Data Hacking with RDSTK 3

RDSTK is a very versatile package. It includes functions to help you convert IP address to geo locations and derive statistics from them. It also allows you to input a body of text and convert it into sentiments.

This is a continuation from the last exercise RDSTK 2
We are going to use the function that we created in our last exercise to have a programmatic way to derive statistics using the coordinates2statistics() function. Last week we talked about local and global variables. This is important to understand before proceeding. Also refresh on ip2coordinates() function.

This package provides an R interface to Pete Warden's Data Science Toolkit. See for more information click here.

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
This week we will give you bigger and badder list to work with. Its a list of more a dozen proxy ip-addresses from the internet. Run the code

```
list=c("97.77.104.22","104.199.228.65","50.93.204.169","107.189.46.5","104.154.142.10","104.131.255.12","209.212.253.44","70.248.28.23","52.119.20.75","192.169.168.15","47.88.31.7580","107.178.4.109","152.160.35.171","104.236.54.196","50.93.1
```

97.102","159.203.117.1","206.125.41.132","50.93.201.28","8.21.67.248 31","104.28.16.199")

Exercise 2

Remember how we used iterators to run through each location and derive the stats with ip2coordinates() function in the first rdstk exercise. Lets do the same here. Store the results in df

Exercise 3

If you came this far, great. Lets recall the function that we created in exercise 2. If you do not remember the function, here is the code for it. Run the code below and then run stat_maker("population_density"). You should see a new column called pop

```
stat_maker=function(s2){
s1="statistics"
s3="value"
s2=as.character(s2)
for (i in 1:nrow(df)) {
df$pop[i]                                                    <<-
coordinates2statistics(df[i,3],df[i,6],s2)[paste(s1,s2,s3, sep = ".")]
assign("test2",50,envir = .GlobalEnv)

}
}
```

You should see an output in the format "statistics.hello.value"

Exercise 4

Modify the function so that the function accepts a string and returns out a global variable that holds the elements of that

string statistic. For example if you input elevation, the function will create a global variable called elevation with the results from the for loop stored

Exercise 5

Test out the function.

```
stat_maker("elevation")
```

Exercise 6

Test the function stat_maker. stat_maker(“population_density”). Notice it did not explicitly make the changes to the df but just returned it once you called the function. This is because we did not define df as a global variable. But thats okay. We will learn it later

Exercise 7

Great. Now before we modify our function, lets learn how we can make a global variable inside a function. Use the same code from exercise 5 but this time instead of defining df$pop2 as a local variable, define it as a global variable. Run the function and test it again.

Exercise 8

Run the code

```
stat_maker("us_population_poverty")
```

Notice that our function does not work for this case. This is because anything with the prefix us_population will return a dataframe with a column value like statistics.us_population.value
So you need to modify the function a little to accomodate for this.

Exercise 9

Run the following commands. You can also use any string starting with us_population fo this function. But the goal is to make global variables that hold this data. You can refer to the whole list of statistic funtions at www.datasciencetoolkit.org

```
stat_maker("us_population")
stat_maker("us_population_poverty")
stat_maker("us_population_asian")
stat_maker("us_population_bachelors_degree")
stat_maker("us_population_black_or_african_american")
stat_maker("us_population_black_or_african_american_not_hispanic ")
stat_maker("us_population_eighteen_to_twenty_four_years_old")
stat_maker("us_population_five_to_seventeen_years_old")
stat_maker("us_population_foreign_born")
stat_maker("us_population_hispanic_or_latino")
```

Exercise 10

Use cbind command to bind all the global variables into the df. Print the results of df.

Note: You can chose to make this df in other ways but this method was used to guide through modifying functions, global/local variables and working with strings.

Data Hacking with RDSTK solution 3

Below are the solutions to these exercises on the RDSTK package

```
###############
#             #
# Exercise 1  #
#             #
###############
list=c("97.77.104.22","104.199.228.65","50.93.204.169","107.18
9.46.5","104.154.142.10","104.131.255.12","209.212.253.44","70
.248.28.23","52.119.20.75","192.169.168.15","47.88.31.75
80","107.178.4.109","152.160.35.171","104.236.54.196","50.93.1
97.102","159.203.117.1","206.125.41.132","50.93.201.28","8.21.
67.248  31","104.28.16.199")

###############
#             #
# Exercise 2  #
#             #
###############

df=data.frame(list)
df[,1]=as.character(df[,1])
data=lapply(df[,1],ip2coordinates)
df=do.call(rbind.data.frame,data)
df


##        ip.address dma_code latitude country_code3 area_code
longitude
## 1    97.77.104.22      641  29.4717           USA       210
-98.5140
## 2   50.93.204.169      539  28.0499           USA       813
-82.3625
## 3  209.212.253.44      515  39.0705           USA       513
-84.2803
## 4    70.248.28.23      641  29.3171           USA       210
-98.5555
## 5    52.119.20.75      504  39.5645           USA       302
-75.5970
## 6  152.160.35.171      505  42.4634           USA       248
-83.4646
## 7   50.93.197.102      539  28.0499           USA       813
-82.3625
## 8   159.203.117.1      505  42.6644           USA       248
```

```
-83.2303
## 9  206.125.41.132      803  34.0530          USA       213
-118.2642
## 10   50.93.201.28      539  28.0499          USA       813
-82.3625
##       country_name postal_code region       locality
country_code
## 1   United States                  TX   San Antonio
US
## 2   United States        33637     FL         Tampa
US
## 3   United States        45245     OH    Cincinnati
US
## 4   United States        78224     TX   San Antonio
US
## 5   United States        19893     DE    Wilmington
US
## 6   United States        48375     MI          Novi
US
## 7   United States        33637     FL         Tampa
US
## 8   United States        48326     MI Auburn Hills
US
## 9   United States        90017     CA   Los Angeles
US
## 10 United States        33637     FL         Tampa
US


###############
#             #
# Exercise 3  #
#             #
###############

stat_maker=function(s2){
   s1="statistics"
   s3="value"
   s2=as.character(s2)
   for (i in 1:nrow(df)) {
                                          df$pop[i]       <-
coordinates2statistics(df[i,3],df[i,6],s2)[paste(s1,s2,s3, sep
```

```
= ".")]
     assign("test2",50,envir = .GlobalEnv)


   }
 }

 stat_maker("population_density")


###############
#             #
# Exercise 4  #
#             #
###############

 stat_maker=function(s2){
   s1="statistics"
   s3="value"
   s2=as.character(s2)
 for (i in 1:nrow(df)) {
a[i]=coordinates2statistics(df[i,3],df[i,6],s2)[paste(s1,s2,s3
, sep = ".")]
   a=unlist(a)
   assign(paste(s2),a,envir = .GlobalEnv)


 }
 }


###############
#             #
# Exercise 5  #
#             #
###############

stat_maker("elevation")
elevation

##  [1] 214  16 278 189   1 266  16 278 116  16
```

```
###############
#             #
# Exercise 6  #
#             #
###############

stat_maker("population_density")

###############
#             #
# Exercise 7  #
#             #
###############
 stat_maker=function(s2){
   s1="statistics"
   s3="value"
   s2=as.character(s2)
   for (i in 1:nrow(df)) {
                                          df$pop2[i]       <-
coordinates2statistics(df[i,4],df[i,7],s2)[paste(s1,s2,s3, sep
= ".")]



   }
 }

 stat_maker("population_density")


###############
#             #
# Exercise 8  #
#             #
###############

 stat_maker=function(s2){
   s1="statistics"
   s3="value"
   s2=as.character(s2)
   for (i in 1:nrow(df)) {
```

```
a[i]=coordinates2statistics(df[i,3],df[i,6],s2)[paste(s1,"us_p
opulation",s3, sep = ".")]
     a=unlist(a)
     assign(paste(s2),a,envir = .GlobalEnv)
   }
 }


###############
#             #
# Exercise 9  #
#             #
###############
stat_maker("us_population")
stat_maker("us_population_poverty")
stat_maker("us_population_asian")
stat_maker("us_population_bachelors_degree")
stat_maker("us_population_black_or_african_american")
stat_maker("us_population_black_or_african_american_not_hispan
ic ")
stat_maker("us_population_eighteen_to_twenty_four_years_old")
stat_maker("us_population_five_to_seventeen_years_old")
stat_maker("us_population_foreign_born")
stat_maker("us_population_hispanic_or_latino")

###############
#             #
# Exercise 10  #
#             #
###############


df=cbind(df,elevation,us_population,us_population_asian,us_pop
ulation_bachelors_degree,us_population_black_or_african_americ
an,us_population_eighteen_to_twenty_four_years_old,us_populati
on_five_to_seventeen_years_old,us_population_foreign_born,us_p
opulation_hispanic_or_latino,us_population_poverty)

df

##        ip.address dma_code latitude country_code3 area_code
```

```
longitude
## 1    97.77.104.22      641  29.4717          USA       210
-98.5140
## 2   50.93.204.169      539  28.0499          USA       813
-82.3625
## 3  209.212.253.44      515  39.0705          USA       513
-84.2803
## 4    70.248.28.23      641  29.3171          USA       210
-98.5555
## 5    52.119.20.75      504  39.5645          USA       302
-75.5970
## 6  152.160.35.171      505  42.4634          USA       248
-83.4646
## 7   50.93.197.102      539  28.0499          USA       813
-82.3625
## 8   159.203.117.1      505  42.6644          USA       248
-83.2303
## 9  206.125.41.132      803  34.0530          USA       213
-118.2642
## 10   50.93.201.28      539  28.0499          USA       813
-82.3625
##       country_name postal_code region       locality
country_code   pop
## 1  United States                  TX  San Antonio
US  3194
## 2  United States       33637      FL         Tampa
US   671
## 3  United States       45245      OH    Cincinnati
US   498
## 4  United States       78224      TX  San Antonio
US   355
## 5  United States       19893      DE    Wilmington
US   188
## 6  United States       48375      MI          Novi
US   610
## 7  United States       33637      FL         Tampa
US   671
## 8  United States       48326      MI Auburn Hills
US   580
## 9  United States       90017      CA  Los Angeles
US 13701
```

```
## 10 United States        33637     FL         Tampa
US   671
##                                        pop2 elevation
us_population
## 1   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0         214
1768
## 2   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0          16
360
## 3   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0         278
547
## 4   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0         189
593
## 5   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0           1
143
## 6   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0         266
793
## 7   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0          16
360
## 8   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0         278
0
## 9   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0         116
5230
## 10 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0          16
360
##    us_population_asian us_population_bachelors_degree
## 1                 1768                          1768
## 2                  360                           360
## 3                  547                           547
## 4                  593                           593
## 5                  143                           143
## 6                  793                           793
## 7                  360                           360
## 8                    0                             0
## 9                 5230                          5230
## 10                 360                           360
##    us_population_black_or_african_american
## 1                                     1768
## 2                                      360
## 3                                      547
## 4                                      593
## 5                                      143
```

```
## 6                                          793
## 7                                          360
## 8                                            0
## 9                                         5230
## 10                                         360
##    us_population_eighteen_to_twenty_four_years_old
## 1                                             1768
## 2                                              360
## 3                                              547
## 4                                              593
## 5                                              143
## 6                                              793
## 7                                              360
## 8                                                0
## 9                                             5230
## 10                                             360
##          us_population_five_to_seventeen_years_old
us_population_foreign_born
## 1                                             1768
1768
## 2                                              360
360
## 3                                              547
547
## 4                                              593
593
## 5                                              143
143
## 6                                              793
793
## 7                                              360
360
## 8                                                0
0
## 9                                             5230
5230
## 10                                             360
360
##    us_population_hispanic_or_latino us_population_poverty
## 1                              1768                  1768
## 2                               360                   360
```

```
## 3                547                547
## 4                593                593
## 5                143                143
## 6                793                793
## 7                360                360
## 8                  0                  0
## 9               5230               5230
## 10               360                360
```

Data Hacking with RDSTK 2

RDSTK is a very versatile package. It includes functions to help you convert IP address to geo locations and derive statistics from them. It also allows you to input a body of text and convert it into sentiments.

This is a continuation from the last exercise RDSTK 1
This package provides an R interface to Pete Warden's Data Science Toolkit. See www.datasciencetoolkit.org for more information.

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
Load the rdstk library and download the dataset here

Exercise 2

a.create a string called s1 and store "statistics" inside

b.create a string called s3 and store “value”
c. create a function that will take a string s2 as an input and output a string in the format s1+s2+s3 seperated by “.”. Name this function “stringer”

Exercise 3

Lets test out this function.

```
stringer("hello")
```

You should see an output in the format “statistics.hello.value”

Exercise 4

Create a for loop that will iterate over the rows in df and derive the population density of the location using coordinates2statistics function. Save the results in df$pop

Exercise 5

Lets now make a function using elements you learned from exercise 3 and 4. So the function is going to take a string as an input like s2 from exercise 3. Inside the function you can combine it with s1 and s3. You have to create the same for loop from exercise 4. Instead of storing the result of the for loop in df$pop, use df$pop2.You should see a new feature inside df with all the results once you return df from it.

Exercise 6

Test the function stat_maker. stat_maker("population_density"). Notice it did not explicitly make the changes to the df but just returned it once you called the function. This is because we did not define df as a global variable. But thats okay. We will learn it later

Exercise 7

Great. Now before we modify our function, lets learn how we can make a global variable inside a function. Use the same code from exercise 5 but this time instead of defining df$pop2 as a local variable, define it as a global variable. Run the function and test it again.

Exercise 8

You can also use the assign() function inside a function and set the results as a global variable. Lets see an example of assign function

```
assign("test",50)
```

Now if you type test in your console. You should see 50. Try it

Exercise 9

Now try putting the same code in exercise 8 while changing test to test2 inside the stat_maker function. Once you test the function, you will see that test2 does not return anything. This is because it was not set as a global variable

Exercise 10

Set test2 as a global variable inside the stat_maker function. Run the function and now you should see test 2 return 50 when you call it.

Data Hacking with RDSTK 2 solution

Below are the solutions to these exercises on sorting and ordering.

```
###############
#             #
# Exercise 1  #
#             #
###############

library(RDSTK)


###############
#             #
# Exercise 2  #
#             #
###############

 s1="statistics"
  s3="value"
  stringer= function(s2){
   s1="statistics"
   s3="value"
   s2=as.character(s2)
   return(paste(s1,s2,s3, sep = "."))
 }

###############
#             #
# Exercise 3  #
#             #
###############

stringer("hello")
```

```
## [1] "statistics.hello.value"

###############
#             #
# Exercise 4  #
#             #
###############

for (i in 1:nrow(df)) {
df$pop[i]=coordinates2statistics(df[i,4],df[i,7],"population_d
ensity")["statistics.population_density.value"]
}

df

##    X       ip.address dma_code latitude country_code3
area_code longitude
## 1 1 165.124.145.197      602  42.0586           USA
847  -87.6845
## 2 2    31.24.74.155        0  57.1000           SWE
0   12.2500
## 3 3   79.129.19.173        0  39.7378           GRC
0   22.2892
##    country_name postal_code region locality country_code
pop pop2
## 1 United States       60208     IL Evanston           US
3040 3040
## 2        Sweden          NA     06  Varberg           SE
60   60
## 3        Greece          NA     21 Tírnavos           GR
33   33

###############
#             #
# Exercise 5  #
#             #
###############

 stat_maker=function(s2){
   s1="statistics"
   s3="value"
```

```
   s2=as.character(s2)
 for (i in 1:nrow(df)) {
df$pop2[i]=coordinates2statistics(df[i,4],df[i,7],s2)[paste(s1
,s2,s3, sep = ".")]
 }
   return(df)}

###############
#             #
# Exercise 6  #
#             #
###############

stat_maker("population_density")

##   X       ip.address dma_code latitude country_code3
area_code longitude
## 1 1 165.124.145.197      602  42.0586           USA
847  -87.6845
## 2 2    31.24.74.155        0  57.1000           SWE
0   12.2500
## 3 3   79.129.19.173        0  39.7378           GRC
0   22.2892
##    country_name postal_code region locality country_code
pop pop2
## 1 United States       60208     IL Evanston           US
3040 3040
## 2        Sweden          NA     06  Varberg           SE
60   60
## 3        Greece          NA     21 Tírnavos           GR
33   33

###############
#             #
# Exercise 7  #
#             #
###############
 stat_maker=function(s2){
   s1="statistics"
   s3="value"
   s2=as.character(s2)
```

```
   for (i in 1:nrow(df)) {
                                          df$pop2[i]       <-
coordinates2statistics(df[i,4],df[i,7],s2)[paste(s1,s2,s3, sep
= ".")]



   }
 }

 stat_maker("population_density")


###############
#             #
# Exercise 8  #
#             #
###############

assign("test",50)

test

## [1] 50

###############
#             #
# Exercise 9  #
#             #
###############

stat_maker=function(s2){
   s1="statistics"
   s3="value"
   s2=as.character(s2)
   for (i in 1:nrow(df)) {
                                          df$pop2[i]       <-
coordinates2statistics(df[i,4],df[i,7],s2)[paste(s1,s2,s3, sep
= ".")]
     assign("test2",50)
```

```
   }
 }
 stat_maker("population_density")
test2
```

```
## Error in eval(expr, envir, enclos): object 'test2' not
found
```

```
###############
#             #
# Exercise 10  #
#             #
###############
```

```
stat_maker=function(s2){
   s1="statistics"
   s3="value"
   s2=as.character(s2)
   for (i in 1:nrow(df)) {
                                          df$pop2[i]       <-
coordinates2statistics(df[i,4],df[i,7],s2)[paste(s1,s2,s3, sep
= ".")]
     assign("test2",50,envir = .GlobalEnv)


   }
 }
stat_maker("population_density")
test2
```

```
## [1] 50
```

Data Hacking with RDSTK (part 1)

RDSTK is a very versatile package. It includes functions to help you convert IP address to geo locations and derive statistics from them. It also allows you to input a body of text and convert it into sentiments.

This package provides an R interface to Pete Warden's Data Science Toolkit. See www.datasciencetoolkit.org for more information.

Answers to the exercises are available here. If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
Install and load the RDSTK package.

Exercise 2

Convert the ip adress to co-ordinates. address="165.124.145.197". Store the results under the variable stat

Exercise 3

Derive the elevation of that location using the lattitude and longitude. Use the function coordinate coordinates2statistics() function to achieve this. Once you get the elevation store this back as one of the features of stat.

Exercise 4

Derive the population_density of that location using the lattitude and longitude. Use the function coordinate coordinates2statistics() function to achieve this. Once you get the elevation store this back as one of the features of stat called pop_den.

Exercise 5

Great. You are getting the hang of it. Let us try getting the mean temperature of that location. You will notice that it returns a list of 12 numbers, each for a month.

Run this code and see yourself

```
coordinates2statistics(stat[3],stat[6],"mean_temperature")[1]
```

Exercise 6

We have to transform the mean_temperature so we can store this as one of the features in our stat dataset. One way to do this is to convert it from long to wide format but that would be too reduntant. Let’s just find the mean temperature from January-December. You might find the sapply function useful to convert each element in the list to integers.

Exercise 7

We decided we do not really need January-December mean value. We actually need the mean temperature from June-December. Make that adjustment to your last code and store the results back in stat under the name mean_temp

Exercise 8

Okay great. Now lets work with more IP-address data. Here is a list of a few ip-addresses scraped from a few commenters of my exercises.

```
list=c("165.124.145.197″,"31.24.74.155″,"79.129.19.173")
df=data.frame(list)
df[,1]=as.character(df[,1])
```

Exercise 9

Use a iterator like apply that will go through the list and derive its statistics with the ip2coordinates() function. This is the first part. You may get a list within list sort of result. Store this in a variable called data

Exercise 10

Use a method to convert that list within list into a dataframe with 3 rows and all columns derived from the ip2coordinates() function. You are open to use any method for this.

Data Hacking with RDSTK (part 1) solution

Below are the solutions to these exercises on data hacking with RDSTK (part 1).

```
###############
#             #
# Exercise 1  #
#             #
###############

library(RDSTK)
```

```
###############
#             #
# Exercise 2  #
#             #
###############
stat=ip2coordinates("165.124.145.197")

###############
#             #
# Exercise 3  #
#             #
###############

stat$elevation=coordinates2statistics(stat[3],stat[6],"elevati
on")['statistics.elevation.value']

###############
#             #
# Exercise 4  #
#             #
###############

stat$pop_den=coordinates2statistics(stat[3],stat[6],"populatio
n_density")["statistics.population_density.value"]

###############
#             #
# Exercise 5  #
#             #
###############

coordinates2statistics(stat[3],stat[6],"mean_temperature")["st
atistics.mean_temperature.value"]

##    statistics.mean_temperature.value
## 1                               -5.2
## 2                               -3.0
## 3                                2.3
## 4                                8.7
## 5                               14.7
## 6                               20.1
```

```
## 7                                    22.8
## 8                                    22.2
## 9                                    18.0
## 10                                   11.7
## 11                                    4.5
## 12                                   -2.2

###############
#             #
# Exercise 6  #
#             #
###############


mean(sapply(coordinates2statistics(stat[3],stat[6],"mean_tempe
rature")["statistics.mean_temperature.value"],as.integer))

## [1] 9.25

###############
#             #
# Exercise 7  #
#             #
###############
stat$mean_temp=mean(sapply(coordinates2statistics(stat[3],stat
[6],"mean_temperature")["statistics.mean_temperature.value"][c
(6:12),],as.integer))


###############
#             #
# Exercise 8  #
#             #
###############

list=c("165.124.145.197","31.24.74.155","79.129.19.173")
df=data.frame(list)
df[,1]=as.character(df[,1])
```

```
###############
#             #
# Exercise 9  #
#             #
###############

data=lapply(df[,1],ip2coordinates)


###############
#             #
# Exercise 10  #
#             #
###############


do.call(rbind.data.frame,data)

##        ip.address dma_code latitude country_code3 area_code
longitude
## 1 165.124.145.197      602  42.0586           USA       847
-87.6845
## 2    31.24.74.155        0  57.1000           SWE         0
12.2500
## 3   79.129.19.173        0  39.7378           GRC         0
22.2892
##    country_name postal_code region    locality country_code
## 1 United States       60208     IL    Evanston           US
## 2        Sweden                 06     Varberg           SE
## 3        Greece                 21 T<ed>rnavos           GR
```

Let’s get started with dplyr

The dplyr package by Hadley Wickham is a very useful package that provides “A Grammar of Data Manipulation”. It aims to simplify common data manipulation tasks, and provides “verbs”, i.e. functions that correspond to the most common data manipulation tasks. Have fun playing with dplyr in the exercises below!

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Exercise 1
Install and load the package dplyr package. Given the metadata:

Wt: weight of the subject (kg).

Dose: dose of theophylline administered orally to the subject (mg/kg).

Time: time since drug administration when the sample was drawn (hr).

conc: theophylline concentration in the sample (mg/L).

Copy and paste this code to get df

```
df=data.frame(Theoph)
library(dplyr)
```

Exercise 2

Use the names() function to get the column names of df.

Exercise 3

Let’s practice using the select() function. This allows you to work with just column names instead of indices.

a) Select only the columns starting from Subject to Dose
b) Only select the Wt and Dose columns now.

Learn more about dplyr in Section 5 *Using dplyr on one and multiple Datasets* of the online course R Data Pre-Processing & Data Management – Shape your Data! Rated 4.6 / 5 (45 ratings) • 473 students enrolled

Exercise 4

Let's look at the sample with Dose greater than 5 mg/kg. Use the filter command() to return df with Dose>5´

Exercise 5

Great. Now use filter command to return df with Dose>5 and Time greater than the mean Time.

Exercise 6

Now let's try sorting the data. Use the arrange() function to
1) arrange df by weight (descending)
2) arrange df by weight (ascending)
3) arrange df by weight (ascending) and Time (descending)

Exercise 7

The mutate() command allows you to create a new column using conditions and data derived from other columns. Use mutate() command to create a new column called trend that equals to Time-mean(Time). This will tell you how far each time value is from its mean. Set na.rm=TRUE.

Exercise 8

Given the meta-data

76.2 kg Super-middleweight
72.57 kg Middleweight
69.85 kg Light-middleweight
66.68 kg Welterweight

Use the mutate function to classify the weight using the information above. For the purpose of this exercise, considering anything above 76.2 kg to be Super-middleweight and anything below 66.8 to be Welterweight. Anything below 76.2 to be middleweight and anything below 72.57 to be light-middleweight. Store the classifications under weight_cat. Hint: Use ifelse function() with mutate() to achieve this. Store this back into df.

Exercise 9

Use the groupby() command to group df by weight_cat. This allows us to use aggregated functions similar to group by in SQL. Store this in a df called weight_group

Exercise 10

Use the summarize() command on the weight_group created in Question 9 to find the mean Time and sum of Dose received by each weight categories.

Let's get started with dplyr Solution

Below are the solutions to these exercises on dplyr.

Learn more about dplyr in Section 5 *Using dplyr on one and multiple Datasets* of the online course R Data Pre-Processing & Data Management – Shape your Data! Rated 4.6 / 5 (45 ratings) • 473 students enrolled

```
###############
#             #
# Exercise 1  #
```

```
#             #
###############

df=data.frame(Theoph)
library(dplyr)


###############
#             #
# Exercise 2  #
#             #
###############
names(df)

## [1] "Subject" "Wt"      "Dose"    "Time"    "conc"

###############
#             #
# Exercise 3  #
#             #
###############

select(df,Subject:Dose)

##     Subject   Wt Dose
## 1         1 79.6 4.02
## 2         1 79.6 4.02
## 3         1 79.6 4.02
## 4         1 79.6 4.02
## 5         1 79.6 4.02
## 6         1 79.6 4.02
## 7         1 79.6 4.02
## 8         1 79.6 4.02
## 9         1 79.6 4.02
## 10        1 79.6 4.02
## 11        1 79.6 4.02
## 12        2 72.4 4.40
## 13        2 72.4 4.40
## 14        2 72.4 4.40
## 15        2 72.4 4.40
## 16        2 72.4 4.40
```

```
## 17        2 72.4 4.40
## 18        2 72.4 4.40
## 19        2 72.4 4.40
## 20        2 72.4 4.40
## 21        2 72.4 4.40
## 22        2 72.4 4.40
## 23        3 70.5 4.53
## 24        3 70.5 4.53
## 25        3 70.5 4.53
## 26        3 70.5 4.53
## 27        3 70.5 4.53
## 28        3 70.5 4.53
## 29        3 70.5 4.53
## 30        3 70.5 4.53
## 31        3 70.5 4.53
## 32        3 70.5 4.53
## 33        3 70.5 4.53
## 34        4 72.7 4.40
## 35        4 72.7 4.40
## 36        4 72.7 4.40
## 37        4 72.7 4.40
## 38        4 72.7 4.40
## 39        4 72.7 4.40
## 40        4 72.7 4.40
## 41        4 72.7 4.40
## 42        4 72.7 4.40
## 43        4 72.7 4.40
## 44        4 72.7 4.40
## 45        5 54.6 5.86
## 46        5 54.6 5.86
## 47        5 54.6 5.86
## 48        5 54.6 5.86
## 49        5 54.6 5.86
## 50        5 54.6 5.86
## 51        5 54.6 5.86
## 52        5 54.6 5.86
## 53        5 54.6 5.86
## 54        5 54.6 5.86
## 55        5 54.6 5.86
## 56        6 80.0 4.00
## 57        6 80.0 4.00
```

```
## 58        6 80.0 4.00
## 59        6 80.0 4.00
## 60        6 80.0 4.00
## 61        6 80.0 4.00
## 62        6 80.0 4.00
## 63        6 80.0 4.00
## 64        6 80.0 4.00
## 65        6 80.0 4.00
## 66        6 80.0 4.00
## 67        7 64.6 4.95
## 68        7 64.6 4.95
## 69        7 64.6 4.95
## 70        7 64.6 4.95
## 71        7 64.6 4.95
## 72        7 64.6 4.95
## 73        7 64.6 4.95
## 74        7 64.6 4.95
## 75        7 64.6 4.95
## 76        7 64.6 4.95
## 77        7 64.6 4.95
## 78        8 70.5 4.53
## 79        8 70.5 4.53
## 80        8 70.5 4.53
## 81        8 70.5 4.53
## 82        8 70.5 4.53
## 83        8 70.5 4.53
## 84        8 70.5 4.53
## 85        8 70.5 4.53
## 86        8 70.5 4.53
## 87        8 70.5 4.53
## 88        8 70.5 4.53
## 89        9 86.4 3.10
## 90        9 86.4 3.10
## 91        9 86.4 3.10
## 92        9 86.4 3.10
## 93        9 86.4 3.10
## 94        9 86.4 3.10
## 95        9 86.4 3.10
## 96        9 86.4 3.10
## 97        9 86.4 3.10
## 98        9 86.4 3.10
```

```
## 99        9 86.4 3.10
## 100      10 58.2 5.50
## 101      10 58.2 5.50
## 102      10 58.2 5.50
## 103      10 58.2 5.50
## 104      10 58.2 5.50
## 105      10 58.2 5.50
## 106      10 58.2 5.50
## 107      10 58.2 5.50
## 108      10 58.2 5.50
## 109      10 58.2 5.50
## 110      10 58.2 5.50
## 111      11 65.0 4.92
## 112      11 65.0 4.92
## 113      11 65.0 4.92
## 114      11 65.0 4.92
## 115      11 65.0 4.92
## 116      11 65.0 4.92
## 117      11 65.0 4.92
## 118      11 65.0 4.92
## 119      11 65.0 4.92
## 120      11 65.0 4.92
## 121      11 65.0 4.92
## 122      12 60.5 5.30
## 123      12 60.5 5.30
## 124      12 60.5 5.30
## 125      12 60.5 5.30
## 126      12 60.5 5.30
## 127      12 60.5 5.30
## 128      12 60.5 5.30
## 129      12 60.5 5.30
## 130      12 60.5 5.30
## 131      12 60.5 5.30
## 132      12 60.5 5.30
```

```
select(df,Subject,Weight)
```

```
## Error in eval(expr, envir, enclos): object 'Weight' not
found
```

```
###############
```

```
#             #
# Exercise 4  #
#             #
###############

filter(df,Dose>5)
```

```
##    Subject   Wt Dose  Time  conc
## 1        5 54.6 5.86  0.00  0.00
## 2        5 54.6 5.86  0.30  2.02
## 3        5 54.6 5.86  0.52  5.63
## 4        5 54.6 5.86  1.00 11.40
## 5        5 54.6 5.86  2.02  9.33
## 6        5 54.6 5.86  3.50  8.74
## 7        5 54.6 5.86  5.02  7.56
## 8        5 54.6 5.86  7.02  7.09
## 9        5 54.6 5.86  9.10  5.90
## 10       5 54.6 5.86 12.00  4.37
## 11       5 54.6 5.86 24.35  1.57
## 12      10 58.2 5.50  0.00  0.24
## 13      10 58.2 5.50  0.37  2.89
## 14      10 58.2 5.50  0.77  5.22
## 15      10 58.2 5.50  1.02  6.41
## 16      10 58.2 5.50  2.05  7.83
## 17      10 58.2 5.50  3.55 10.21
## 18      10 58.2 5.50  5.05  9.18
## 19      10 58.2 5.50  7.08  8.02
## 20      10 58.2 5.50  9.38  7.14
## 21      10 58.2 5.50 12.10  5.68
## 22      10 58.2 5.50 23.70  2.42
## 23      12 60.5 5.30  0.00  0.00
## 24      12 60.5 5.30  0.25  1.25
## 25      12 60.5 5.30  0.50  3.96
## 26      12 60.5 5.30  1.00  7.82
## 27      12 60.5 5.30  2.00  9.72
## 28      12 60.5 5.30  3.52  9.75
## 29      12 60.5 5.30  5.07  8.57
## 30      12 60.5 5.30  7.07  6.59
## 31      12 60.5 5.30  9.03  6.11
## 32      12 60.5 5.30 12.05  4.57
## 33      12 60.5 5.30 24.15  1.17
```

```
###############
#             #
# Exercise 5  #
#             #
###############

filter(df,Dose>5 & Time>mean(Time))

##    Subject   Wt Dose  Time conc
## 1        5 54.6 5.86  7.02 7.09
## 2        5 54.6 5.86  9.10 5.90
## 3        5 54.6 5.86 12.00 4.37
## 4        5 54.6 5.86 24.35 1.57
## 5       10 58.2 5.50  7.08 8.02
## 6       10 58.2 5.50  9.38 7.14
## 7       10 58.2 5.50 12.10 5.68
## 8       10 58.2 5.50 23.70 2.42
## 9       12 60.5 5.30  7.07 6.59
## 10      12 60.5 5.30  9.03 6.11
## 11      12 60.5 5.30 12.05 4.57
## 12      12 60.5 5.30 24.15 1.17

###############
#             #
# Exercise 6  #
#             #
###############


arrange(df,desc(Wt))

##     Subject   Wt Dose  Time  conc
## 1         9 86.4 3.10  0.00  0.00
## 2         9 86.4 3.10  0.30  7.37
## 3         9 86.4 3.10  0.63  9.03
## 4         9 86.4 3.10  1.05  7.14
## 5         9 86.4 3.10  2.02  6.33
## 6         9 86.4 3.10  3.53  5.66
## 7         9 86.4 3.10  5.02  5.67
## 8         9 86.4 3.10  7.17  4.24
## 9         9 86.4 3.10  8.80  4.11
```

```
## 10        9 86.4 3.10 11.60  3.16
## 11        9 86.4 3.10 24.43  1.12
## 12        6 80.0 4.00  0.00  0.00
## 13        6 80.0 4.00  0.27  1.29
## 14        6 80.0 4.00  0.58  3.08
## 15        6 80.0 4.00  1.15  6.44
## 16        6 80.0 4.00  2.03  6.32
## 17        6 80.0 4.00  3.57  5.53
## 18        6 80.0 4.00  5.00  4.94
## 19        6 80.0 4.00  7.00  4.02
## 20        6 80.0 4.00  9.22  3.46
## 21        6 80.0 4.00 12.10  2.78
## 22        6 80.0 4.00 23.85  0.92
## 23        1 79.6 4.02  0.00  0.74
## 24        1 79.6 4.02  0.25  2.84
## 25        1 79.6 4.02  0.57  6.57
## 26        1 79.6 4.02  1.12 10.50
## 27        1 79.6 4.02  2.02  9.66
## 28        1 79.6 4.02  3.82  8.58
## 29        1 79.6 4.02  5.10  8.36
## 30        1 79.6 4.02  7.03  7.47
## 31        1 79.6 4.02  9.05  6.89
## 32        1 79.6 4.02 12.12  5.94
## 33        1 79.6 4.02 24.37  3.28
## 34        4 72.7 4.40  0.00  0.00
## 35        4 72.7 4.40  0.35  1.89
## 36        4 72.7 4.40  0.60  4.60
## 37        4 72.7 4.40  1.07  8.60
## 38        4 72.7 4.40  2.13  8.38
## 39        4 72.7 4.40  3.50  7.54
## 40        4 72.7 4.40  5.02  6.88
## 41        4 72.7 4.40  7.02  5.78
## 42        4 72.7 4.40  9.02  5.33
## 43        4 72.7 4.40 11.98  4.19
## 44        4 72.7 4.40 24.65  1.15
## 45        2 72.4 4.40  0.00  0.00
## 46        2 72.4 4.40  0.27  1.72
## 47        2 72.4 4.40  0.52  7.91
## 48        2 72.4 4.40  1.00  8.31
## 49        2 72.4 4.40  1.92  8.33
## 50        2 72.4 4.40  3.50  6.85
```

```
## 51        2 72.4 4.40  5.02  6.08
## 52        2 72.4 4.40  7.03  5.40
## 53        2 72.4 4.40  9.00  4.55
## 54        2 72.4 4.40 12.00  3.01
## 55        2 72.4 4.40 24.30  0.90
## 56        3 70.5 4.53  0.00  0.00
## 57        3 70.5 4.53  0.27  4.40
## 58        3 70.5 4.53  0.58  6.90
## 59        3 70.5 4.53  1.02  8.20
## 60        3 70.5 4.53  2.02  7.80
## 61        3 70.5 4.53  3.62  7.50
## 62        3 70.5 4.53  5.08  6.20
## 63        3 70.5 4.53  7.07  5.30
## 64        3 70.5 4.53  9.00  4.90
## 65        3 70.5 4.53 12.15  3.70
## 66        3 70.5 4.53 24.17  1.05
## 67        8 70.5 4.53  0.00  0.00
## 68        8 70.5 4.53  0.25  3.05
## 69        8 70.5 4.53  0.52  3.05
## 70        8 70.5 4.53  0.98  7.31
## 71        8 70.5 4.53  2.02  7.56
## 72        8 70.5 4.53  3.53  6.59
## 73        8 70.5 4.53  5.05  5.88
## 74        8 70.5 4.53  7.15  4.73
## 75        8 70.5 4.53  9.07  4.57
## 76        8 70.5 4.53 12.10  3.00
## 77        8 70.5 4.53 24.12  1.25
## 78       11 65.0 4.92  0.00  0.00
## 79       11 65.0 4.92  0.25  4.86
## 80       11 65.0 4.92  0.50  7.24
## 81       11 65.0 4.92  0.98  8.00
## 82       11 65.0 4.92  1.98  6.81
## 83       11 65.0 4.92  3.60  5.87
## 84       11 65.0 4.92  5.02  5.22
## 85       11 65.0 4.92  7.03  4.45
## 86       11 65.0 4.92  9.03  3.62
## 87       11 65.0 4.92 12.12  2.69
## 88       11 65.0 4.92 24.08  0.86
## 89        7 64.6 4.95  0.00  0.15
## 90        7 64.6 4.95  0.25  0.85
## 91        7 64.6 4.95  0.50  2.35
```

```
## 92        7 64.6 4.95  1.02  5.02
## 93        7 64.6 4.95  2.02  6.58
## 94        7 64.6 4.95  3.48  7.09
## 95        7 64.6 4.95  5.00  6.66
## 96        7 64.6 4.95  6.98  5.25
## 97        7 64.6 4.95  9.00  4.39
## 98        7 64.6 4.95 12.05  3.53
## 99        7 64.6 4.95 24.22  1.15
## 100      12 60.5 5.30  0.00  0.00
## 101      12 60.5 5.30  0.25  1.25
## 102      12 60.5 5.30  0.50  3.96
## 103      12 60.5 5.30  1.00  7.82
## 104      12 60.5 5.30  2.00  9.72
## 105      12 60.5 5.30  3.52  9.75
## 106      12 60.5 5.30  5.07  8.57
## 107      12 60.5 5.30  7.07  6.59
## 108      12 60.5 5.30  9.03  6.11
## 109      12 60.5 5.30 12.05  4.57
## 110      12 60.5 5.30 24.15  1.17
## 111      10 58.2 5.50  0.00  0.24
## 112      10 58.2 5.50  0.37  2.89
## 113      10 58.2 5.50  0.77  5.22
## 114      10 58.2 5.50  1.02  6.41
## 115      10 58.2 5.50  2.05  7.83
## 116      10 58.2 5.50  3.55 10.21
## 117      10 58.2 5.50  5.05  9.18
## 118      10 58.2 5.50  7.08  8.02
## 119      10 58.2 5.50  9.38  7.14
## 120      10 58.2 5.50 12.10  5.68
## 121      10 58.2 5.50 23.70  2.42
## 122       5 54.6 5.86  0.00  0.00
## 123       5 54.6 5.86  0.30  2.02
## 124       5 54.6 5.86  0.52  5.63
## 125       5 54.6 5.86  1.00 11.40
## 126       5 54.6 5.86  2.02  9.33
## 127       5 54.6 5.86  3.50  8.74
## 128       5 54.6 5.86  5.02  7.56
## 129       5 54.6 5.86  7.02  7.09
## 130       5 54.6 5.86  9.10  5.90
## 131       5 54.6 5.86 12.00  4.37
## 132       5 54.6 5.86 24.35  1.57
```

```
arrange(df,Wt)
```

```
##     Subject   Wt Dose  Time  conc
## 1         5 54.6 5.86  0.00  0.00
## 2         5 54.6 5.86  0.30  2.02
## 3         5 54.6 5.86  0.52  5.63
## 4         5 54.6 5.86  1.00 11.40
## 5         5 54.6 5.86  2.02  9.33
## 6         5 54.6 5.86  3.50  8.74
## 7         5 54.6 5.86  5.02  7.56
## 8         5 54.6 5.86  7.02  7.09
## 9         5 54.6 5.86  9.10  5.90
## 10        5 54.6 5.86 12.00  4.37
## 11        5 54.6 5.86 24.35  1.57
## 12       10 58.2 5.50  0.00  0.24
## 13       10 58.2 5.50  0.37  2.89
## 14       10 58.2 5.50  0.77  5.22
## 15       10 58.2 5.50  1.02  6.41
## 16       10 58.2 5.50  2.05  7.83
## 17       10 58.2 5.50  3.55 10.21
## 18       10 58.2 5.50  5.05  9.18
## 19       10 58.2 5.50  7.08  8.02
## 20       10 58.2 5.50  9.38  7.14
## 21       10 58.2 5.50 12.10  5.68
## 22       10 58.2 5.50 23.70  2.42
## 23       12 60.5 5.30  0.00  0.00
## 24       12 60.5 5.30  0.25  1.25
## 25       12 60.5 5.30  0.50  3.96
## 26       12 60.5 5.30  1.00  7.82
## 27       12 60.5 5.30  2.00  9.72
## 28       12 60.5 5.30  3.52  9.75
## 29       12 60.5 5.30  5.07  8.57
## 30       12 60.5 5.30  7.07  6.59
## 31       12 60.5 5.30  9.03  6.11
## 32       12 60.5 5.30 12.05  4.57
## 33       12 60.5 5.30 24.15  1.17
## 34        7 64.6 4.95  0.00  0.15
## 35        7 64.6 4.95  0.25  0.85
## 36        7 64.6 4.95  0.50  2.35
## 37        7 64.6 4.95  1.02  5.02
## 38        7 64.6 4.95  2.02  6.58
```

```
## 39        7 64.6 4.95  3.48  7.09
## 40        7 64.6 4.95  5.00  6.66
## 41        7 64.6 4.95  6.98  5.25
## 42        7 64.6 4.95  9.00  4.39
## 43        7 64.6 4.95 12.05  3.53
## 44        7 64.6 4.95 24.22  1.15
## 45       11 65.0 4.92  0.00  0.00
## 46       11 65.0 4.92  0.25  4.86
## 47       11 65.0 4.92  0.50  7.24
## 48       11 65.0 4.92  0.98  8.00
## 49       11 65.0 4.92  1.98  6.81
## 50       11 65.0 4.92  3.60  5.87
## 51       11 65.0 4.92  5.02  5.22
## 52       11 65.0 4.92  7.03  4.45
## 53       11 65.0 4.92  9.03  3.62
## 54       11 65.0 4.92 12.12  2.69
## 55       11 65.0 4.92 24.08  0.86
## 56        3 70.5 4.53  0.00  0.00
## 57        3 70.5 4.53  0.27  4.40
## 58        3 70.5 4.53  0.58  6.90
## 59        3 70.5 4.53  1.02  8.20
## 60        3 70.5 4.53  2.02  7.80
## 61        3 70.5 4.53  3.62  7.50
## 62        3 70.5 4.53  5.08  6.20
## 63        3 70.5 4.53  7.07  5.30
## 64        3 70.5 4.53  9.00  4.90
## 65        3 70.5 4.53 12.15  3.70
## 66        3 70.5 4.53 24.17  1.05
## 67        8 70.5 4.53  0.00  0.00
## 68        8 70.5 4.53  0.25  3.05
## 69        8 70.5 4.53  0.52  3.05
## 70        8 70.5 4.53  0.98  7.31
## 71        8 70.5 4.53  2.02  7.56
## 72        8 70.5 4.53  3.53  6.59
## 73        8 70.5 4.53  5.05  5.88
## 74        8 70.5 4.53  7.15  4.73
## 75        8 70.5 4.53  9.07  4.57
## 76        8 70.5 4.53 12.10  3.00
## 77        8 70.5 4.53 24.12  1.25
## 78        2 72.4 4.40  0.00  0.00
## 79        2 72.4 4.40  0.27  1.72
```

```
## 80        2 72.4 4.40  0.52  7.91
## 81        2 72.4 4.40  1.00  8.31
## 82        2 72.4 4.40  1.92  8.33
## 83        2 72.4 4.40  3.50  6.85
## 84        2 72.4 4.40  5.02  6.08
## 85        2 72.4 4.40  7.03  5.40
## 86        2 72.4 4.40  9.00  4.55
## 87        2 72.4 4.40 12.00  3.01
## 88        2 72.4 4.40 24.30  0.90
## 89        4 72.7 4.40  0.00  0.00
## 90        4 72.7 4.40  0.35  1.89
## 91        4 72.7 4.40  0.60  4.60
## 92        4 72.7 4.40  1.07  8.60
## 93        4 72.7 4.40  2.13  8.38
## 94        4 72.7 4.40  3.50  7.54
## 95        4 72.7 4.40  5.02  6.88
## 96        4 72.7 4.40  7.02  5.78
## 97        4 72.7 4.40  9.02  5.33
## 98        4 72.7 4.40 11.98  4.19
## 99        4 72.7 4.40 24.65  1.15
## 100       1 79.6 4.02  0.00  0.74
## 101       1 79.6 4.02  0.25  2.84
## 102       1 79.6 4.02  0.57  6.57
## 103       1 79.6 4.02  1.12 10.50
## 104       1 79.6 4.02  2.02  9.66
## 105       1 79.6 4.02  3.82  8.58
## 106       1 79.6 4.02  5.10  8.36
## 107       1 79.6 4.02  7.03  7.47
## 108       1 79.6 4.02  9.05  6.89
## 109       1 79.6 4.02 12.12  5.94
## 110       1 79.6 4.02 24.37  3.28
## 111       6 80.0 4.00  0.00  0.00
## 112       6 80.0 4.00  0.27  1.29
## 113       6 80.0 4.00  0.58  3.08
## 114       6 80.0 4.00  1.15  6.44
## 115       6 80.0 4.00  2.03  6.32
## 116       6 80.0 4.00  3.57  5.53
## 117       6 80.0 4.00  5.00  4.94
## 118       6 80.0 4.00  7.00  4.02
## 119       6 80.0 4.00  9.22  3.46
## 120       6 80.0 4.00 12.10  2.78
```

```
## 121       6 80.0 4.00 23.85  0.92
## 122       9 86.4 3.10  0.00  0.00
## 123       9 86.4 3.10  0.30  7.37
## 124       9 86.4 3.10  0.63  9.03
## 125       9 86.4 3.10  1.05  7.14
## 126       9 86.4 3.10  2.02  6.33
## 127       9 86.4 3.10  3.53  5.66
## 128       9 86.4 3.10  5.02  5.67
## 129       9 86.4 3.10  7.17  4.24
## 130       9 86.4 3.10  8.80  4.11
## 131       9 86.4 3.10 11.60  3.16
## 132       9 86.4 3.10 24.43  1.12
```

```
arrange(df,Wt,desc(Time))
```

```
##     Subject   Wt Dose  Time  conc
## 1         5 54.6 5.86 24.35  1.57
## 2         5 54.6 5.86 12.00  4.37
## 3         5 54.6 5.86  9.10  5.90
## 4         5 54.6 5.86  7.02  7.09
## 5         5 54.6 5.86  5.02  7.56
## 6         5 54.6 5.86  3.50  8.74
## 7         5 54.6 5.86  2.02  9.33
## 8         5 54.6 5.86  1.00 11.40
## 9         5 54.6 5.86  0.52  5.63
## 10        5 54.6 5.86  0.30  2.02
## 11        5 54.6 5.86  0.00  0.00
## 12       10 58.2 5.50 23.70  2.42
## 13       10 58.2 5.50 12.10  5.68
## 14       10 58.2 5.50  9.38  7.14
## 15       10 58.2 5.50  7.08  8.02
## 16       10 58.2 5.50  5.05  9.18
## 17       10 58.2 5.50  3.55 10.21
## 18       10 58.2 5.50  2.05  7.83
## 19       10 58.2 5.50  1.02  6.41
## 20       10 58.2 5.50  0.77  5.22
## 21       10 58.2 5.50  0.37  2.89
## 22       10 58.2 5.50  0.00  0.24
## 23       12 60.5 5.30 24.15  1.17
## 24       12 60.5 5.30 12.05  4.57
## 25       12 60.5 5.30  9.03  6.11
```

```
## 26       12 60.5 5.30  7.07  6.59
## 27       12 60.5 5.30  5.07  8.57
## 28       12 60.5 5.30  3.52  9.75
## 29       12 60.5 5.30  2.00  9.72
## 30       12 60.5 5.30  1.00  7.82
## 31       12 60.5 5.30  0.50  3.96
## 32       12 60.5 5.30  0.25  1.25
## 33       12 60.5 5.30  0.00  0.00
## 34        7 64.6 4.95 24.22  1.15
## 35        7 64.6 4.95 12.05  3.53
## 36        7 64.6 4.95  9.00  4.39
## 37        7 64.6 4.95  6.98  5.25
## 38        7 64.6 4.95  5.00  6.66
## 39        7 64.6 4.95  3.48  7.09
## 40        7 64.6 4.95  2.02  6.58
## 41        7 64.6 4.95  1.02  5.02
## 42        7 64.6 4.95  0.50  2.35
## 43        7 64.6 4.95  0.25  0.85
## 44        7 64.6 4.95  0.00  0.15
## 45       11 65.0 4.92 24.08  0.86
## 46       11 65.0 4.92 12.12  2.69
## 47       11 65.0 4.92  9.03  3.62
## 48       11 65.0 4.92  7.03  4.45
## 49       11 65.0 4.92  5.02  5.22
## 50       11 65.0 4.92  3.60  5.87
## 51       11 65.0 4.92  1.98  6.81
## 52       11 65.0 4.92  0.98  8.00
## 53       11 65.0 4.92  0.50  7.24
## 54       11 65.0 4.92  0.25  4.86
## 55       11 65.0 4.92  0.00  0.00
## 56        3 70.5 4.53 24.17  1.05
## 57        8 70.5 4.53 24.12  1.25
## 58        3 70.5 4.53 12.15  3.70
## 59        8 70.5 4.53 12.10  3.00
## 60        8 70.5 4.53  9.07  4.57
## 61        3 70.5 4.53  9.00  4.90
## 62        8 70.5 4.53  7.15  4.73
## 63        3 70.5 4.53  7.07  5.30
## 64        3 70.5 4.53  5.08  6.20
## 65        8 70.5 4.53  5.05  5.88
## 66        3 70.5 4.53  3.62  7.50
```

```
## 67        8 70.5 4.53  3.53  6.59
## 68        3 70.5 4.53  2.02  7.80
## 69        8 70.5 4.53  2.02  7.56
## 70        3 70.5 4.53  1.02  8.20
## 71        8 70.5 4.53  0.98  7.31
## 72        3 70.5 4.53  0.58  6.90
## 73        8 70.5 4.53  0.52  3.05
## 74        3 70.5 4.53  0.27  4.40
## 75        8 70.5 4.53  0.25  3.05
## 76        3 70.5 4.53  0.00  0.00
## 77        8 70.5 4.53  0.00  0.00
## 78        2 72.4 4.40 24.30  0.90
## 79        2 72.4 4.40 12.00  3.01
## 80        2 72.4 4.40  9.00  4.55
## 81        2 72.4 4.40  7.03  5.40
## 82        2 72.4 4.40  5.02  6.08
## 83        2 72.4 4.40  3.50  6.85
## 84        2 72.4 4.40  1.92  8.33
## 85        2 72.4 4.40  1.00  8.31
## 86        2 72.4 4.40  0.52  7.91
## 87        2 72.4 4.40  0.27  1.72
## 88        2 72.4 4.40  0.00  0.00
## 89        4 72.7 4.40 24.65  1.15
## 90        4 72.7 4.40 11.98  4.19
## 91        4 72.7 4.40  9.02  5.33
## 92        4 72.7 4.40  7.02  5.78
## 93        4 72.7 4.40  5.02  6.88
## 94        4 72.7 4.40  3.50  7.54
## 95        4 72.7 4.40  2.13  8.38
## 96        4 72.7 4.40  1.07  8.60
## 97        4 72.7 4.40  0.60  4.60
## 98        4 72.7 4.40  0.35  1.89
## 99        4 72.7 4.40  0.00  0.00
## 100       1 79.6 4.02 24.37  3.28
## 101       1 79.6 4.02 12.12  5.94
## 102       1 79.6 4.02  9.05  6.89
## 103       1 79.6 4.02  7.03  7.47
## 104       1 79.6 4.02  5.10  8.36
## 105       1 79.6 4.02  3.82  8.58
## 106       1 79.6 4.02  2.02  9.66
## 107       1 79.6 4.02  1.12 10.50
```

```
## 108       1 79.6 4.02  0.57  6.57
## 109       1 79.6 4.02  0.25  2.84
## 110       1 79.6 4.02  0.00  0.74
## 111       6 80.0 4.00 23.85  0.92
## 112       6 80.0 4.00 12.10  2.78
## 113       6 80.0 4.00  9.22  3.46
## 114       6 80.0 4.00  7.00  4.02
## 115       6 80.0 4.00  5.00  4.94
## 116       6 80.0 4.00  3.57  5.53
## 117       6 80.0 4.00  2.03  6.32
## 118       6 80.0 4.00  1.15  6.44
## 119       6 80.0 4.00  0.58  3.08
## 120       6 80.0 4.00  0.27  1.29
## 121       6 80.0 4.00  0.00  0.00
## 122       9 86.4 3.10 24.43  1.12
## 123       9 86.4 3.10 11.60  3.16
## 124       9 86.4 3.10  8.80  4.11
## 125       9 86.4 3.10  7.17  4.24
## 126       9 86.4 3.10  5.02  5.67
## 127       9 86.4 3.10  3.53  5.66
## 128       9 86.4 3.10  2.02  6.33
## 129       9 86.4 3.10  1.05  7.14
## 130       9 86.4 3.10  0.63  9.03
## 131       9 86.4 3.10  0.30  7.37
## 132       9 86.4 3.10  0.00  0.00

###############
#             #
# Exercise 7  #
#             #
###############
mutate(df,trend=Time-mean(Time,na.rm = TRUE))

##     Subject   Wt Dose  Time  conc      trend
## 1         1 79.6 4.02  0.00  0.74 -5.8946212
## 2         1 79.6 4.02  0.25  2.84 -5.6446212
## 3         1 79.6 4.02  0.57  6.57 -5.3246212
## 4         1 79.6 4.02  1.12 10.50 -4.7746212
## 5         1 79.6 4.02  2.02  9.66 -3.8746212
## 6         1 79.6 4.02  3.82  8.58 -2.0746212
## 7         1 79.6 4.02  5.10  8.36 -0.7946212
```

```
## 8         1 79.6 4.02  7.03  7.47  1.1353788
## 9         1 79.6 4.02  9.05  6.89  3.1553788
## 10        1 79.6 4.02 12.12  5.94  6.2253788
## 11        1 79.6 4.02 24.37  3.28 18.4753788
## 12        2 72.4 4.40  0.00  0.00 -5.8946212
## 13        2 72.4 4.40  0.27  1.72 -5.6246212
## 14        2 72.4 4.40  0.52  7.91 -5.3746212
## 15        2 72.4 4.40  1.00  8.31 -4.8946212
## 16        2 72.4 4.40  1.92  8.33 -3.9746212
## 17        2 72.4 4.40  3.50  6.85 -2.3946212
## 18        2 72.4 4.40  5.02  6.08 -0.8746212
## 19        2 72.4 4.40  7.03  5.40  1.1353788
## 20        2 72.4 4.40  9.00  4.55  3.1053788
## 21        2 72.4 4.40 12.00  3.01  6.1053788
## 22        2 72.4 4.40 24.30  0.90 18.4053788
## 23        3 70.5 4.53  0.00  0.00 -5.8946212
## 24        3 70.5 4.53  0.27  4.40 -5.6246212
## 25        3 70.5 4.53  0.58  6.90 -5.3146212
## 26        3 70.5 4.53  1.02  8.20 -4.8746212
## 27        3 70.5 4.53  2.02  7.80 -3.8746212
## 28        3 70.5 4.53  3.62  7.50 -2.2746212
## 29        3 70.5 4.53  5.08  6.20 -0.8146212
## 30        3 70.5 4.53  7.07  5.30  1.1753788
## 31        3 70.5 4.53  9.00  4.90  3.1053788
## 32        3 70.5 4.53 12.15  3.70  6.2553788
## 33        3 70.5 4.53 24.17  1.05 18.2753788
## 34        4 72.7 4.40  0.00  0.00 -5.8946212
## 35        4 72.7 4.40  0.35  1.89 -5.5446212
## 36        4 72.7 4.40  0.60  4.60 -5.2946212
## 37        4 72.7 4.40  1.07  8.60 -4.8246212
## 38        4 72.7 4.40  2.13  8.38 -3.7646212
## 39        4 72.7 4.40  3.50  7.54 -2.3946212
## 40        4 72.7 4.40  5.02  6.88 -0.8746212
## 41        4 72.7 4.40  7.02  5.78  1.1253788
## 42        4 72.7 4.40  9.02  5.33  3.1253788
## 43        4 72.7 4.40 11.98  4.19  6.0853788
## 44        4 72.7 4.40 24.65  1.15 18.7553788
## 45        5 54.6 5.86  0.00  0.00 -5.8946212
## 46        5 54.6 5.86  0.30  2.02 -5.5946212
## 47        5 54.6 5.86  0.52  5.63 -5.3746212
## 48        5 54.6 5.86  1.00 11.40 -4.8946212
```

```
## 49        5 54.6 5.86  2.02  9.33 -3.8746212
## 50        5 54.6 5.86  3.50  8.74 -2.3946212
## 51        5 54.6 5.86  5.02  7.56 -0.8746212
## 52        5 54.6 5.86  7.02  7.09  1.1253788
## 53        5 54.6 5.86  9.10  5.90  3.2053788
## 54        5 54.6 5.86 12.00  4.37  6.1053788
## 55        5 54.6 5.86 24.35  1.57 18.4553788
## 56        6 80.0 4.00  0.00  0.00 -5.8946212
## 57        6 80.0 4.00  0.27  1.29 -5.6246212
## 58        6 80.0 4.00  0.58  3.08 -5.3146212
## 59        6 80.0 4.00  1.15  6.44 -4.7446212
## 60        6 80.0 4.00  2.03  6.32 -3.8646212
## 61        6 80.0 4.00  3.57  5.53 -2.3246212
## 62        6 80.0 4.00  5.00  4.94 -0.8946212
## 63        6 80.0 4.00  7.00  4.02  1.1053788
## 64        6 80.0 4.00  9.22  3.46  3.3253788
## 65        6 80.0 4.00 12.10  2.78  6.2053788
## 66        6 80.0 4.00 23.85  0.92 17.9553788
## 67        7 64.6 4.95  0.00  0.15 -5.8946212
## 68        7 64.6 4.95  0.25  0.85 -5.6446212
## 69        7 64.6 4.95  0.50  2.35 -5.3946212
## 70        7 64.6 4.95  1.02  5.02 -4.8746212
## 71        7 64.6 4.95  2.02  6.58 -3.8746212
## 72        7 64.6 4.95  3.48  7.09 -2.4146212
## 73        7 64.6 4.95  5.00  6.66 -0.8946212
## 74        7 64.6 4.95  6.98  5.25  1.0853788
## 75        7 64.6 4.95  9.00  4.39  3.1053788
## 76        7 64.6 4.95 12.05  3.53  6.1553788
## 77        7 64.6 4.95 24.22  1.15 18.3253788
## 78        8 70.5 4.53  0.00  0.00 -5.8946212
## 79        8 70.5 4.53  0.25  3.05 -5.6446212
## 80        8 70.5 4.53  0.52  3.05 -5.3746212
## 81        8 70.5 4.53  0.98  7.31 -4.9146212
## 82        8 70.5 4.53  2.02  7.56 -3.8746212
## 83        8 70.5 4.53  3.53  6.59 -2.3646212
## 84        8 70.5 4.53  5.05  5.88 -0.8446212
## 85        8 70.5 4.53  7.15  4.73  1.2553788
## 86        8 70.5 4.53  9.07  4.57  3.1753788
## 87        8 70.5 4.53 12.10  3.00  6.2053788
## 88        8 70.5 4.53 24.12  1.25 18.2253788
## 89        9 86.4 3.10  0.00  0.00 -5.8946212
```

```
## 90        9 86.4 3.10  0.30  7.37 -5.5946212
## 91        9 86.4 3.10  0.63  9.03 -5.2646212
## 92        9 86.4 3.10  1.05  7.14 -4.8446212
## 93        9 86.4 3.10  2.02  6.33 -3.8746212
## 94        9 86.4 3.10  3.53  5.66 -2.3646212
## 95        9 86.4 3.10  5.02  5.67 -0.8746212
## 96        9 86.4 3.10  7.17  4.24  1.2753788
## 97        9 86.4 3.10  8.80  4.11  2.9053788
## 98        9 86.4 3.10 11.60  3.16  5.7053788
## 99        9 86.4 3.10 24.43  1.12 18.5353788
## 100      10 58.2 5.50  0.00  0.24 -5.8946212
## 101      10 58.2 5.50  0.37  2.89 -5.5246212
## 102      10 58.2 5.50  0.77  5.22 -5.1246212
## 103      10 58.2 5.50  1.02  6.41 -4.8746212
## 104      10 58.2 5.50  2.05  7.83 -3.8446212
## 105      10 58.2 5.50  3.55 10.21 -2.3446212
## 106      10 58.2 5.50  5.05  9.18 -0.8446212
## 107      10 58.2 5.50  7.08  8.02  1.1853788
## 108      10 58.2 5.50  9.38  7.14  3.4853788
## 109      10 58.2 5.50 12.10  5.68  6.2053788
## 110      10 58.2 5.50 23.70  2.42 17.8053788
## 111      11 65.0 4.92  0.00  0.00 -5.8946212
## 112      11 65.0 4.92  0.25  4.86 -5.6446212
## 113      11 65.0 4.92  0.50  7.24 -5.3946212
## 114      11 65.0 4.92  0.98  8.00 -4.9146212
## 115      11 65.0 4.92  1.98  6.81 -3.9146212
## 116      11 65.0 4.92  3.60  5.87 -2.2946212
## 117      11 65.0 4.92  5.02  5.22 -0.8746212
## 118      11 65.0 4.92  7.03  4.45  1.1353788
## 119      11 65.0 4.92  9.03  3.62  3.1353788
## 120      11 65.0 4.92 12.12  2.69  6.2253788
## 121      11 65.0 4.92 24.08  0.86 18.1853788
## 122      12 60.5 5.30  0.00  0.00 -5.8946212
## 123      12 60.5 5.30  0.25  1.25 -5.6446212
## 124      12 60.5 5.30  0.50  3.96 -5.3946212
## 125      12 60.5 5.30  1.00  7.82 -4.8946212
## 126      12 60.5 5.30  2.00  9.72 -3.8946212
## 127      12 60.5 5.30  3.52  9.75 -2.3746212
## 128      12 60.5 5.30  5.07  8.57 -0.8246212
## 129      12 60.5 5.30  7.07  6.59  1.1753788
## 130      12 60.5 5.30  9.03  6.11  3.1353788
```

```
## 131      12 60.5 5.30 12.05  4.57  6.1553788
## 132      12 60.5 5.30 24.15  1.17 18.2553788

###############
#             #
# Exercise 8  #
#             #
###############

mutate(df,  weight_cat  =  ifelse(Wt  <  69.85  ,
"Welterweight",ifelse(Wt   <   72.57   ,   "Light-
middleweight",ifelse(Wt  <  76.2   ,  "middleweight",
ifelse(Wt>76.2, "Super-Middleweight", "Unknown")))))

##     Subject   Wt Dose  Time  conc         weight_cat
## 1         1 79.6 4.02  0.00  0.74 Super-Middleweight
## 2         1 79.6 4.02  0.25  2.84 Super-Middleweight
## 3         1 79.6 4.02  0.57  6.57 Super-Middleweight
## 4         1 79.6 4.02  1.12 10.50 Super-Middleweight
## 5         1 79.6 4.02  2.02  9.66 Super-Middleweight
## 6         1 79.6 4.02  3.82  8.58 Super-Middleweight
## 7         1 79.6 4.02  5.10  8.36 Super-Middleweight
## 8         1 79.6 4.02  7.03  7.47 Super-Middleweight
## 9         1 79.6 4.02  9.05  6.89 Super-Middleweight
## 10        1 79.6 4.02 12.12  5.94 Super-Middleweight
## 11        1 79.6 4.02 24.37  3.28 Super-Middleweight
## 12        2 72.4 4.40  0.00  0.00 Light-middleweight
## 13        2 72.4 4.40  0.27  1.72 Light-middleweight
## 14        2 72.4 4.40  0.52  7.91 Light-middleweight
## 15        2 72.4 4.40  1.00  8.31 Light-middleweight
## 16        2 72.4 4.40  1.92  8.33 Light-middleweight
## 17        2 72.4 4.40  3.50  6.85 Light-middleweight
## 18        2 72.4 4.40  5.02  6.08 Light-middleweight
## 19        2 72.4 4.40  7.03  5.40 Light-middleweight
## 20        2 72.4 4.40  9.00  4.55 Light-middleweight
## 21        2 72.4 4.40 12.00  3.01 Light-middleweight
## 22        2 72.4 4.40 24.30  0.90 Light-middleweight
## 23        3 70.5 4.53  0.00  0.00 Light-middleweight
## 24        3 70.5 4.53  0.27  4.40 Light-middleweight
## 25        3 70.5 4.53  0.58  6.90 Light-middleweight
## 26        3 70.5 4.53  1.02  8.20 Light-middleweight
```

```
## 27         3 70.5 4.53  2.02  7.80 Light-middleweight
## 28         3 70.5 4.53  3.62  7.50 Light-middleweight
## 29         3 70.5 4.53  5.08  6.20 Light-middleweight
## 30         3 70.5 4.53  7.07  5.30 Light-middleweight
## 31         3 70.5 4.53  9.00  4.90 Light-middleweight
## 32         3 70.5 4.53 12.15  3.70 Light-middleweight
## 33         3 70.5 4.53 24.17  1.05 Light-middleweight
## 34         4 72.7 4.40  0.00  0.00       middleweight
## 35         4 72.7 4.40  0.35  1.89       middleweight
## 36         4 72.7 4.40  0.60  4.60       middleweight
## 37         4 72.7 4.40  1.07  8.60       middleweight
## 38         4 72.7 4.40  2.13  8.38       middleweight
## 39         4 72.7 4.40  3.50  7.54       middleweight
## 40         4 72.7 4.40  5.02  6.88       middleweight
## 41         4 72.7 4.40  7.02  5.78       middleweight
## 42         4 72.7 4.40  9.02  5.33       middleweight
## 43         4 72.7 4.40 11.98  4.19       middleweight
## 44         4 72.7 4.40 24.65  1.15       middleweight
## 45         5 54.6 5.86  0.00  0.00       Welterweight
## 46         5 54.6 5.86  0.30  2.02       Welterweight
## 47         5 54.6 5.86  0.52  5.63       Welterweight
## 48         5 54.6 5.86  1.00 11.40       Welterweight
## 49         5 54.6 5.86  2.02  9.33       Welterweight
## 50         5 54.6 5.86  3.50  8.74       Welterweight
## 51         5 54.6 5.86  5.02  7.56       Welterweight
## 52         5 54.6 5.86  7.02  7.09       Welterweight
## 53         5 54.6 5.86  9.10  5.90       Welterweight
## 54         5 54.6 5.86 12.00  4.37       Welterweight
## 55         5 54.6 5.86 24.35  1.57       Welterweight
## 56         6 80.0 4.00  0.00  0.00 Super-Middleweight
## 57         6 80.0 4.00  0.27  1.29 Super-Middleweight
## 58         6 80.0 4.00  0.58  3.08 Super-Middleweight
## 59         6 80.0 4.00  1.15  6.44 Super-Middleweight
## 60         6 80.0 4.00  2.03  6.32 Super-Middleweight
## 61         6 80.0 4.00  3.57  5.53 Super-Middleweight
## 62         6 80.0 4.00  5.00  4.94 Super-Middleweight
## 63         6 80.0 4.00  7.00  4.02 Super-Middleweight
## 64         6 80.0 4.00  9.22  3.46 Super-Middleweight
## 65         6 80.0 4.00 12.10  2.78 Super-Middleweight
## 66         6 80.0 4.00 23.85  0.92 Super-Middleweight
## 67         7 64.6 4.95  0.00  0.15       Welterweight
```

```
## 68         7 64.6 4.95  0.25  0.85       Welterweight
## 69         7 64.6 4.95  0.50  2.35       Welterweight
## 70         7 64.6 4.95  1.02  5.02       Welterweight
## 71         7 64.6 4.95  2.02  6.58       Welterweight
## 72         7 64.6 4.95  3.48  7.09       Welterweight
## 73         7 64.6 4.95  5.00  6.66       Welterweight
## 74         7 64.6 4.95  6.98  5.25       Welterweight
## 75         7 64.6 4.95  9.00  4.39       Welterweight
## 76         7 64.6 4.95 12.05  3.53       Welterweight
## 77         7 64.6 4.95 24.22  1.15       Welterweight
## 78         8 70.5 4.53  0.00  0.00 Light-middleweight
## 79         8 70.5 4.53  0.25  3.05 Light-middleweight
## 80         8 70.5 4.53  0.52  3.05 Light-middleweight
## 81         8 70.5 4.53  0.98  7.31 Light-middleweight
## 82         8 70.5 4.53  2.02  7.56 Light-middleweight
## 83         8 70.5 4.53  3.53  6.59 Light-middleweight
## 84         8 70.5 4.53  5.05  5.88 Light-middleweight
## 85         8 70.5 4.53  7.15  4.73 Light-middleweight
## 86         8 70.5 4.53  9.07  4.57 Light-middleweight
## 87         8 70.5 4.53 12.10  3.00 Light-middleweight
## 88         8 70.5 4.53 24.12  1.25 Light-middleweight
## 89         9 86.4 3.10  0.00  0.00 Super-Middleweight
## 90         9 86.4 3.10  0.30  7.37 Super-Middleweight
## 91         9 86.4 3.10  0.63  9.03 Super-Middleweight
## 92         9 86.4 3.10  1.05  7.14 Super-Middleweight
## 93         9 86.4 3.10  2.02  6.33 Super-Middleweight
## 94         9 86.4 3.10  3.53  5.66 Super-Middleweight
## 95         9 86.4 3.10  5.02  5.67 Super-Middleweight
## 96         9 86.4 3.10  7.17  4.24 Super-Middleweight
## 97         9 86.4 3.10  8.80  4.11 Super-Middleweight
## 98         9 86.4 3.10 11.60  3.16 Super-Middleweight
## 99         9 86.4 3.10 24.43  1.12 Super-Middleweight
## 100       10 58.2 5.50  0.00  0.24       Welterweight
## 101       10 58.2 5.50  0.37  2.89       Welterweight
## 102       10 58.2 5.50  0.77  5.22       Welterweight
## 103       10 58.2 5.50  1.02  6.41       Welterweight
## 104       10 58.2 5.50  2.05  7.83       Welterweight
## 105       10 58.2 5.50  3.55 10.21       Welterweight
## 106       10 58.2 5.50  5.05  9.18       Welterweight
## 107       10 58.2 5.50  7.08  8.02       Welterweight
## 108       10 58.2 5.50  9.38  7.14       Welterweight
```

```
## 109      10 58.2 5.50 12.10  5.68       Welterweight
## 110      10 58.2 5.50 23.70  2.42       Welterweight
## 111      11 65.0 4.92  0.00  0.00       Welterweight
## 112      11 65.0 4.92  0.25  4.86       Welterweight
## 113      11 65.0 4.92  0.50  7.24       Welterweight
## 114      11 65.0 4.92  0.98  8.00       Welterweight
## 115      11 65.0 4.92  1.98  6.81       Welterweight
## 116      11 65.0 4.92  3.60  5.87       Welterweight
## 117      11 65.0 4.92  5.02  5.22       Welterweight
## 118      11 65.0 4.92  7.03  4.45       Welterweight
## 119      11 65.0 4.92  9.03  3.62       Welterweight
## 120      11 65.0 4.92 12.12  2.69       Welterweight
## 121      11 65.0 4.92 24.08  0.86       Welterweight
## 122      12 60.5 5.30  0.00  0.00       Welterweight
## 123      12 60.5 5.30  0.25  1.25       Welterweight
## 124      12 60.5 5.30  0.50  3.96       Welterweight
## 125      12 60.5 5.30  1.00  7.82       Welterweight
## 126      12 60.5 5.30  2.00  9.72       Welterweight
## 127      12 60.5 5.30  3.52  9.75       Welterweight
## 128      12 60.5 5.30  5.07  8.57       Welterweight
## 129      12 60.5 5.30  7.07  6.59       Welterweight
## 130      12 60.5 5.30  9.03  6.11       Welterweight
## 131      12 60.5 5.30 12.05  4.57       Welterweight
## 132      12 60.5 5.30 24.15  1.17       Welterweight
```

```
###############
#             #
# Exercise 9  #
#             #
###############
```

```
weight_group=group_by(df_2,weight_cat)
```

```
## Error in group_by_(.data, .dots = lazyeval::lazy_dots(...),
add = add): object 'df_2' not found
```

```
###############
#             #
# Exercise 10  #
#             #
###############
```

```
summarize(weight_cat,Avg_Time=mean(Time),Total_Dose=sum(Dose))

## # A tibble: 4 × 3
##           weight_cat Avg_Time Total_Dose
##                <chr>    <dbl>      <dbl>
## 1 Light-middleweight 5.888788     148.06
## 2       middleweight 5.940000      48.40
## 3 Super-Middleweight 5.902121     122.32
## 4       Welterweight 5.884545     291.83
```

Tidy the data up!

Tidyr makes it easier for you to tidy your data. Some common uses are seperating columns with a seperator, uniting two columns into one, stacking and unstacking column categories etc.

Answers to the exercises are available here.

If you obtained a different (correct) answer than those listed on the solutions page, please feel free to post your answer as a comment on that page.

Learn more about Tidyr in the online course R Data Pre-Processing & Data Management – Shape your Data! This course has 5 lectures on Tidyr, in addition to lectures on importing data, filtering and querying using dplyr, and SQL.

Exercise 1

Install and load the package tidyr. Copy and paste this code to get df

```
First=c("Josh","Emily","Clark","Barack")
```

```
Last=c(“Blah”,”Choo”,”Kent”,”Obama”)
age=c(21,24,30,50)
jobs=c(“programmer/Athlete”,”Blogger/Doctor”,”Reporter/Crime Fighter”,”President/commander in chief”)
hometown=c(“san-antonio,texas”,”corona,california”,”smallville,Kansas”,”chicago,iilinois”)

df= data.frame(First,Last,age,jobs,hometown)
```

Exercise 2

Use the separate() function to seperate jobs into First_job and Second_job using the appropriate seperator. Store it back into df.

Exercise 3

Combine the First and Last column together with one space between them to form the column “Name”. Use the function unite(). Store this back into df

Exercise 4

Use the separate() function to seperate hometown column into City and State using the appropriate seperator. Store it back into df.

Exercise 5

Add this code

```
x2014=c(“32131″,”41231″,”51341″,”12312”)
x2015=c(“2132131″,”12341231″,”12351341″,”12312312”)
x2016=c(“3122132131″,”512341231″,”12312351341″,”4412312312”)
inst_followers=data.frame(name,x2014,x2015,x2016)
```

view the inst_followers dataframe using the str() command

Exercise 6

Stack the year columns from the inst_followers data from wide to long format using the gather() command. Dataframe should now have columns name, year and followers only. Store this back in the dataframe.

Exercise 7

Unstack the columns and spread the year categories of the new dataframe out into columns, bringing it back to the original wide format. Use the spread() command.

Exercise 8

Type this code:

```
Month=c(“Jan”,”Feb”,”Mar”,”Apr”)
stock_A=c(1854.8,1850.3,1812.1,1835.3)
stock_B=c(1854.8,1850.3,1812.1,1835.3)
stocks=data.frame(Month,stock_A,stock_B)
stocks
```

Exercise 9

Use the gather() command to gather stock_A and stock_B. The columns should now look like | Month | Stock | Earnings|
Store this in stocks.

Exercise 10

Use the spread() command to bring the stocks d back to its original shape.

Tidy the Data Up Solutions

Below are the solutions to these exercises on sorting and ordering.

```
###############
#             #
# Exercise 1  #
#             #
###############

library(tidyr)
First=c("Josh","Emily","Clark","Barack")
Last=c("Blah","Choo","Kent","Obama")
age=c(21,24,30,50)
jobs=c("programmer/Athlete","Blogger/Doctor","Reporter/Crime
Fighter","President/commander in chief")
hometown=c("san-
antonio,texas","corona,california","smallville,Kansas","chicag
o,iilinois")

df= data.frame(First,Last,age,jobs,hometown)


###############
#             #
# Exercise 2  #
#             #
###############
df=separate(df,jobs,c("First_job","Second_job"),sep="/")

###############
#             #
# Exercise 3  #
#             #
###############

df=unite(df,"Name",c(First,Last),sep=" ")
```

```
###############
#             #
# Exercise 4  #
#             #
###############

df=separate(df,hometown,c("City","State"),sep=",")

###############
#             #
# Exercise 5  #
#             #
###############

x2014=c("32131","41231","51341","12312")
x2015=c("2132131","12341231","12351341","12312312")
x2016=c("3122132131","512341231","12312351341","4412312312")
inst_followers=data.frame(name,x2014,x2015,x2016)
str(inst_followers)

## 'data.frame':        4 obs. of  4 variables:
##  $ name : Factor w/ 4 levels "Barack","Clark",..: 4 3 2 1
##  $ x2014: Factor w/ 4 levels "12312","32131",..: 2 3 4 1
##  $ x2015: Factor w/ 4 levels "12312312","12341231",..: 4 2
3 1
##  $ x2016: Factor w/ 4 levels "12312351341",..: 2 4 1 3

###############
#             #
# Exercise 6  #
#             #
###############
inst_followers=gather(inst_followers,year,followers,x2014:x201
6)

## Warning: attributes are not identical across measure
variables; they will
## be dropped

###############
```

```
#             #
# Exercise 7  #
#             #
###############
spread(inst_followers,year,followers)
```

```
##     name x2014    x2015       x2016
## 1 Barack 12312 12312312  4412312312
## 2  Clark 51341 12351341 12312351341
## 3  Emily 41231 12341231   512341231
## 4   Josh 32131  2132131  3122132131
```

```
###############
#             #
# Exercise 8  #
#             #
###############

Month=c("Jan","Feb","Mar","Apr")
stock_A=c(1854.8,1850.3,1812.1,1835.3)
stock_B=c(1854.8,1850.3,1812.1,1835.3)
stocks=data.frame(Month,stock_A,stock_B)
stocks
```

```
##   Month stock_A stock_B
## 1   Jan  1854.8  1854.8
## 2   Feb  1850.3  1850.3
## 3   Mar  1812.1  1812.1
## 4   Apr  1835.3  1835.3
```

```
###############
#             #
# Exercise 9  #
#             #
###############

stocks=gather(stocks,stock,earnings,stock_A:stock_B)


###############
```

```
#             #
# Exercise 10  #
#             #
###############

spread(stocks,stock,earnings)

##   Month stock_A stock_B
## 1   Apr  1835.3  1835.3
## 2   Feb  1850.3  1850.3
## 3   Jan  1854.8  1854.8
## 4   Mar  1812.1  1812.1
```