Week 8 - Input / Output

```
knitr::opts_knit$set(root.dir = "D:/Projects/Introduction-to-R/Data")
library(dplyr)                                                    # going to use this for the arrange
```

```
##
## 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
```

In previous weeks, the course briefly discussed the import and export of data. This week will take an even more in-depth look at the ability for R to import and export data files. This includes both local copies of data as well as access to online sources.

Weekly Learning Objectives

Create an R script with the following components:

1. Build a simple R program to read in data from a file and save the results into a data frame.
2. Build a function to read data from a web-URL.

- Use the public archive data on Relative CPU performance:
- URL: http://archive.ics.uci.edu/ml/machine-learning-databases/
- Dataset: machine.data
- Name File: machine.names

3. Build a simple lookup function for CPU performance. Assume the data will be pulled from the same URL, and that data is updated on a frequent basis. Given a vendor name turn the following results:

- A message if the vendor is not found (based on previous experience in the data)
- Vendor published relative performance (PRP) and estimated relative performance (ERP)
- If the vendor is the best in category, return a flag that they are the best. For example, "Best in Published Relative Performance".
- If they are not the best in category, return the vendor is the best including the PRP and ERP numbers.

1. Read data into a data frame

This is simply a load into a data frame. There are multiple ways of solving this question.

Full Points Load data into a data frame. You may also want to print off the results and use the class() function to check to make sure it's a data frame.

```
sample_dataframe <- read.csv("sample.txt", header = TRUE,sep="\t")     #  First row is a header.
                                                                       #  Field separated by a tab (\t)
class(sample_dataframe)    # If it worked it will say data.frame
```

```
## [1] "data.frame"
```

2. Read data from a web URL

This simply extends the function to pull from a web address. Here I will use the

Full Points Load data into a data frame from a web link. You may also want to print off the results and use the class() function to check to make sure it's a data frame.

```
RED_URL <-"http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
RED_data <- read.csv(RED_URL, header = TRUE, sep = ";")
head(RED_data)
```

Red Wine Data

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            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             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality
## 1       5
## 2       5
## 3       5
## 4       6
## 5       5
## 6       5
```

```
WHITE_URL <-"http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.cs
WHITE_data <- read.csv(WHITE_URL, header = TRUE, sep = ";")
head(WHITE_data)
```

White Wine Data

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4           7.2             0.23        0.32            8.5     0.058
## 5           7.2             0.23        0.32            8.5     0.058
## 6           8.1             0.28        0.40            6.9     0.050
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 5                  47                  186  0.9956 3.19      0.40     9.9
```

```
## 6                  30                   97  0.9951 3.26      0.44    10.1
##   quality
## 1       6
## 2       6
## 3       6
## 4       6
## 5       6
## 6       6
```

3. Lookup Function

This comes down to a simple sort - A message if it is not found - Top record / bottom record - Best in category

This can be solved in a number of ways.

Full Points Create a function that will take at least one parameter of the Vendor. It should return a message if the vendor is not found. It should return the top and bottom vendor for each category (best in category)

Demonstration using Wine Data These statements will use the dplyr package to perform these functions. You can also use the sort() function. This should be dropped into a function with if/then statements.

Pseudo Code: - Lookup vendor. If vendor does not exist return a message - Sort and store the top - Sort and store the bottom - Do a compare to see if they are the top - Return top, bottom, and whether or not they were the top

First look at sorting. This will sort the red wine dataset by alcohol content in a descending order:

```
sorted_red <- arrange(RED_data, desc(alcohol))
sorted_red$generated_uid <- 1:nrow(sorted_red)    # This will create a unique identifier for each recor
head(select(sorted_red, generated_uid, alcohol), 25)
```

```
##    generated_uid  alcohol
## 1              1 14.90000
## 2              2 14.00000
## 3              3 14.00000
## 4              4 14.00000
## 5              5 14.00000
## 6              6 14.00000
## 7              7 14.00000
## 8              8 14.00000
## 9              9 13.60000
## 10            10 13.60000
## 11            11 13.60000
## 12            12 13.60000
## 13            13 13.56667
## 14            14 13.50000
## 15            15 13.40000
## 16            16 13.40000
## 17            17 13.40000
## 18            18 13.30000
## 19            19 13.30000
## 20            20 13.30000
## 21            21 13.20000
## 22            22 13.10000
```

```
## 23            23 13.10000
## 24            24 13.00000
## 25            25 13.00000
```

Next let's do the same but sort in a ascending order

```
sorted_red <- arrange(sorted_red, alcohol)
head(select(sorted_red, generated_uid, alcohol), 25)
```

```
##    generated_uid alcohol
## 1           1598     8.4
## 2           1599     8.4
## 3           1597     8.5
## 4           1595     8.7
## 5           1596     8.7
## 6           1593     8.8
## 7           1594     8.8
## 8           1563     9.0
## 9           1564     9.0
## 10          1565     9.0
## 11          1566     9.0
## 12          1567     9.0
## 13          1568     9.0
## 14          1569     9.0
## 15          1570     9.0
## 16          1571     9.0
## 17          1572     9.0
## 18          1573     9.0
## 19          1574     9.0
## 20          1575     9.0
## 21          1576     9.0
## 22          1577     9.0
## 23          1578     9.0
## 24          1579     9.0
## 25          1580     9.0
```

So at this point, the generated_uid of 1 is the record with the highest alcohol, and that with the record of 1598 is the lowest one. You can then do a comparison to see if the record is included in the dataset. In this example, I would compare against the generated_uid to see if the number is under 1598 (meaning it is in the dataset).

For your function, the key fields are:

- Vendor_Name
- PRP
- ERP

1. First to see if the parameter you pass in matches a Vendor_Name in the dataset. If it does not, return an error message.

2. Sort your data by PRP and capture the top and bottom
3. Sort your data by ERP and capture the top and bottom
4. Do comparisons to see if the vendor name you gave as the parameter matches one of these two. It becomes a simple if-then-else statement

Reviewed for 2020 - MSP