Homework4

Patrick Seebold

```
# start by opening some libraries we will need
library(readr) # for loading data
library(readxl) # for excel files
library(dplyr) # for manipulating data
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Question 1: Conceptual Questions

1.1) If your working directory is myfolder/homework/, what relative path would you specify to get the file located at myfolder/MyData.csv?

> With this setup, our relative path to the specified file would be "../MyData.csv"

1.2.) What are the major benefits of using R projects?

> R projects are glorious things! They let easily integrate with github, which makes collaborations much easier. Also, they make switching between and resuming projects much easier since save they the R environment for that project. I've had many Rprojects over the course of my PhD and MStat training, and keeping every project in its own Rproject made it easier to switch between them without losing track of things.

1.3) What is git and what is github?

> Git is a version control software which helps us keep track of past versions of our projects. Github is an online platform that allows us to store and share projects with other data scientists/programmers. It integrates nicely with git, allowing us to push and pull our local version-controlled projects to a remote repository.

1.4) What are the two main differences between a tibble and a data.frame?

> We discussed two main differences between tibbles, which are updated versions of data frames that "do less and complain more". This is meant to make it easier for us to see the problems in our data and solve them more quickly. The two examples we discussed in lecture were that they provide a fancier print output, and they do not coerce columns down to vectors when using the [ ] notation.

1.5). Rewrite the following nested function call using BaseR's chaining operator: arrange(filter(select(as_tibble(iris), starts_with("Petal"), Species), Petal.Length < 1.55), Species)

```
# this is easier to show the format in a coding chunk!
iris |>
  as_tibble() |>
  select(starts_with("Petal"), Species) |>
  filter(Petal.Length < 1.55) |>
  arrange(Species)
```

```
# A tibble: 37 x 3
   Petal.Length Petal.Width Species
          <dbl>       <dbl> <fct>
 1          1.4         0.2 setosa
 2          1.4         0.2 setosa
 3          1.3         0.2 setosa
 4          1.5         0.2 setosa
 5          1.4         0.2 setosa
 6          1.4         0.3 setosa
 7          1.5         0.2 setosa
 8          1.4         0.2 setosa
 9          1.5         0.1 setosa
10          1.5         0.2 setosa
# i 27 more rows
```

Task 2: Reading Delimited data

First we load in our glass data from the online source:

```
glass = read_csv('https://www4.stat.ncsu.edu/~online/datasets/glass.data',
                 col_names = c('Id','RI','Na','Mg','Al','Si','K','Ca',
                               'Ba', 'Fe','Type_of_glass'))
```

```
Rows: 214 Columns: 11
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (11): Id, RI, Na, Mg, Al, Si, K, Ca, Ba, Fe, Type_of_glass

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glass # print out the tibble by calling the name
```

```
# A tibble: 214 x 11
      Id    RI    Na    Mg    Al    Si     K    Ca    Ba    Fe Type_of_glass
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>         <dbl>
 1     1  1.52  13.6  4.49  1.1   71.8  0.06  8.75     0  0                1
 2     2  1.52  13.9  3.6   1.36  72.7  0.48  7.83     0  0                1
 3     3  1.52  13.5  3.55  1.54  73.0  0.39  7.78     0  0                1
 4     4  1.52  13.2  3.69  1.29  72.6  0.57  8.22     0  0                1
 5     5  1.52  13.3  3.62  1.24  73.1  0.55  8.07     0  0                1
 6     6  1.52  12.8  3.61  1.62  73.0  0.64  8.07     0  0.26             1
 7     7  1.52  13.3  3.6   1.14  73.1  0.58  8.17     0  0                1
 8     8  1.52  13.2  3.61  1.05  73.2  0.57  8.24     0  0                1
 9     9  1.52  14.0  3.58  1.37  72.1  0.56  8.3      0  0                1
10    10  1.52  13    3.6   1.36  73.0  0.57  8.4      0  0.11             1
# i 204 more rows
```

This completes the first step of this task. Now we handle parts 2 and 3, where we will chain our functions to mutate and filter our tibble as specified.

```
# Now start the chain to add the factor vector for type of glass
glass |> # first we feed our glass tibble to mutate, which includes the factor()
  mutate(Type_of_glass = factor(Type_of_glass,levels = c(1,2,3,4,5,6,7),
         labels = c("building_windows_float_processed",
                    "building_windows_non_float_processed",
```

```
                    "vehicle_windows_float_processed",
                    "vehicle_windows_non_float_processed",
                    "containers", "tableware", "headlamp")))|>
  # then we filter our the specific subset we want with filter()!
  filter(Fe < 0.2 & Type_of_glass %in% c('tableware','headlamp'))
```

```
# A tibble: 38 x 11
      Id    RI    Na    Mg    Al    Si     K    Ca    Ba    Fe Type_of_glass
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct>
 1   177  1.52  14    2.39  1.56  72.4  0     9.57  0        0 tableware
 2   178  1.52  13.8  2.41  1.19  72.8  0     9.77  0        0 tableware
 3   179  1.52  14.5  2.24  1.62  72.4  0     9.26  0        0 tableware
 4   180  1.52  14.1  2.19  1.66  72.7  0     9.32  0        0 tableware
 5   181  1.51  14.4  1.74  1.54  74.6  0     7.59  0        0 tableware
 6   182  1.52  15.0  0.78  1.74  72.5  0     9.95  0        0 tableware
 7   183  1.52  14.2  0     2.09  72.7  0    10.9   0        0 tableware
 8   184  1.52  14.6  0     0.56  73.5  0    11.2   0        0 tableware
 9   185  1.51  17.4  0     0.34  75.4  0     6.65  0        0 tableware
10   186  1.51  13.7  3.2   1.81  72.8  1.76  5.43  1.19     0 headlamp
# i 28 more rows
```

We see that only 38 of our total observations meet these criteria, and they are now contained within the tibble above.

Yeast Data:

Now we will do the second part of Task 2 with the yeast data. First we get the data:

```
yeast = read_table("https://www4.stat.ncsu.edu/~online/datasets/yeast.data",
                   col_names = c("seq_name","mcg","gvh","alm","mit",
                                 "erl","pox","vac","nuc","class"))
```

```
-- Column specification --------------------------------------------------------
cols(
  seq_name = col_character(),
  mcg = col_double(),
  gvh = col_double(),
  alm = col_double(),
```

```
  mit = col_double(),
  erl = col_double(),
  pox = col_double(),
  vac = col_double(),
  nuc = col_double(),
  class = col_character()
)
```

```
yeast # print out the tibble - all looks good!
```

```
# A tibble: 1,484 x 10
   seq_name     mcg   gvh   alm   mit   erl   pox   vac   nuc class
   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
 1 ADT1_YEAST  0.58  0.61  0.47  0.13   0.5   0    0.48  0.22 MIT
 2 ADT2_YEAST  0.43  0.67  0.48  0.27   0.5   0    0.53  0.22 MIT
 3 ADT3_YEAST  0.64  0.62  0.49  0.15   0.5   0    0.53  0.22 MIT
 4 AAR2_YEAST  0.58  0.44  0.57  0.13   0.5   0    0.54  0.22 NUC
 5 AATM_YEAST  0.42  0.44  0.48  0.54   0.5   0    0.48  0.22 MIT
 6 AATC_YEAST  0.51  0.4   0.56  0.17   0.5   0.5  0.49  0.22 CYT
 7 ABC1_YEAST  0.5   0.54  0.48  0.65   0.5   0    0.53  0.22 MIT
 8 BAF1_YEAST  0.48  0.45  0.59  0.2    0.5   0    0.58  0.34 NUC
 9 ABF2_YEAST  0.55  0.5   0.66  0.36   0.5   0    0.49  0.22 MIT
10 ABP1_YEAST  0.4   0.39  0.6   0.15   0.5   0    0.58  0.3  CYT
# i 1,474 more rows
```

Now we can start our chain to make the desired changes:

```
# first remove seq_name and nuc columns
yeast_processed = yeast|>
  select(c(mcg, gvh,alm,mit,erl,pox,vac,class))|> # select all but  seqname & nuc
  group_by(class)|> # group by class before adding the new columns
  mutate(across(where(is.numeric), list(mean = mean,median = median),
                .names= "{.col}_{.fn}")) # we can tell .names to use the fn as
                                         # as well as the names for each measure
yeast_processed # view the updated version of the yeast data
```

```
# A tibble: 1,484 x 22
# Groups:   class [10]
     mcg   gvh   alm   mit   erl   pox   vac class mcg_mean mcg_median gvh_mean
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>    <dbl>      <dbl>    <dbl>
 1  0.58  0.61  0.47  0.13   0.5   0    0.48 MIT      0.521       0.51    0.533
```

```
 2  0.43  0.67  0.48  0.27   0.5   0    0.53 MIT      0.521       0.51    0.533
 3  0.64  0.62  0.49  0.15   0.5   0    0.53 MIT      0.521       0.51    0.533
 4  0.58  0.44  0.57  0.13   0.5   0    0.54 NUC      0.452       0.45    0.456
 5  0.42  0.44  0.48  0.54   0.5   0    0.48 MIT      0.521       0.51    0.533
 6  0.51  0.4   0.56  0.17   0.5   0.5  0.49 CYT      0.481       0.48    0.470
 7  0.5   0.54  0.48  0.65   0.5   0    0.53 MIT      0.521       0.51    0.533
 8  0.48  0.45  0.59  0.2    0.5   0    0.58 NUC      0.452       0.45    0.456
 9  0.55  0.5   0.66  0.36   0.5   0    0.49 MIT      0.521       0.51    0.533
10  0.4   0.39  0.6   0.15   0.5   0    0.58 CYT      0.481       0.48    0.470
# i 1,474 more rows
# i 11 more variables: gvh_median <dbl>, alm_mean <dbl>, alm_median <dbl>,
#   mit_mean <dbl>, mit_median <dbl>, erl_mean <dbl>, erl_median <dbl>,
#   pox_mean <dbl>, pox_median <dbl>, vac_mean <dbl>, vac_median <dbl>
```

Task 3: Combining Data Sets

Now we can move onto our final task - merging these two wine data sets.

```
# first, load in the first sheet of the xlsx white wine sheet already in our wd
white_wine = read_excel("white-wine.xlsx", sheet = "white-wine")
white_wine
```

```
# A tibble: 4,898 x 12
   `fixed acidity` `volatile acidity` `citric acid` `residual sugar` chlorides
             <dbl>              <dbl>         <dbl>             <dbl>     <dbl>
 1               7               0.27          0.36              20.7     0.045
 2              63               0.3           0.34               1.6     0.049
 3              81               0.28          0.4                6.9     0.05
 4              72               0.23          0.32               8.5     0.058
 5              72               0.23          0.32               8.5     0.058
 6              81               0.28          0.4                6.9     0.05
 7              62               0.32          0.16               7       0.045
 8               7               0.27          0.36              20.7     0.045
 9              63               0.3           0.34               1.6     0.049
10              81               0.22          0.43               1.5     0.044
# i 4,888 more rows
# i 7 more variables: `free sulfur dioxide` <dbl>,
#   `total sulfur dioxide` <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
#   alcohol <dbl>, quality <dbl>
```

Since some of our titles have spaces, we will replace them with versions without spaces, which we can grab from the second sheet of the excel:

```
white_wine_titles = read_excel("white-wine.xlsx", sheet = "variables")
white_wine_titles = white_wine_titles$Variables # change this to character vector
                                                # for use with colnames

colnames(white_wine) = white_wine_titles # set the new names to white_wine tibble
# Finally, add a wine type column where all of these are white
white_wine$wine_type = 'white' # nice and simple!

white_wine # success! We now have the proper names and the wine_type variable
```

```
# A tibble: 4,898 x 13
   fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
 1             7             0.27        0.36           20.7     0.045
 2            63             0.3         0.34            1.6     0.049
 3            81             0.28        0.4             6.9     0.05
 4            72             0.23        0.32            8.5     0.058
 5            72             0.23        0.32            8.5     0.058
 6            81             0.28        0.4             6.9     0.05
 7            62             0.32        0.16            7       0.045
 8             7             0.27        0.36           20.7     0.045
 9            63             0.3         0.34            1.6     0.049
10            81             0.22        0.43            1.5     0.044
# i 4,888 more rows
# i 8 more variables: free_sulfur_dioxide <dbl>, total_sulfur_dioxide <dbl>,
#   density <dbl>, pH <dbl>, sulphates <dbl>, alcohol <dbl>, quality <dbl>,
#   wine_type <chr>
```

Now let's do the same for the red wine. This time we can read directly from the URL:

```
red_wine = read_delim("https://www4.stat.ncsu.edu/~online/datasets/red-wine.csv",";")
```

```
Rows: 1599 Columns: 12
-- Column specification --------------------------------------------------------
Delimiter: ";"
dbl (12): fixed acidity, volatile acidity, citric acid, residual sugar, chlo...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# The column types are correct - interestingly, they were NOT when I tried open_csv2!
colnames(red_wine) = white_wine_titles # we can use same names as before

# add the wine_type column here too
red_wine$wine_type = 'red'
red_wine
```

```
# A tibble: 1,599 x 13
   fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
 1           7.4              0.7         0              1.9     0.076
 2           7.8              0.88        0              2.6     0.098
 3           7.8              0.76        0.04           2.3     0.092
 4          11.2              0.28        0.56           1.9     0.075
 5           7.4              0.7         0              1.9     0.076
 6           7.4              0.66        0              1.8     0.075
 7           7.9              0.6         0.06           1.6     0.069
 8           7.3              0.65        0              1.2     0.065
 9           7.8              0.58        0.02           2       0.073
10           7.5              0.5         0.36           6.1     0.071
# i 1,589 more rows
# i 8 more variables: free_sulfur_dioxide <dbl>, total_sulfur_dioxide <dbl>,
#   density <dbl>, pH <dbl>, sulphates <dbl>, alcohol <dbl>, quality <dbl>,
#   wine_type <chr>
```

Now we have both wine sets in the same format with the wine_type variable added. Next, we will combine the two together:

```r
# combine the two tibbles as specified
full_wine = dplyr::bind_rows(white_wine,red_wine)
full_wine
```

```
# A tibble: 6,497 x 13
   fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
 1             7             0.27        0.36           20.7     0.045
 2            63             0.3         0.34            1.6     0.049
 3            81             0.28        0.4             6.9     0.05
 4            72             0.23        0.32            8.5     0.058
 5            72             0.23        0.32            8.5     0.058
 6            81             0.28        0.4             6.9     0.05
```

```
 7            62             0.32        0.16            7       0.045
 8             7             0.27        0.36           20.7     0.045
 9            63             0.3         0.34            1.6     0.049
10            81             0.22        0.43            1.5     0.044
# i 6,487 more rows
# i 8 more variables: free_sulfur_dioxide <dbl>, total_sulfur_dioxide <dbl>,
#   density <dbl>, pH <dbl>, sulphates <dbl>, alcohol <dbl>, quality <dbl>,
#   wine_type <chr>
```

Now we can do our final chain:

```
processed_full_wine = full_wine|>
  filter(quality > 6.5 & alcohol < 132) |>
  arrange(desc(quality))|>
  select(contains('acid'),alcohol, wine_type, quality)|>
  group_by(quality)|>
  mutate(across(alcohol, list(mean = mean,sd = sd),
                .names= "{.col}_{.fn}"))
processed_full_wine
```

```
# A tibble: 1,206 x 8
# Groups:   quality [3]
   fixed_acidity volatile_acidity citric_acid alcohol wine_type quality
           <dbl>            <dbl>       <dbl>   <dbl> <chr>       <dbl>
 1            91             0.27        0.45     104 white           9
 2            66             0.36        0.29     124 white           9
 3            74             0.24        0.36     125 white           9
 4            69             0.36        0.34     127 white           9
 5            71             0.26        0.49     129 white           9
 6            62             0.66        0.48     128 white           8
 7            62             0.66        0.48     128 white           8
 8            68             0.26        0.42     105 white           8
 9            67             0.23        0.31     107 white           8
10            67             0.23        0.31     107 white           8
# i 1,196 more rows
# i 2 more variables: alcohol_mean <dbl>, alcohol_sd <dbl>
```

We now have our updated tibble, complete with the alcohol mean and sd for each quality level!