# 36350-A HW5

Joong Ho Choi

TOTAL POINTS

## 60 / 60

QUESTION 1

### 1 Q1 10 / 10

✓ **- 0 pts Correct**

**- 3 pts** did not display dataframe

**- 1 pts** incorrect data types (character for the name column and double for all the other columns)

**- 2 pts** incorrect number of datapoints

**- 2 pts** dataframe was not transposed

**- 2 pts** did not assign column names

**- 2 pts** question links incorrectly

**- 10 pts** blank

QUESTION 2

### 2 Q2 10 / 10

✓ **- 0 pts Correct**

**- 1 pts** did not replace tab symbols (\t) with empty strings

**- 1 pts** did not replace leading spaces with empty strings

**- 1 pts** did not replace trailing spaces with empty strings

**- 1 pts** did not properly clean up col names

**- 3 pts** did not properly retrieve the data (not including the header) from the file

**- 3 pts** did not properly retrieve the header from the file

**- 3 pts** did not display dataframe

**- 2 pts** dataframe displayed in incorrect format (first 4 cols & first 6 rows)

**- 2 pts** question links incorrectly

**- 10 pts** blank

QUESTION 3

### 3 Q3 10 / 10

✓ **- 0 pts Correct**

**- 2 pts** did not fix data types of columns

**- 3 pts** did not display output

**- 3 pts** the displayed output is incorrect

**- 2 pts** does not only display output from later than 2009

**- 2 pts** question links incorrectly

**- 10 pts** blank

QUESTION 4

### 4 Q4 10 / 10

✓ **- 0 pts Correct**

**- 2 pts** did not read in csv file correctly

**- 3 pts** final answer is incorrect

**- 2 pts** question links incorrectly

**- 10 pts** blank

QUESTION 5

### 5 Q5 10 / 10

✓ **- 0 pts Correct**

**- 3 pts** doesn't properly extract variable names

**- 3 pts** doesn't properly extract variable definitions

**- 2 pts** doesn't display dataframe

**- 2 pts** doesn't display head of dataframe

**- 2 pts** question links incorrectly

**- 10 pts** blank

**- 0.5 pts** Did not remove '# COLUMN' from variable names

QUESTION 6

### 6 Q6 10 / 10

✓ **- 0 pts Correct**

**- 3 pts** doesn't display dataframe

**- 2 pts** Incorrect data frame (team names)

**- 2 pts** final dataframe isn't 30 rows

**- 2 pts** dataframe not ordered by decreasing number of wins

- **2 pts** question links incorrectly
- **10 pts** blank

QUESTION 7

## 7 Late Penalty 0 / 0

✓ **- 0 pts** **Correct**

# HW: Week 5

36-350 – Statistical Computing

Week 5 – Spring 2021

Name: Joong Ho Choi

Andrew ID: joonghoc

You must submit **your own** lab as a PDF file on Gradescope.

---

```
suppressWarnings(library(tidyverse))
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.6     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

---

## Question 1

*(10 points)*

An alternative to `read.table()` and such is the `scan()` function. The `scan()` function is *very* handy, particularly when someone gives you weirdly formatted text data files. (Maybe groups of unequal-length rows map to one record, etc., etc.) In this problem, use `scan()` to read in `simple.txt` (which you downloaded for Lab 5) and then post-process what you've read in to create a data frame with correct column names and correct data types (`character` for the `name` column and `double` for all the other columns).

Your final step will be to print out the data frame. Look at the documentation for `scan()` and pay particular attention to the `what` argument.

Once you've scanned the data, use a combination of, e.g., `matrix()` and `data.frame()` to bend the data to your will, and then cast the data in columns 2 through 8 to `numeric`. Hint: `t()` transposes a matrix. Also, pass `stringsAsFactors=FALSE` as an argument to `data.frame()`.

```
# FILL ME IN
a<-scan("/Users/joonghochoi/Desktop/simple.txt",what = "character")
df=data.frame(name=character(),
                 u=double(),
                 g=double(),
                 r=double(),
                 i=double(),
              z=double(),
              y=double(),
              redshift=double(),
                 stringsAsFactors=FALSE)
a=a[9:length(a)]

m<-matrix(a,ncol=9)
ans<-t(m)

res<-rbind(df,ans)
names(res)<-names(df)
res
```

```
##       name       u       g       r       i       z       y redshift
## 1 galaxy.A 17.8313 16.9077 16.4431 16.2099 16.0613 15.8732 0.038356
## 2 galaxy.B 19.0731 17.7448 16.9789 16.5288 16.2551 15.9531 0.058309
## 3 galaxy.C  21.638 21.0106 20.8286 20.6283 20.6552  20.528 0.063701
## 4 galaxy.D 20.5474 19.5542 19.2387 19.0568 19.0887 18.9865 0.059006
## 5 galaxy.E 21.2378 20.6876 20.5661 20.4371 20.4799 20.4503 0.063202
## 6 galaxy.F 22.4627 21.4597 21.0484 20.8274 20.7639 20.6385 0.057773
## 7 galaxy.G 23.8221  22.895 22.5779 22.3543 22.3225 22.2038 0.061548
## 8 galaxy.H 23.0491 22.1536 21.8791 21.6889 21.7044 21.6381 0.063769
## 9 galaxy.I 23.6742 23.0346 22.7857 22.6116 22.5813 22.5462 0.061427
```

## Question 2

*(10 points)*

Let's up the ante a bit here. Download `branch.txt` from the `DATA` directory on Canvas. Examine it with an external viewer. This one's a bit of a mess. (Welcome to real-world data.) Construct a data frame from these data. Assume all the columns are character (there is no need in this exercise to do a final cast of the numeric columns to numeric type).

To read in the data themselves, I'd advise you to use `scan()` while skipping the first line and using "|" as the separator. (See the documentation for `scan()`.) To make the data frame, you could use a combination of `matrix()` and `data.frame()` as in Q1, but before doing do, clean up your strings: replace all tab symbols (`\t`) with empty strings, and replace any leading spaces and trailing spaces with empty strings. (Hint: `gsub()`.) Note that the data comprise 14 columns and 39 rows (not including the header).

Getting the column names is a bit trickier: they are separated by `|_.`, which `scan()` cannot handle. So I'd advise you to use `scan()` to read in *just the first line* (use `\n` as a separator; see the argument `n`), then use `strsplit()` to split the line into 14 column names. You might have to "escape" (i.e., apply double backslashes) some or all of the characters used in splitting. Again, clean things up: get rid of `\t` symbols and trailing spaces.

In the end, display the first four columns and first six rows of your beautiful data frame, rising like a phoenix from the ashes of the terribly formatted ASCII file that you began with.

1 Q1 **10 / 10**

✓ **- 0 pts** **Correct**

**- 3 pts** did not display dataframe

**- 1 pts** incorrect data types (character for the name column and double for all the other columns)

**- 2 pts** incorrect number of datapoints

**- 2 pts** dataframe was not transposed

**- 2 pts** did not assign column names

**- 2 pts** question links incorrectly

**- 10 pts** blank

```
# FILL ME IN
a<-scan("/Users/joonghochoi/Desktop/simple.txt",what = "character")
df=data.frame(name=character(),
                 u=double(),
                 g=double(),
                 r=double(),
                 i=double(),
              z=double(),
              y=double(),
              redshift=double(),
                 stringsAsFactors=FALSE)
a=a[9:length(a)]

m<-matrix(a,ncol=9)
ans<-t(m)

res<-rbind(df,ans)
names(res)<-names(df)
res
```

```
##       name       u       g       r       i       z       y redshift
## 1 galaxy.A 17.8313 16.9077 16.4431 16.2099 16.0613 15.8732 0.038356
## 2 galaxy.B 19.0731 17.7448 16.9789 16.5288 16.2551 15.9531 0.058309
## 3 galaxy.C  21.638 21.0106 20.8286 20.6283 20.6552  20.528 0.063701
## 4 galaxy.D 20.5474 19.5542 19.2387 19.0568 19.0887 18.9865 0.059006
## 5 galaxy.E 21.2378 20.6876 20.5661 20.4371 20.4799 20.4503 0.063202
## 6 galaxy.F 22.4627 21.4597 21.0484 20.8274 20.7639 20.6385 0.057773
## 7 galaxy.G 23.8221  22.895 22.5779 22.3543 22.3225 22.2038 0.061548
## 8 galaxy.H 23.0491 22.1536 21.8791 21.6889 21.7044 21.6381 0.063769
## 9 galaxy.I 23.6742 23.0346 22.7857 22.6116 22.5813 22.5462 0.061427
```

## Question 2

*(10 points)*

Let's up the ante a bit here. Download `branch.txt` from the `DATA` directory on Canvas. Examine it with an external viewer. This one's a bit of a mess. (Welcome to real-world data.) Construct a data frame from these data. Assume all the columns are character (there is no need in this exercise to do a final cast of the numeric columns to numeric type).

To read in the data themselves, I'd advise you to use `scan()` while skipping the first line and using "|" as the separator. (See the documentation for `scan()`.) To make the data frame, you could use a combination of `matrix()` and `data.frame()` as in Q1, but before doing do, clean up your strings: replace all tab symbols (`\t`) with empty strings, and replace any leading spaces and trailing spaces with empty strings. (Hint: `gsub()`.) Note that the data comprise 14 columns and 39 rows (not including the header).

Getting the column names is a bit trickier: they are separated by `|_.`, which `scan()` cannot handle. So I'd advise you to use `scan()` to read in *just the first line* (use `\n` as a separator; see the argument `n`), then use `strsplit()` to split the line into 14 column names. You might have to "escape" (i.e., apply double backslashes) some or all of the characters used in splitting. Again, clean things up: get rid of `\t` symbols and trailing spaces.

In the end, display the first four columns and first six rows of your beautiful data frame, rising like a phoenix from the ashes of the terribly formatted ASCII file that you began with.

```
# FILL ME IN
b<-scan("/Users/joonghochoi/Desktop/branch.txt",what="character",sep ="|",skip=TRUE)
c=sapply(b,function(x){gsub("[[:space:]]*$","",x)}) #remove trailing spaces
c=sapply(c,function(x){gsub("^[[:space:]]{1,3}","",x)}) #remove leading spaces
c=sapply(c,function(x){gsub("\t","",x)}) #remove tab
m=matrix(c,nrow=14,ncol=39)

df<-data.frame(t(m))

d<-scan("/Users/joonghochoi/Desktop/branch.txt",what="character",sep ="\n")
first_line=d[1]
first_line=unlist(strsplit(first_line,"\\|{1}\\_{1}\\.{1}"))

e = gsub("[[:space:]]*$","",first_line)
e = gsub("^[[:space:]]*","",e)
e = gsub("\t","",e)

names(df)=e
df[1:6,1:4]
```

```
##    Subm_ID Score Sigma_s Detection_image
## 1 A_SP_0.0  80.9    0.25              No
## 2 A_SP_0.1 100.3    0.25              No
## 3 A_SP_0.4 579.8     1.0              No
## 4 A_SP_1.0 120.4    0.25              No
## 5 A_SP_1.7  78.5    10.0              No
## 6 A_SP_1.9 939.1     1.0              No
```

## Question 3

*(10 points)*

Read in data from `https://download.bls.gov/pub/time.series/ap/ap.data.0.Current`, which are housed at the Bureau of Labor Statistics. Note before you start that the data are *tab delimited*, and you might find it helpful to remember that a tab is denoted `\t` in a string. The data may not read in cleanly with a simple function call; you may need to skip the header, in which case you will need to provide column names yourself.

Also, the parser may misidentify column types, so you may have to set those too. And. . . you may have to cast data in some columns to be of proper type, after the reading in of the data is done. (Data wrangling is a messy business.) Once everything is read in and cast to (if necessary) proper type, display the mean and standard deviation of the data in the value column for every year *after* 2009 (i.e., 2010 and later). The tidyverse will help you here. Hint: `group_by()`.

```
# FILL ME IN
library(tidyverse)
library(dplyr)
url = read.delim("https://download.bls.gov/pub/time.series/ap/ap.data.0.Current")
data<-url%>%filter(year>2009)%>%group_by(year)
data$value<-as.numeric(data$value)
```

```
## Warning: NAs introduced by coercion
```

## 2 Q2 10 / 10

✓ **- 0 pts** **Correct**

- **1 pts** did not replace tab symbols (\t) with empty strings
- **1 pts** did not replace leading spaces with empty strings
- **1 pts** did not replace trailing spaces with empty strings
- **1 pts** did not properly clean up col names
- **3 pts** did not properly retrieve the data (not including the header) from the file
- **3 pts** did not properly retrieve the header from the file
- **3 pts** did not display dataframe
- **2 pts** dataframe displayed in incorrect format (first 4 cols & first 6 rows)
- **2 pts** question links incorrectly
- **10 pts** blank

```
# FILL ME IN
b<-scan("/Users/joonghochoi/Desktop/branch.txt",what="character",sep ="|",skip=TRUE)
c=sapply(b,function(x){gsub("[[:space:]]*$","",x)}) #remove trailing spaces
c=sapply(c,function(x){gsub("^[[:space:]]{1,3}","",x)}) #remove leading spaces
c=sapply(c,function(x){gsub("\t","",x)}) #remove tab
m=matrix(c,nrow=14,ncol=39)

df<-data.frame(t(m))

d<-scan("/Users/joonghochoi/Desktop/branch.txt",what="character",sep ="\n")
first_line=d[1]
first_line=unlist(strsplit(first_line,"\\|{1}\\_{1}\\.{1}"))

e = gsub("[[:space:]]*$","",first_line)
e = gsub("^[[:space:]]*","",e)
e = gsub("\t","",e)

names(df)=e
df[1:6,1:4]
```

```
##    Subm_ID Score Sigma_s Detection_image
## 1 A_SP_0.0  80.9    0.25              No
## 2 A_SP_0.1 100.3    0.25              No
## 3 A_SP_0.4 579.8     1.0              No
## 4 A_SP_1.0 120.4    0.25              No
## 5 A_SP_1.7  78.5    10.0              No
## 6 A_SP_1.9 939.1     1.0              No
```

## Question 3

*(10 points)*

Read in data from `https://download.bls.gov/pub/time.series/ap/ap.data.0.Current`, which are housed at the Bureau of Labor Statistics. Note before you start that the data are *tab delimited*, and you might find it helpful to remember that a tab is denoted `\t` in a string. The data may not read in cleanly with a simple function call; you may need to skip the header, in which case you will need to provide column names yourself.

Also, the parser may misidentify column types, so you may have to set those too. And. . . you may have to cast data in some columns to be of proper type, after the reading in of the data is done. (Data wrangling is a messy business.) Once everything is read in and cast to (if necessary) proper type, display the mean and standard deviation of the data in the value column for every year *after* 2009 (i.e., 2010 and later). The tidyverse will help you here. Hint: `group_by()`.

```
# FILL ME IN
library(tidyverse)
library(dplyr)
url = read.delim("https://download.bls.gov/pub/time.series/ap/ap.data.0.Current")
data<-url%>%filter(year>2009)%>%group_by(year)
data$value<-as.numeric(data$value)
```

```
## Warning: NAs introduced by coercion
```

```
data$footnote_codes<-as.character(data$footnote_codes)
mean<-data%>%group_by(year)%>%summarize(mn=mean(value,na.rm=TRUE))
sdv<-data%>%group_by(year)%>%summarize(sd=sd(value,na.rm=TRUE))
mean
```

```
## # A tibble: 12 x 2
##     year    mn
##  * <int> <dbl>
##  1  2010 14.8
##  2  2011 15.0
##  3  2012 14.5
##  4  2013  9.66
##  5  2014  3.07
##  6  2015  2.81
##  7  2016  2.61
##  8  2017  2.71
##  9  2018  2.75
## 10  2019  2.70
## 11  2020  2.64
## 12  2021  2.75
```

```
sdv
```

```
## # A tibble: 12 x 2
##     year    sd
##  * <int> <dbl>
##  1  2010 30.2
##  2  2011 29.7
##  3  2012 27.9
##  4  2013 22.5
##  5  2014  1.90
##  6  2015  2.00
##  7  2016  1.90
##  8  2017  1.89
##  9  2018  1.83
## 10  2019  1.81
## 11  2020  1.93
## 12  2021  1.99
```

```
#
```

## Question 4

*(10 points)*

Download `planets.csv` from the Canvas site. It is in the Week 7 directory. Use an external viewer (your choice) to look at the file. Then apply an appropriate function to read the file's contents into `R`. Your goal: to determine what proportion of the columns have data in at least 20% of their rows. (In other words, step from column to column and see if the proportion of `NA`'s is less than 80%. Then determine the proportion of the columns that fulfill this condition.) Your final answer should be 82.86% [or 0.8286].

### 3 Q3 10 / 10

✓ **- 0 pts** **Correct**

**- 2 pts** did not fix data types of columns

**- 3 pts** did not display output

**- 3 pts** the displayed output is incorrect

**- 2 pts** does not only display output from later than 2009

**- 2 pts** question links incorrectly

**- 10 pts** blank

```
data$footnote_codes<-as.character(data$footnote_codes)
mean<-data%>%group_by(year)%>%summarize(mn=mean(value,na.rm=TRUE))
sdv<-data%>%group_by(year)%>%summarize(sd=sd(value,na.rm=TRUE))
mean
```

```
## # A tibble: 12 x 2
##     year    mn
##  * <int> <dbl>
##  1  2010 14.8
##  2  2011 15.0
##  3  2012 14.5
##  4  2013  9.66
##  5  2014  3.07
##  6  2015  2.81
##  7  2016  2.61
##  8  2017  2.71
##  9  2018  2.75
## 10  2019  2.70
## 11  2020  2.64
## 12  2021  2.75
```

```
sdv
```

```
## # A tibble: 12 x 2
##     year    sd
##  * <int> <dbl>
##  1  2010 30.2
##  2  2011 29.7
##  3  2012 27.9
##  4  2013 22.5
##  5  2014  1.90
##  6  2015  2.00
##  7  2016  1.90
##  8  2017  1.89
##  9  2018  1.83
## 10  2019  1.81
## 11  2020  1.93
## 12  2021  1.99
```

```
#
```

## Question 4

*(10 points)*

Download `planets.csv` from the Canvas site. It is in the Week 7 directory. Use an external viewer (your choice) to look at the file. Then apply an appropriate function to read the file's contents into `R`. Your goal: to determine what proportion of the columns have data in at least 20% of their rows. (In other words, step from column to column and see if the proportion of `NA`'s is less than 80%. Then determine the proportion of the columns that fulfill this condition.) Your final answer should be 82.86% [or 0.8286].

```
# FILL ME IN
data<-read.csv("/Users/joonghochoi/Desktop/planets.csv",skip=73)
df<-as.data.frame(data)
v<-names(data)
col_count=length(v)
r=nrow(data)
count=0
for (ii in 1:length(v)){
  k=v[ii]
  if (sum(is.na(df[k]))/r < 0.8){
    count=count+1
  }
}
res=count/col_count
res
```

```
## [1] 0.8285714
```

## Question 5

*(10 points)*

Make a data frame that is in essence a "dictionary" for the data in the `planets.csv` file. What this means is: extract those lines of the file that contain variable names and corresponding definitions, and from those lines extract the variable names into a vector called `variable` and the definitions into a vector called `definition`.

Output the first six rows only! (Hint: in your call to `data.frame()`, set the argument `stringsAsFactors` to `FALSE`. This changes the column contents to character strings rather than factor variables.)

Hint: let's say you do an `strsplit()` to split the variable from the definition in each line. The output will be a list, with one list element for each line that contains two strings, one for the variable and one for the definition. A handy way to extract all of the variables would be, e.g., sapply(<output from strplit>,`[[`,1). That `[[` function is really useful.

```
# FILL ME IN

data<-read.csv("/Users/joonghochoi/Desktop/planets.csv",skip=2,nrows=69)
ans<-sapply(data,function(x){strsplit(x,":")})
variable=c()
definition=c()
for (ii in 1:length(ans)){
  d=gsub("^#{1}[[:space:]]{1}C{1}O{1}L{1}U{1}M{1}N{1}","",ans[[ii]][[1]])
  variable=c(variable,d)
  k= gsub("^[[:space:]]*","",ans[[ii]][[2]])
  definition=c(definition,k)
}
df=data.frame(variable,definition,stringsAsFactors=FALSE)
head(df,6)
```

```
##         variable                       definition
## 1    pl_hostname                        Host Name
## 2      pl_letter                     Planet Letter
## 3  pl_discmethod                  Discovery Method
## 4        pl_pnum      Number of Planets in System
```

## 4 Q4 10 / 10

✓ **- 0 pts** **Correct**

**- 2 pts** did not read in csv file correctly

**- 3 pts** final answer is incorrect

**- 2 pts** question links incorrectly

**- 10 pts** blank

```
# FILL ME IN
data<-read.csv("/Users/joonghochoi/Desktop/planets.csv",skip=73)
df<-as.data.frame(data)
v<-names(data)
col_count=length(v)
r=nrow(data)
count=0
for (ii in 1:length(v)){
  k=v[ii]
  if (sum(is.na(df[k]))/r < 0.8){
    count=count+1
  }
}
res=count/col_count
res
```

```
## [1] 0.8285714
```

## Question 5

*(10 points)*

Make a data frame that is in essence a "dictionary" for the data in the `planets.csv` file. What this means is: extract those lines of the file that contain variable names and corresponding definitions, and from those lines extract the variable names into a vector called `variable` and the definitions into a vector called `definition`.

Output the first six rows only! (Hint: in your call to `data.frame()`, set the argument `stringsAsFactors` to `FALSE`. This changes the column contents to character strings rather than factor variables.)

Hint: let's say you do an `strsplit()` to split the variable from the definition in each line. The output will be a list, with one list element for each line that contains two strings, one for the variable and one for the definition. A handy way to extract all of the variables would be, e.g., sapply(<output from strplit>,`[[`,1). That `[[` function is really useful.

```
# FILL ME IN

data<-read.csv("/Users/joonghochoi/Desktop/planets.csv",skip=2,nrows=69)
ans<-sapply(data,function(x){strsplit(x,":")})
variable=c()
definition=c()
for (ii in 1:length(ans)){
  d=gsub("^#{1}[[:space:]]{1}C{1}O{1}L{1}U{1}M{1}N{1}","",ans[[ii]][[1]])
  variable=c(variable,d)
  k= gsub("^[[:space:]]*","",ans[[ii]][[2]])
  definition=c(definition,k)
}
df=data.frame(variable,definition,stringsAsFactors=FALSE)
head(df,6)
```

```
##         variable                       definition
## 1    pl_hostname                        Host Name
## 2      pl_letter                     Planet Letter
## 3  pl_discmethod                  Discovery Method
## 4        pl_pnum      Number of Planets in System
```

```
## 5      pl_orbper            Orbital Period [days]
## 6  pl_orbpererr1 Orbital Period Upper Unc. [days]
```

## Question 6

*(10 points)*

Extract the 2020 Major League Baseball standings from the web site given below and put them into a *single* data frame that contains all 30 MLB teams, with the first column being the team name, the second column being the number of wins, and the third column being the number of losses. Order the data frame by decreasing number of wins. Use `rvest` functions to extract any tables you need, which are of class `data.frame`, and then process the data frames until you get a single one as described above.

```
library(dplyr)
if ( require(rvest) == FALSE ) {
  install.packages("rvest",repos="https://cloud.r-project.org")
  library(rvest)
}
```

```
## Loading required package: rvest
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:purrr':
##
##     pluck
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
site = read_html("https://www.baseball-reference.com/leagues/MLB-standings.shtml")
df=site %>% html_nodes("table")%>%.[1:6]%>%html_table()

Tm=c()
W=c()
L=c()

for (ii in 1:6){
  data=df[[ii]]
  Tm=append(Tm,data$Tm)
  W=append(W,data$W)
  L=append(L,data$L)
}

ans<-data.frame(Tm,W,L)
res<-ans%>%arrange(desc(W))
res
```

## 5 Q5 10 / 10

✓ **- 0 pts** **Correct**

**- 3 pts** doesn’t properly extract variable names

**- 3 pts** doesn’t properly extract variable definitions

**- 2 pts** doesn’t display dataframe

**- 2 pts** doesn’t display head of dataframe

**- 2 pts** question links incorrectly

**- 10 pts** blank

**- 0.5 pts** Did not remove '# COLUMN' from variable names

```
## 5      pl_orbper            Orbital Period [days]
## 6  pl_orbpererr1 Orbital Period Upper Unc. [days]
```

## Question 6

*(10 points)*

Extract the 2020 Major League Baseball standings from the web site given below and put them into a *single* data frame that contains all 30 MLB teams, with the first column being the team name, the second column being the number of wins, and the third column being the number of losses. Order the data frame by decreasing number of wins. Use `rvest` functions to extract any tables you need, which are of class `data.frame`, and then process the data frames until you get a single one as described above.

```
library(dplyr)
if ( require(rvest) == FALSE ) {
  install.packages("rvest",repos="https://cloud.r-project.org")
  library(rvest)
}
```

```
## Loading required package: rvest
```

```
## Loading required package: xml2
```

```
##
## Attaching package: 'rvest'
```

```
## The following object is masked from 'package:purrr':
##
##     pluck
```

```
## The following object is masked from 'package:readr':
##
##     guess_encoding
```

```
site = read_html("https://www.baseball-reference.com/leagues/MLB-standings.shtml")
df=site %>% html_nodes("table")%>%.[1:6]%>%html_table()

Tm=c()
W=c()
L=c()

for (ii in 1:6){
  data=df[[ii]]
  Tm=append(Tm,data$Tm)
  W=append(W,data$W)
  L=append(L,data$L)
}

ans<-data.frame(Tm,W,L)
res<-ans%>%arrange(desc(W))
res
```

```
##                       Tm  W  L
## 1    Los Angeles Dodgers 43 17
## 2         Tampa Bay Rays 40 20
## 3       San Diego Padres 37 23
## 4        Minnesota Twins 36 24
## 5      Oakland Athletics 36 24
## 6      Chicago White Sox 35 25
## 7      Cleveland Indians 35 25
## 8         Atlanta Braves 35 25
## 9           Chicago Cubs 34 26
## 10      New York Yankees 33 27
## 11     Toronto Blue Jays 32 28
## 12         Miami Marlins 31 29
## 13       Cincinnati Reds 31 29
## 14   St. Louis Cardinals 30 28
## 15        Houston Astros 29 31
## 16     Milwaukee Brewers 29 31
## 17  San Francisco Giants 29 31
## 18 Philadelphia Phillies 28 32
## 19      Seattle Mariners 27 33
## 20    Kansas City Royals 26 34
## 21    Los Angeles Angels 26 34
## 22         New York Mets 26 34
## 23  Washington Nationals 26 34
## 24      Colorado Rockies 26 34
## 25     Baltimore Orioles 25 35
## 26  Arizona Diamondbacks 25 35
## 27        Boston Red Sox 24 36
## 28        Detroit Tigers 23 35
## 29         Texas Rangers 22 38
## 30    Pittsburgh Pirates 19 41
```

### 6 Q6 10 / 10

✓ **- 0 pts** **Correct**

**- 3 pts** doesn't display dataframe

**- 2 pts** Incorrect data frame (team names)

**- 2 pts** final dataframe isn't 30 rows

**- 2 pts** dataframe not ordered by decreasing number of wins

**- 2 pts** question links incorrectly

**- 10 pts** blank

### 7 Late Penalty 0 / 0

✓ - 0 pts Correct