. How is it arranged? How, therefore, will you be reading in the data? (You might need to refer back to your C32 notes.)

   Solution:

   The data are in aligned columns. Thus, we will shortly be using `read_table`.

   (b) Read in and display (some of) the data. (You will need to find, for yourself, something that works in order to do the rest of the question.)

   Solution:

   As mentioned above, `read_table`:

```
my_url <- "http://ritsokiguess.site/STAD29/stocks.txt"
my_url
```

```
## [1] "http://ritsokiguess.site/STAD29/stocks.txt"
```

```
stocks <- read_table(my_url)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Row = col_double(),
##   Year = col_double(),
##   Month = col_double(),
##   Interest_Rate = col_double(),
```

```
##   Unemployment_Rate = col_double(),
##   Stock_Index_Price = col_double()
## )
```

```
stocks
```

```
## # A tibble: 24 x 6
##      Row  Year Month Interest_Rate Unemployment_Rate Stock_Index_Price
##    <dbl> <dbl> <dbl>         <dbl>              <dbl>              <dbl>
##  1     0  2017    12          2.75                5.3               1464
##  2     1  2017    11          2.5                 5.3               1394
##  3     2  2017    10          2.5                 5.3               1357
##  4     3  2017     9          2.5                 5.3               1293
##  5     4  2017     8          2.5                 5.4               1256
##  6     5  2017     7          2.5                 5.6               1254
##  7     6  2017     6          2.5                 5.5               1234
##  8     7  2017     5          2.25                5.5               1195
##  9     8  2017     4          2.25                5.5               1159
## 10     9  2017     3          2.25                5.6               1167
## # ... with 14 more rows
```

Extra 1: the base R `read.table` also works:

```
stocks0 <- read.table(my_url, header = TRUE)
stocks0
```

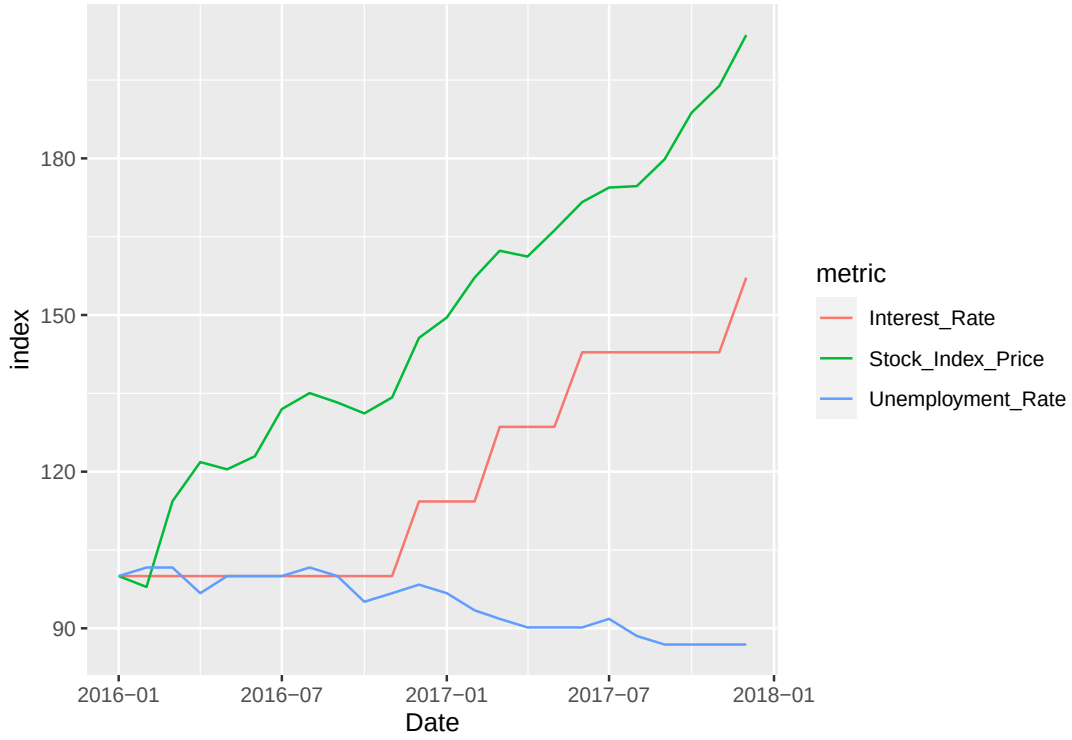
```
##    Row Year Month Interest_Rate Unemployment_Rate Stock_Index_Price
## 1    0 2017    12          2.75                5.3               1464
## 2    1 2017    11          2.50                5.3               1394
## 3    2 2017    10          2.50                5.3               1357
## 4    3 2017     9          2.50                5.3               1293
## 5    4 2017     8          2.50                5.4               1256
## 6    5 2017     7          2.50                5.6               1254
## 7    6 2017     6          2.50                5.5               1234
## 8    7 2017     5          2.25                5.5               1195
## 9    8 2017     4          2.25                5.5               1159
## 10   9 2017     3          2.25                5.6               1167
## 11  10 2017     2          2.00                5.7               1130
## 12  11 2017     1          2.00                5.9               1075
## 13  12 2016    12          2.00                6.0               1047
## 14  13 2016    11          1.75                5.9                965
## 15  14 2016    10          1.75                5.8                943
## 16  15 2016     9          1.75                6.1                958
## 17  16 2016     8          1.75                6.2                971
## 18  17 2016     7          1.75                6.1                949
## 19  18 2016     6          1.75                6.1                884
## 20  19 2016     5          1.75                6.1                866
## 21  20 2016     4          1.75                5.9                876
## 22  21 2016     3          1.75                6.2                822
## 23  22 2016     2          1.75                6.2                704
## 24  23 2016     1          1.75                6.1                719
```

If you can't figure out any other way to get it to work, you can use this, but *I never taught you this*, so if you use it, you have to say where you got it from. This can cause trouble if you

have text columns (they get turned into `factors`), but everything is numeric here, so there are no problems in this case. (`read.table` reads columns separated by any number of spaces, but you should probably read section 11.2.1 of this for discussion of why using it yourself is a bad idea unless you know what you're doing, and maybe even then.)

Extra 2: these are actually in backwards time order, which doesn't make much sense from an economic point of view, but we're not using the time order of these data. An economist would plot everything against time, using three different y-axes (!) or by indexing the values so that the first one is 100 on each variable and everything is relative to that. This is a bit fiddly, but can be done:

```
library(lubridate)
stocks %>%
  arrange(Year, Month) %>%
  mutate(datetext = str_c(Year, "-", Month, "-1")) %>%
  mutate(Date = ymd(datetext)) %>%
  pivot_longer(contains("_"), names_to = "metric", values_to = "value") %>%
  group_by(metric) %>%
  mutate(index = value / value[1] * 100) %>%
  ggplot(aes(x = Date, y = index, colour = metric)) + geom_line()
```



Over time, economic conditions were improving: the interest rate and stock index price were going up, and the unemployment rate was going down. That, however, is not our focus in this question.

To talk about my code:

- turn the years and months into proper R dates:
  - glue them together as text with a day 1 to represent 1st of the month

  - use `ymd` from `lubridate` to make real dates. (`as.Date` would also have worked here.)
- bearing in mind that `ggplot` likes the x and y for a plot in one column each, pivot longer. I've called the economic indicators `metric` and their value `value`.
- create the index values using Jan 2016 as 100. What I want to scale by is the first one *of each metric*, so the idea of grouping and the use of `[1]` to indicate the first one (within each group, that is, metric), will get this done.
- plot the time traces of index values for each metric against time, distinguishing them by colour.

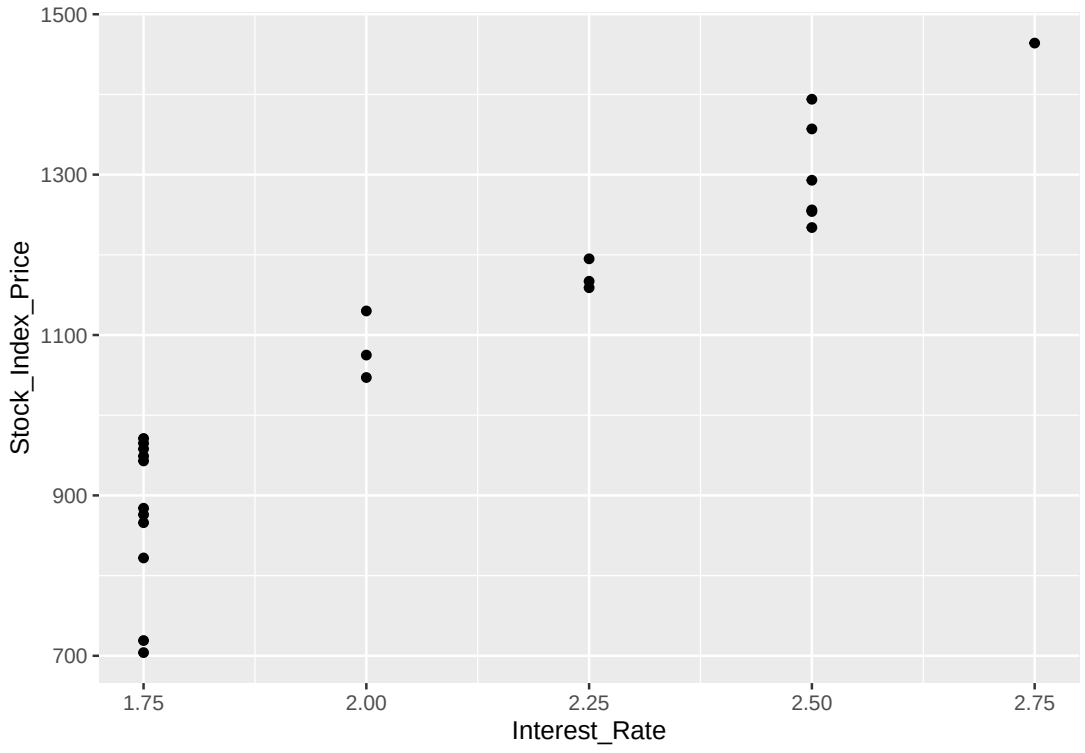
You may choose to be happy that I didn't ask *you* to figure that out!

(c) Make plots of stock index price against interest rate and against unemployment rate.

Solution:

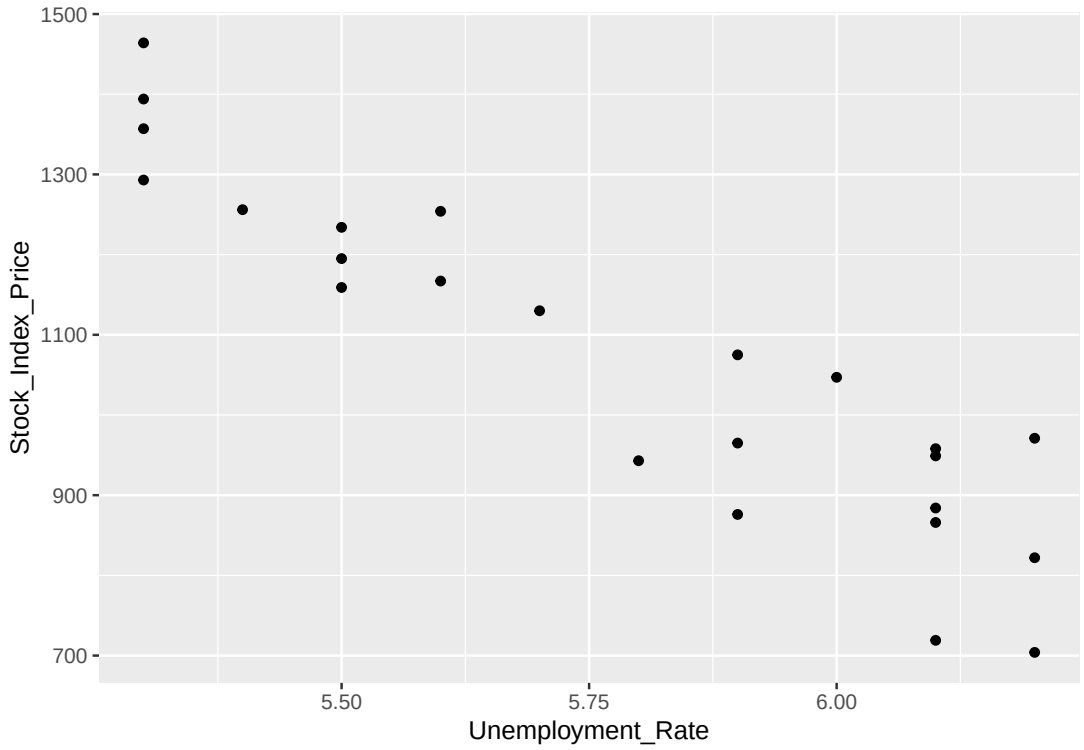
The obvious way is fine. These are (evidently) scatterplots. Note that the variable names have Capital Letters:

```
ggplot(stocks, aes(x=Interest_Rate, y=Stock_Index_Price)) + geom_point()
```



and

```
ggplot(stocks, aes(x=Unemployment_Rate, y=Stock_Index_Price)) + geom_point()
```

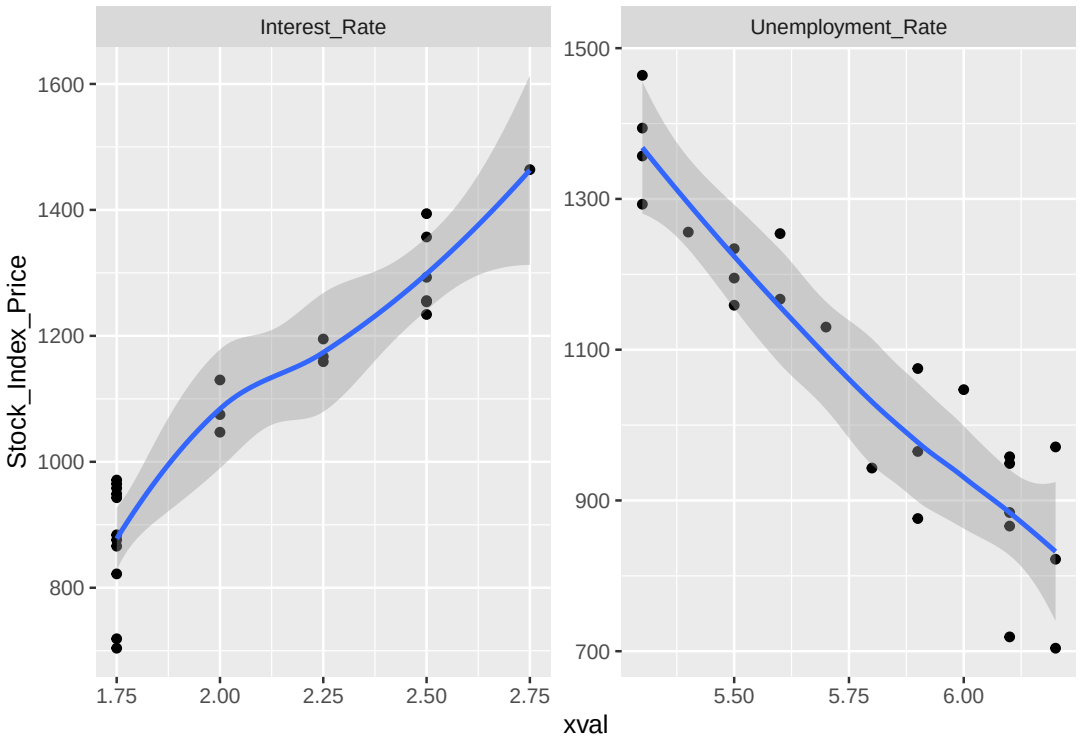


Add `geom_smooth()`s to these if you like. See below for why I think that is better than adding regression lines at this stage.

If you wish to show off, get them both on one plot, using facets. Add a `geom_smooth` if you like:

```
stocks %>% pivot_longer(ends_with("rate"), names_to = "xname", values_to = "xval") %>%
  ggplot(aes(x=xval, y=Stock_Index_Price)) + geom_point() + geom_smooth() +
  facet_wrap(~xname, scales = "free")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



The two x-variables have different kinds of values, so `scales=free` is needed. (I forgot, the first time, to put the columns in `names_to` and `values_to` in quotes. Happens to us all.)

I like a plain `geom_smooth` better than a regression line (with `method="lm"`) because this will make it clearer whether the trends we have *are* lines, or whether we'll need to be thinking about fitting curves. Adding a regression line to the plot makes more sense later, after we've decided that a straight line is the thing, rather than now. At this stage, we are still in exploratory mode, asking "what do we have", and allowing the smooth to be a curve gives us a clearer sense of what we have.

(d) Describe what you see in your scatterplots. Comment on anything relevant to a linear regression (which we will fit shortly).

Solution:

"Form, direction, strength", or something equivalent: linear or curved, up or down, strong, moderate or weak. Comment on anything else interesting that you see:

- interest rate: more or less linear, upward, and a strong relationship (I would say). The only doubt I have is that the prices at the lowest interest rate seem to be too spread-out, or they go down too far. If it weren't for that, I would really call this a very strong linear trend.
- unemployment rate: a clear linear downward trend (not much doubt about either of those). This is also mostly a strong trend. The only thing that stops me calling it "strong" without reservation is those points where the unemployment rate is high (on the right); some of them are further away from the trend (another way to say this is that there is a hint of fanning out).

(e) Fit a multiple regression predicting stock index price from interest rate and unemployment rate, and display the results.

Solution:

The usual:

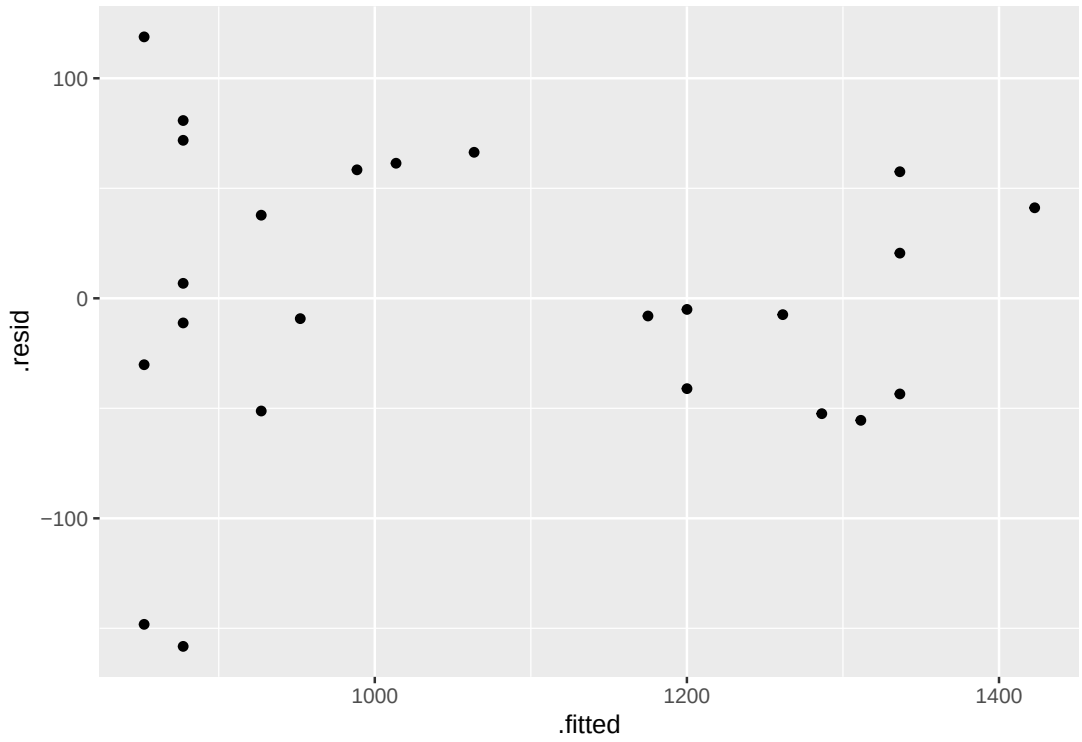
```
stocks.1 <- lm(Stock_Index_Price~Interest_Rate+Unemployment_Rate,  data = stocks)
summary(stocks.1)
```

```
##
## Call:
## lm(formula = Stock_Index_Price ~ Interest_Rate + Unemployment_Rate,
##     data = stocks)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -158.205  -41.667   -6.248   57.741  118.810
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1798.4      899.2   2.000  0.05861 .
## Interest_Rate        345.5      111.4   3.103  0.00539 **
## Unemployment_Rate   -250.1      117.9  -2.121  0.04601 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.56 on 21 degrees of freedom
## Multiple R-squared:  0.8976, Adjusted R-squared:  0.8879
## F-statistic: 92.07 on 2 and 21 DF,  p-value: 4.043e-11
```

You might note that both explanatory variables are significant, so the model we have here looks good.

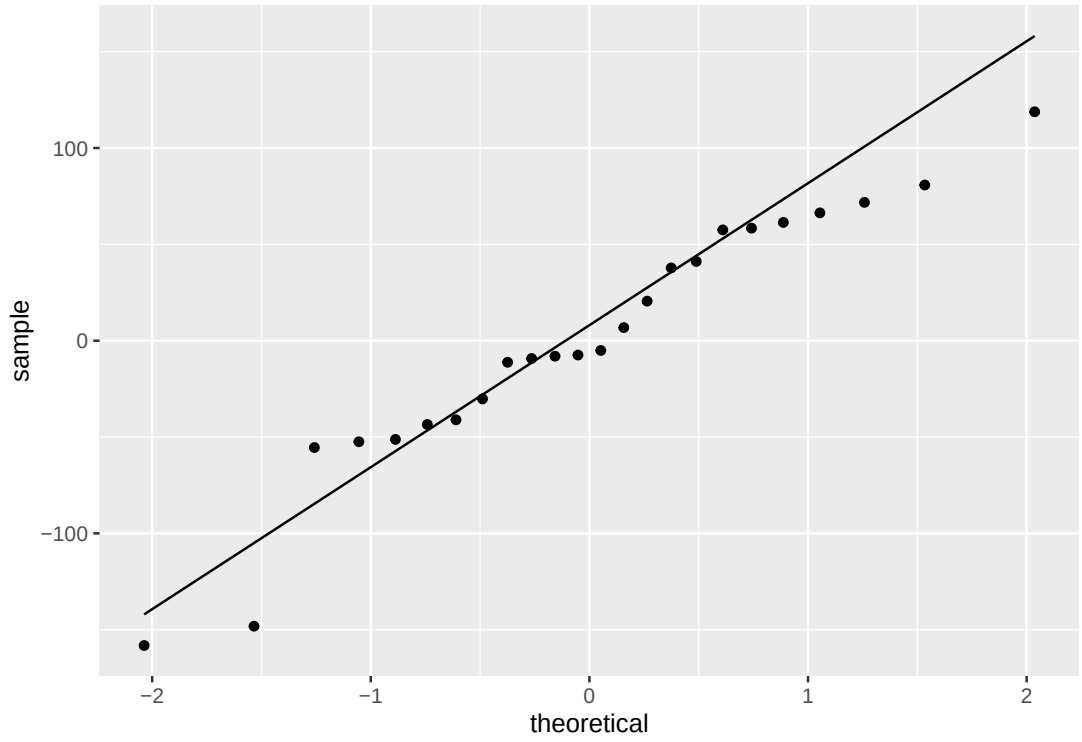
Extra: normally we'd look at residuals:

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



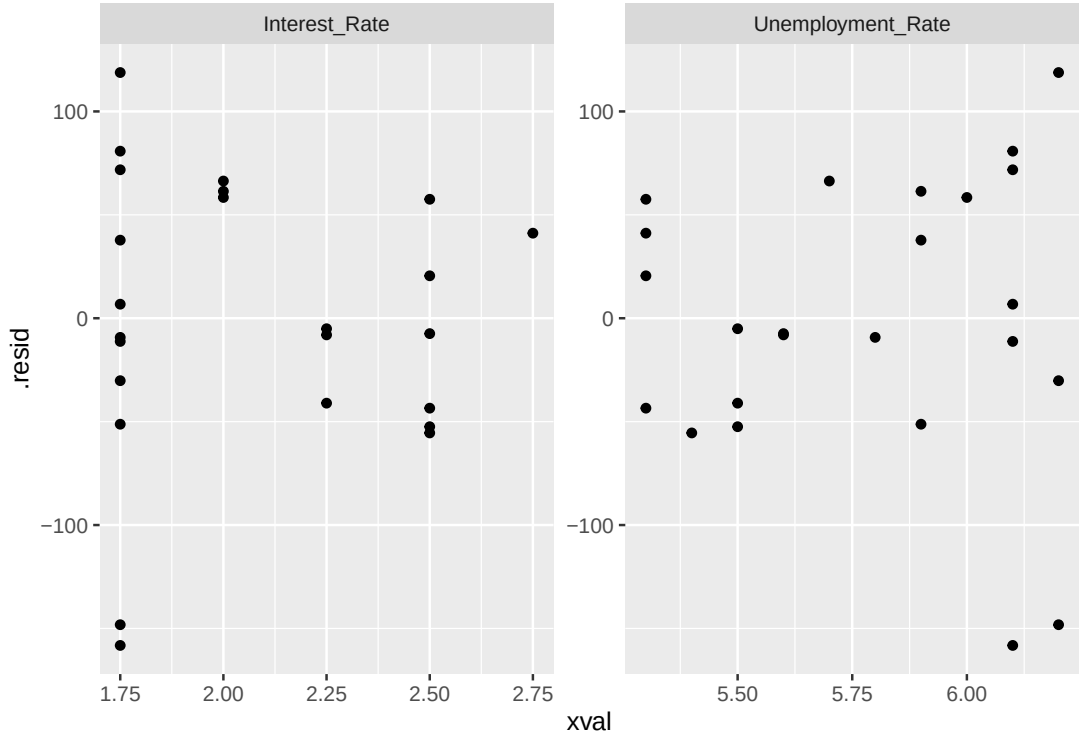
Is that fanning-in? I'd be more inclined to think of it as two low outliers.

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



However, in the context of all the residuals, the lowest two are not really too low, even though they are very different from the others (the third-lowest one is too high). And, using a code idea from C32:[1]

```
stocks.1 %>% augment(stocks) %>%
  pivot_longer(ends_with("Rate"), names_to="xname", values_to="xval") %>%
  ggplot(aes(x=xval, y=.resid)) + geom_point() + facet_wrap(~xname, scales = "free")
```



Those two low residuals are standing out again, but other than that I don't see any problems.

(f) Predict the mean stock index price for all months that have (i) interest rate 2.00 and unemployment rate 5.75; (ii) interest rate 2.75 and unemployment rate 6.00. Obtain suitable intervals for each of your (two) predictions. Display the predictions next to the values they are predictions for.

Solution:

This is going to be `predict` with `interval="c"`, because we are looking for confidence intervals for the mean response under each of those two conditions. But first, you need to set up a data frame whose values are the values to predict for, and whose column names are the names of the explanatory variables they are values of. I like to use `tribble` for this job:

```
new <- tribble(
  ~Interest_Rate, ~Unemployment_Rate,
      2.00,             5.75,
      2.75,             6.00
)
new
```

```
## # A tibble: 2 x 2
##   Interest_Rate Unemployment_Rate
##           <dbl>             <dbl>
## 1          2                 5.75
## 2          2.75              6
```

If you don't like this, make a mini-data file, and read it in. You can also read in a piece of text as if it were a data file:

```
text <- "
Interest_Rate Unemployment_Rate
2.00 5.75
2.75 6.00
"
read_delim(text, " ")
```

```
## # A tibble: 2 x 2
##   Interest_Rate Unemployment_Rate
##           <dbl>             <dbl>
## 1          2                 5.75
## 2          2.75              6
```

Or, of course, use `tibble` and put the data values inside `c`:

```
tibble(Interest_Rate = c(2.00, 2.75), Unemployment_Rate = c(5.75, 6.00))
```

```
## # A tibble: 2 x 2
##   Interest_Rate Unemployment_Rate
##           <dbl>             <dbl>
## 1          2                 5.75
## 2          2.75              6
```

I don't mind what your process is *as long as you show it*, and you end up with the right predictions. This is a little unlike the lecture notes, so you will have to think some, but the clue is that the *columns* of the data frame you create will need to have the same names as the explanatory variables you had in the regression, and this will help you get the numbers in the right places.

Then run `predict`, with three inputs: the fitted model, the new values to predict for, and the kind of interval to get:

```
p <- predict(stocks.1, new, interval = "c")
cbind(new, p)
```

```
##   Interest_Rate Unemployment_Rate      fit      lwr      upr
## 1          2.00              5.75 1051.141 1013.568 1088.715
## 2          2.75              6.00 1247.760 1036.671 1458.848
```

(g) Explain briefly why the interval for interest rate 2.75 and unemployment rate 6.00 is longer than for the interest rate 2.00 and unemployment rate 5.75. There are (best) *two* reasons. You may have to do some further investigation of the data to support your claims.

Solution:

You might guess that the first row of my `new` is closer to the mean on both variables than the

second row. Is this true?

```
stocks %>% summarize(m1=mean(Interest_Rate), m2=mean(Unemployment_Rate))
```

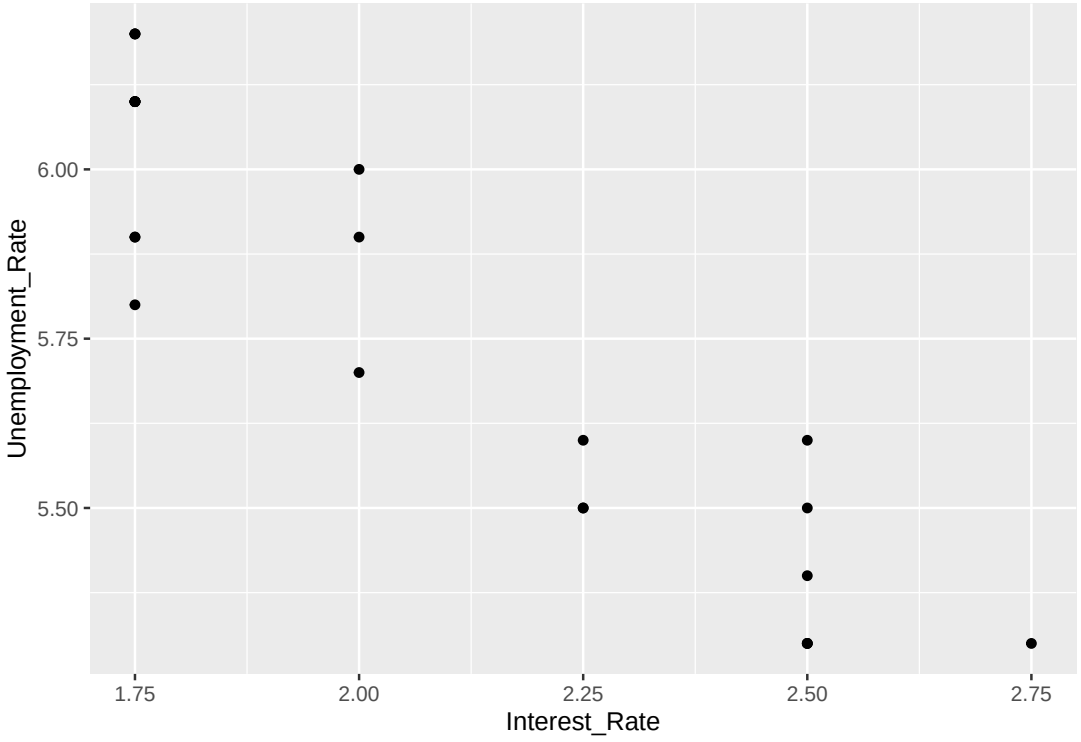
```
## # A tibble: 1 x 2
##      m1    m2
##   <dbl> <dbl>
## 1  2.07  5.78
```

The first row is very close to the mean on both variables, while the second one is further away, so that's why its interval is shorter. The first reason.

Or you can infer this from your scatterplots in (c): the centre of the distribution of interest rate is between 2 and 2.25 (because there are a lot of values at 1.75), and of unemployment rate is somewhere near 5.75.

To think about the second reason, recall that predictions for values that are "like the data" will be more accurate than ones that are unlike the data, or "unusual" if you prefer to think of it that way. How are the values in the second row unusual? One way that we have already seen is that they are far from the means. But there is something else: there are two explanatory variables, so maybe the *combination* of them is unusual too? To see whether this is the case, one way is to plot the two explanatory variables against *each other*:

```
ggplot(stocks, aes(x=Interest_Rate, y=Unemployment_Rate)) + geom_point()
```



The values 2.75 and 6.00 are up in the top right corner of this plot, where there are *no observations at all*. It's hard to predict for observations unlike what you observed (indeed, it's extrapolation), so the interval came out longer than the other one, to reflect that we don't really know what's going to happen *even if* the linear model continues to hold (which is another question). That's the second reason: the explanatory variable values are unusual *in combination*.

When the interest rate is high, the unemployment rate in the data is low (and vice versa), but the values we are predicting for are high on both.

Notes

1. This needs *broom*.