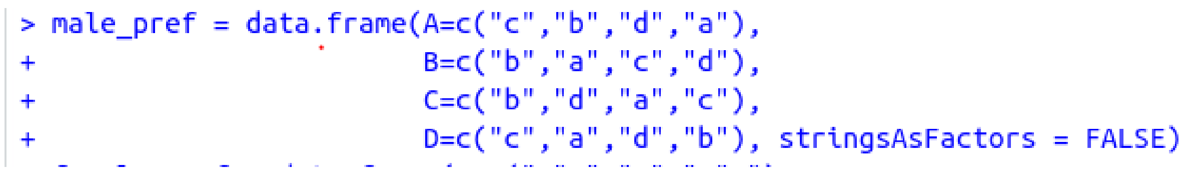
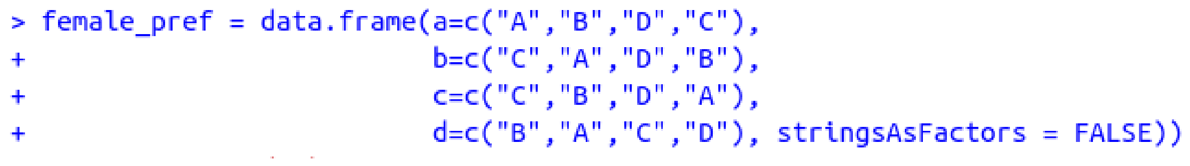
Task 1

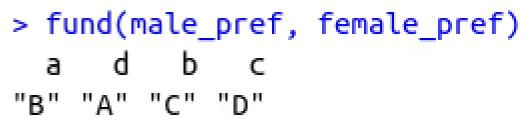
I tested my function in R by inputting the example preference tables from the notes and checking that it outputting the correct stable matching.



```
> male_pref = data.frame(A=c("c","b","d","a"),
+                        B=c("b","a","c","d"),
+                        C=c("b","d","a","c"),
+                        D=c("c","a","d","b"), stringsAsFactors = FALSE)
```



```
> female_pref = data.frame(a=c("A","B","D","C"),
+                          b=c("C","A","D","B"),
+                          c=c("C","B","D","A"),
+                          d=c("B","A","C","D"), stringsAsFactors = FALSE))
```

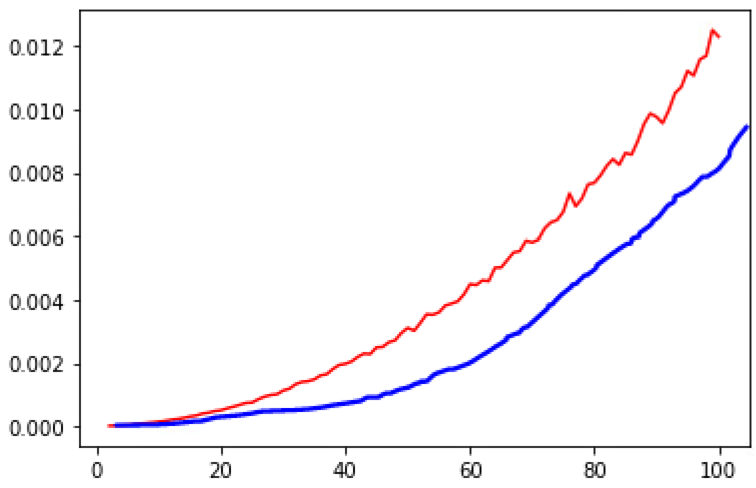


```
> fund(male_pref, female_pref)
  a   d   b   c 
"B" "A" "C" "D" 
```

This is the correct stable matching from the notes.

Task 2

I ran out of time here unfortunately; however, I would assume the times for the R-C++ implementation would be faster than that for python. Using the figure from the original python assessment, I have added a blue line representing how I would imagine the R-C++ implementation to scale.



I imagine the function would scale quadratically for the same reason the python implementation did: because the fundamental algorithm consists of two nested loops.

Task 3

Python Implementation

Replicable (5/5) I added detailed explanation of the inputs, processes and outputs of each function with the aim that it would be possible to just read these and replicate the behaviour of the function in any programming language.

Repeatable (5/5) To account for the randomness, I set the random number seed at the start of the notebook.

Re-runnable (5/5) I provided details of the Python version, the Jupyter notebook version and the OS the code was tested on, so that another person could recreate the same environment I worked in.

Reproducible (5/5) I added assertions to check the random number generator was behaving the same each time. I also made sure the plots were from the latest version of the code.

Reusable (5/5) I added more in-line comments to code and provided more detailed information about the system the code had been tested on - e.g. OS version.

C++ Implementation

Replicable (3/5) I added some information about the inputs and processes of the function in comments, however, not in as much detail.

Repeatable (3/5) I did not set a random seed, however, this wasn't as necessary as the notebook would output the same random preference tables each time anyway.

Re-runnable (0/5) I didn't provide any of the details that I did in the python implementation, such as OS version.

Reproducible (4/5) I didn't add any assertions which could have been useful. Otherwise, the code was reproducible.

Reusable (0/5) I have not provided details of the OS etc.

RCPP implementation

Replicable (2/5) Again, there is some information about the inputs etc. in comments, but not in as much detail as I would have liked.

Repeatable (5/5) There is no randomness here, so no setting of a seed is required for the code to be repeatable.

Re-runnable (0/5) Same as above, I haven't provided any details such as the OS version.

Reproducible (5/5) There is no randomness, so no need for assertions etc., therefore the code is reproducible.

Reusable (0/5) I have not provided details of the OS etc.