Data wrangling : Reshaping

Data wrangling is a task of great importance in data analysis. Data wrangling, is the process of importing, cleaning and transforming raw data into actionable information for analysis. It is a time-consuming process which is estimated to take about 60-80% of analyst's time. In this series we will go through this process. It will be a brief series with goal to craft the reader's skills on the data wrangling task. This is the second part of this series and it aims to cover the reshaping of data used to turn them into a tidy form. By tidy form, we mean that each feature forms a column and each observation forms a row.

Before proceeding, it might be helpful to look over the help pages for the spread, gather, unite, separate, replace_na, fill, extract_numeric.

Moreover please load the following libraries.

```
install.packages("magrittr")
library(magrittr)
install.packages("tidyr")
library(tidyr)
```

Please run the code below in order to load the data set:

```
data <- airquality[4:6]
```

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

Print out the structure of the data frame.

Exercise 2

Let’s turn the data frame in a wider form, from above and and turn the Month variable into column headings and spread the Temp values across the months they are related to.

Exercise 3

Turn the wide (exercise 2) data frame into its initial format using the gather function, specify the columns you would like to gather by index number.

Exercise 4

Turn the wide (exercise 2) data frame into its initial format using the gather function, specify the columns you would like to gather by column name.

Exercise 5

Turn the wide (exercise 2) data frame into its initial format using the gather function, specify the columns by using remaining column names(the ones you don’t use for gathering).

Exercise 6

Unite the variables Day and Month to a new feature named Date with the format %d-%m .

Exercise 7

Create the data frame at its previous format (exercise 6). Separate the variable you have created before (Date) to Day, Month.

Exercise 8

Replace the missing values (NA) with 'Unknown'.

Exercise 9

Run the script below, so that you make a new feature year.
back2long_na$year <- rep(NA, nrow(back2long_na))

back2long_na$year[1] <- '2015'
back2long_na$year[as.integer(nrow(back2long_na)/3)] <- '2016'

back2long_na$year[as.integer(2*nrow(back2long_na)/3)] <- '2017'

You have noticed, that the new column has many values. Fill the NAs with the non-missing value write above it. (eg.the NA’s that are below the ‘2016’ and ‘2017’ value assign it to ‘2016’.

Hint: use the fill function.

Exercise 10

Extract the numeric values from the Temp feature.

Hint: extract_numeric, this is a very important function when the variable we apply the function on is a character with ‘noise’, for example ‘$40’ and you want to transform it to 40.

Data wrangling : Reshaping Solution

Below are the solutions to these exercises on data reshaping.

```
#####################  #  #  #  Exercise  1  #  #  #
#################### str(data)
```

```
## 'data.frame': 153 obs. of 3 variables: ## $ Temp : int 67
72 74 62 56 66 65 59 61 69 ... ## $ Month: int 5 5 5 5 5 5 5 5
5 5 ... ## $ Day : int 1 2 3 4 5 6 7 8 9 10 ...
```

```
#####################  #  #  #  Exercise  2  #  #  #
#################### long2wide <- data %>% spread(Month,
Temp); long2wide
```

```
## Day 5 6 7 8 9 ## 1 1 67 78 84 81 91 ## 2 2 72 74 85 81 92
## 3 3 74 67 81 82 93 ## 4 4 62 84 84 86 93 ## 5 5 56 85 83 85
87 ## 6 6 66 79 83 87 84 ## 7 7 65 82 88 89 80 ## 8 8 59 87 92
90 78 ## 9 9 61 90 92 90 75 ## 10 10 69 87 89 92 73 ## 11 11
74 93 82 86 81 ## 12 12 69 92 73 86 76 ## 13 13 66 82 81 82 77
## 14 14 68 80 91 80 71 ## 15 15 58 79 80 79 71 ## 16 16 64 77
81 77 78 ## 17 17 66 72 82 79 67 ## 18 18 57 65 84 76 76 ## 19
19 68 73 87 78 68 ## 20 20 62 76 85 78 82 ## 21 21 59 77 74 77
64 ## 22 22 73 76 81 72 71 ## 23 23 61 76 82 75 81 ## 24 24 61
76 86 79 69 ## 25 25 57 75 85 81 63 ## 26 26 58 78 82 86 70 ##
27 27 57 73 86 88 77 ## 28 28 67 80 88 97 75 ## 29 29 81 77 86
94 76 ## 30 30 79 83 83 96 68 ## 31 31 76 NA 81 94 NA
```

```
#####################  #  #  #  Exercise  3  #  #  #
#################### back2long <- long2wide %>% gather(Month,
Temp, 2:6); back2long
```

```
## Day Month Temp ## 1 1 5 67 ## 2 2 5 72 ## 3 3 5 74 ## 4 4 5
62 ## 5 5 5 56 ## 6 6 5 66 ## 7 7 5 65 ## 8 8 5 59 ## 9 9 5 61
## 10 10 5 69 ## 11 11 5 74 ## 12 12 5 69 ## 13 13 5 66 ## 14
14 5 68 ## 15 15 5 58 ## 16 16 5 64 ## 17 17 5 66 ## 18 18 5
57 ## 19 19 5 68 ## 20 20 5 62 ## 21 21 5 59 ## 22 22 5 73 ##
```

23 23 5 61 ## 24 24 5 61 ## 25 25 5 57 ## 26 26 5 58 ## 27 27 5 57 ## 28 28 5 67 ## 29 29 5 81 ## 30 30 5 79 ## 31 31 5 76 ## 32 1 6 78 ## 33 2 6 74 ## 34 3 6 67 ## 35 4 6 84 ## 36 5 6 85 ## 37 6 6 79 ## 38 7 6 82 ## 39 8 6 87 ## 40 9 6 90 ## 41 10 6 87 ## 42 11 6 93 ## 43 12 6 92 ## 44 13 6 82 ## 45 14 6 80 ## 46 15 6 79 ## 47 16 6 77 ## 48 17 6 72 ## 49 18 6 65 ## 50 19 6 73 ## 51 20 6 76 ## 52 21 6 77 ## 53 22 6 76 ## 54 23 6 76 ## 55 24 6 76 ## 56 25 6 75 ## 57 26 6 78 ## 58 27 6 73 ## 59 28 6 80 ## 60 29 6 77 ## 61 30 6 83 ## 62 31 6 NA ## 63 1 7 84 ## 64 2 7 85 ## 65 3 7 81 ## 66 4 7 84 ## 67 5 7 83 ## 68 6 7 83 ## 69 7 7 88 ## 70 8 7 92 ## 71 9 7 92 ## 72 10 7 89 ## 73 11 7 82 ## 74 12 7 73 ## 75 13 7 81 ## 76 14 7 91 ## 77 15 7 80 ## 78 16 7 81 ## 79 17 7 82 ## 80 18 7 84 ## 81 19 7 87 ## 82 20 7 85 ## 83 21 7 74 ## 84 22 7 81 ## 85 23 7 82 ## 86 24 7 86 ## 87 25 7 85 ## 88 26 7 82 ## 89 27 7 86 ## 90 28 7 88 ## 91 29 7 86 ## 92 30 7 83 ## 93 31 7 81 ## 94 1 8 81 ## 95 2 8 81 ## 96 3 8 82 ## 97 4 8 86 ## 98 5 8 85 ## 99 6 8 87 ## 100 7 8 89 ## 101 8 8 90 ## 102 9 8 90 ## 103 10 8 92 ## 104 11 8 86 ## 105 12 8 86 ## 106 13 8 82 ## 107 14 8 80 ## 108 15 8 79 ## 109 16 8 77 ## 110 17 8 79 ## 111 18 8 76 ## 112 19 8 78 ## 113 20 8 78 ## 114 21 8 77 ## 115 22 8 72 ## 116 23 8 75 ## 117 24 8 79 ## 118 25 8 81 ## 119 26 8 86 ## 120 27 8 88 ## 121 28 8 97 ## 122 29 8 94 ## 123 30 8 96 ## 124 31 8 94 ## 125 1 9 91 ## 126 2 9 92 ## 127 3 9 93 ## 128 4 9 93 ## 129 5 9 87 ## 130 6 9 84 ## 131 7 9 80 ## 132 8 9 78 ## 133 9 9 75 ## 134 10 9 73 ## 135 11 9 81 ## 136 12 9 76 ## 137 13 9 77 ## 138 14 9 71 ## 139 15 9 71 ## 140 16 9 78 ## 141 17 9 67 ## 142 18 9 76 ## 143 19 9 68 ## 144 20 9 82 ## 145 21 9 64 ## 146 22 9 71 ## 147 23 9 81 ## 148 24 9 69 ## 149 25 9 63 ## 150 26 9 70 ## 151 27 9 77 ## 152 28 9 75 ## 153 29 9 76 ## 154 30 9 68 ## 155 31 9 NA

########## # # # Exercise 4 # # # ########## back2long <- long2wide %>% gather(Month, Temp, "5", "6", "7", "8", "9"); back2long

Day Month Temp ## 1 1 5 67 ## 2 2 5 72 ## 3 3 5 74 ## 4 4 5 62 ## 5 5 5 56 ## 6 6 5 66 ## 7 7 5 65 ## 8 8 5 59 ## 9 9 5 61 ## 10 10 5 69 ## 11 11 5 74 ## 12 12 5 69 ## 13 13 5 66 ## 14 14 5 68 ## 15 15 5 58 ## 16 16 5 64 ## 17 17 5 66 ## 18 18 5 57 ## 19 19 5 68 ## 20 20 5 62 ## 21 21 5 59 ## 22 22 5 73

23 23 5 61 ## 24 24 5 61 ## 25 25 5 57 ## 26 26 5 58 ## 27 27 5 57 ## 28 28 5 67 ## 29 29 5 81 ## 30 30 5 79 ## 31 31 5 76 ## 32 1 6 78 ## 33 2 6 74 ## 34 3 6 67 ## 35 4 6 84 ## 36 5 6 85 ## 37 6 6 79 ## 38 7 6 82 ## 39 8 6 87 ## 40 9 6 90 ## 41 10 6 87 ## 42 11 6 93 ## 43 12 6 92 ## 44 13 6 82 ## 45 14 6 80 ## 46 15 6 79 ## 47 16 6 77 ## 48 17 6 72 ## 49 18 6 65 ## 50 19 6 73 ## 51 20 6 76 ## 52 21 6 77 ## 53 22 6 76 ## 54 23 6 76 ## 55 24 6 76 ## 56 25 6 75 ## 57 26 6 78 ## 58 27 6 73 ## 59 28 6 80 ## 60 29 6 77 ## 61 30 6 83 ## 62 31 6 NA ## 63 1 7 84 ## 64 2 7 85 ## 65 3 7 81 ## 66 4 7 84 ## 67 5 7 83 ## 68 6 7 83 ## 69 7 7 88 ## 70 8 7 92 ## 71 9 7 92 ## 72 10 7 89 ## 73 11 7 82 ## 74 12 7 73 ## 75 13 7 81 ## 76 14 7 91 ## 77 15 7 80 ## 78 16 7 81 ## 79 17 7 82 ## 80 18 7 84 ## 81 19 7 87 ## 82 20 7 85 ## 83 21 7 74 ## 84 22 7 81 ## 85 23 7 82 ## 86 24 7 86 ## 87 25 7 85 ## 88 26 7 82 ## 89 27 7 86 ## 90 28 7 88 ## 91 29 7 86 ## 92 30 7 83 ## 93 31 7 81 ## 94 1 8 81 ## 95 2 8 81 ## 96 3 8 82 ## 97 4 8 86 ## 98 5 8 85 ## 99 6 8 87 ## 100 7 8 89 ## 101 8 8 90 ## 102 9 8 90 ## 103 10 8 92 ## 104 11 8 86 ## 105 12 8 86 ## 106 13 8 82 ## 107 14 8 80 ## 108 15 8 79 ## 109 16 8 77 ## 110 17 8 79 ## 111 18 8 76 ## 112 19 8 78 ## 113 20 8 78 ## 114 21 8 77 ## 115 22 8 72 ## 116 23 8 75 ## 117 24 8 79 ## 118 25 8 81 ## 119 26 8 86 ## 120 27 8 88 ## 121 28 8 97 ## 122 29 8 94 ## 123 30 8 96 ## 124 31 8 94 ## 125 1 9 91 ## 126 2 9 92 ## 127 3 9 93 ## 128 4 9 93 ## 129 5 9 87 ## 130 6 9 84 ## 131 7 9 80 ## 132 8 9 78 ## 133 9 9 75 ## 134 10 9 73 ## 135 11 9 81 ## 136 12 9 76 ## 137 13 9 77 ## 138 14 9 71 ## 139 15 9 71 ## 140 16 9 78 ## 141 17 9 67 ## 142 18 9 76 ## 143 19 9 68 ## 144 20 9 82 ## 145 21 9 64 ## 146 22 9 71 ## 147 23 9 81 ## 148 24 9 69 ## 149 25 9 63 ## 150 26 9 70 ## 151 27 9 77 ## 152 28 9 75 ## 153 29 9 76 ## 154 30 9 68 ## 155 31 9 NA

########## # # # Exercise 5 # # # ########## back2long <- long2wide %>% gather(Month, Temp, -Day); back2long

Day Month Temp ## 1 1 5 67 ## 2 2 5 72 ## 3 3 5 74 ## 4 4 5 62 ## 5 5 5 56 ## 6 6 5 66 ## 7 7 5 65 ## 8 8 5 59 ## 9 9 5 61 ## 10 10 5 69 ## 11 11 5 74 ## 12 12 5 69 ## 13 13 5 66 ## 14 14 5 68 ## 15 15 5 58 ## 16 16 5 64 ## 17 17 5 66 ## 18 18 5 57 ## 19 19 5 68 ## 20 20 5 62 ## 21 21 5 59 ## 22 22 5 73

23 23 5 61 ## 24 24 5 61 ## 25 25 5 57 ## 26 26 5 58 ## 27 27 5 57 ## 28 28 5 67 ## 29 29 5 81 ## 30 30 5 79 ## 31 31 5 76 ## 32 1 6 78 ## 33 2 6 74 ## 34 3 6 67 ## 35 4 6 84 ## 36 5 6 85 ## 37 6 6 79 ## 38 7 6 82 ## 39 8 6 87 ## 40 9 6 90 ## 41 10 6 87 ## 42 11 6 93 ## 43 12 6 92 ## 44 13 6 82 ## 45 14 6 80 ## 46 15 6 79 ## 47 16 6 77 ## 48 17 6 72 ## 49 18 6 65 ## 50 19 6 73 ## 51 20 6 76 ## 52 21 6 77 ## 53 22 6 76 ## 54 23 6 76 ## 55 24 6 76 ## 56 25 6 75 ## 57 26 6 78 ## 58 27 6 73 ## 59 28 6 80 ## 60 29 6 77 ## 61 30 6 83 ## 62 31 6 NA ## 63 1 7 84 ## 64 2 7 85 ## 65 3 7 81 ## 66 4 7 84 ## 67 5 7 83 ## 68 6 7 83 ## 69 7 7 88 ## 70 8 7 92 ## 71 9 7 92 ## 72 10 7 89 ## 73 11 7 82 ## 74 12 7 73 ## 75 13 7 81 ## 76 14 7 91 ## 77 15 7 80 ## 78 16 7 81 ## 79 17 7 82 ## 80 18 7 84 ## 81 19 7 87 ## 82 20 7 85 ## 83 21 7 74 ## 84 22 7 81 ## 85 23 7 82 ## 86 24 7 86 ## 87 25 7 85 ## 88 26 7 82 ## 89 27 7 86 ## 90 28 7 88 ## 91 29 7 86 ## 92 30 7 83 ## 93 31 7 81 ## 94 1 8 81 ## 95 2 8 81 ## 96 3 8 82 ## 97 4 8 86 ## 98 5 8 85 ## 99 6 8 87 ## 100 7 8 89 ## 101 8 8 90 ## 102 9 8 90 ## 103 10 8 92 ## 104 11 8 86 ## 105 12 8 86 ## 106 13 8 82 ## 107 14 8 80 ## 108 15 8 79 ## 109 16 8 77 ## 110 17 8 79 ## 111 18 8 76 ## 112 19 8 78 ## 113 20 8 78 ## 114 21 8 77 ## 115 22 8 72 ## 116 23 8 75 ## 117 24 8 79 ## 118 25 8 81 ## 119 26 8 86 ## 120 27 8 88 ## 121 28 8 97 ## 122 29 8 94 ## 123 30 8 96 ## 124 31 8 94 ## 125 1 9 91 ## 126 2 9 92 ## 127 3 9 93 ## 128 4 9 93 ## 129 5 9 87 ## 130 6 9 84 ## 131 7 9 80 ## 132 8 9 78 ## 133 9 9 75 ## 134 10 9 73 ## 135 11 9 81 ## 136 12 9 76 ## 137 13 9 77 ## 138 14 9 71 ## 139 15 9 71 ## 140 16 9 78 ## 141 17 9 67 ## 142 18 9 76 ## 143 19 9 68 ## 144 20 9 82 ## 145 21 9 64 ## 146 22 9 71 ## 147 23 9 81 ## 148 24 9 69 ## 149 25 9 63 ## 150 26 9 70 ## 151 27 9 77 ## 152 28 9 75 ## 153 29 9 76 ## 154 30 9 68 ## 155 31 9 NA

```
#####################  #  #  # Exercise 6 #  #  #
#####################
back2long_unite <- back2long %>% unite(col = "Date", c(Day, Month), sep = "-"); back2long_unite
```

Date Temp ## 1 1-5 67 ## 2 2-5 72 ## 3 3-5 74 ## 4 4-5 62 ## 5 5-5 56 ## 6 6-5 66 ## 7 7-5 65 ## 8 8-5 59 ## 9 9-5 61 ## 10 10-5 69 ## 11 11-5 74 ## 12 12-5 69 ## 13 13-5 66 ## 14 14-5 68 ## 15 15-5 58 ## 16 16-5 64 ## 17 17-5 66 ## 18 18-5 57 ## 19 19-5 68 ## 20 20-5 62 ## 21 21-5 59 ## 22 22-5 73

23 23-5 61 ## 24 24-5 61 ## 25 25-5 57 ## 26 26-5 58 ## 27 27-5 57 ## 28 28-5 67 ## 29 29-5 81 ## 30 30-5 79 ## 31 31-5 76 ## 32 1-6 78 ## 33 2-6 74 ## 34 3-6 67 ## 35 4-6 84 ## 36 5-6 85 ## 37 6-6 79 ## 38 7-6 82 ## 39 8-6 87 ## 40 9-6 90 ## 41 10-6 87 ## 42 11-6 93 ## 43 12-6 92 ## 44 13-6 82 ## 45 14-6 80 ## 46 15-6 79 ## 47 16-6 77 ## 48 17-6 72 ## 49 18-6 65 ## 50 19-6 73 ## 51 20-6 76 ## 52 21-6 77 ## 53 22-6 76 ## 54 23-6 76 ## 55 24-6 76 ## 56 25-6 75 ## 57 26-6 78 ## 58 27-6 73 ## 59 28-6 80 ## 60 29-6 77 ## 61 30-6 83 ## 62 31-6 NA ## 63 1-7 84 ## 64 2-7 85 ## 65 3-7 81 ## 66 4-7 84 ## 67 5-7 83 ## 68 6-7 83 ## 69 7-7 88 ## 70 8-7 92 ## 71 9-7 92 ## 72 10-7 89 ## 73 11-7 82 ## 74 12-7 73 ## 75 13-7 81 ## 76 14-7 91 ## 77 15-7 80 ## 78 16-7 81 ## 79 17-7 82 ## 80 18-7 84 ## 81 19-7 87 ## 82 20-7 85 ## 83 21-7 74 ## 84 22-7 81 ## 85 23-7 82 ## 86 24-7 86 ## 87 25-7 85 ## 88 26-7 82 ## 89 27-7 86 ## 90 28-7 88 ## 91 29-7 86 ## 92 30-7 83 ## 93 31-7 81 ## 94 1-8 81 ## 95 2-8 81 ## 96 3-8 82 ## 97 4-8 86 ## 98 5-8 85 ## 99 6-8 87 ## 100 7-8 89 ## 101 8-8 90 ## 102 9-8 90 ## 103 10-8 92 ## 104 11-8 86 ## 105 12-8 86 ## 106 13-8 82 ## 107 14-8 80 ## 108 15-8 79 ## 109 16-8 77 ## 110 17-8 79 ## 111 18-8 76 ## 112 19-8 78 ## 113 20-8 78 ## 114 21-8 77 ## 115 22-8 72 ## 116 23-8 75 ## 117 24-8 79 ## 118 25-8 81 ## 119 26-8 86 ## 120 27-8 88 ## 121 28-8 97 ## 122 29-8 94 ## 123 30-8 96 ## 124 31-8 94 ## 125 1-9 91 ## 126 2-9 92 ## 127 3-9 93 ## 128 4-9 93 ## 129 5-9 87 ## 130 6-9 84 ## 131 7-9 80 ## 132 8-9 78 ## 133 9-9 75 ## 134 10-9 73 ## 135 11-9 81 ## 136 12-9 76 ## 137 13-9 77 ## 138 14-9 71 ## 139 15-9 71 ## 140 16-9 78 ## 141 17-9 67 ## 142 18-9 76 ## 143 19-9 68 ## 144 20-9 82 ## 145 21-9 64 ## 146 22-9 71 ## 147 23-9 81 ## 148 24-9 69 ## 149 25-9 63 ## 150 26-9 70 ## 151 27-9 77 ## 152 28-9 75 ## 153 29-9 76 ## 154 30-9 68 ## 155 31-9 NA

```
#####################  #  #  #  Exercise  7  #  #  #  ##################### back2long_separate <- back2long_unite %>% separate(col = Date, into = c("Day", "Month")) ; back2long_separate
```

Day Month Temp ## 1 1 5 67 ## 2 2 5 72 ## 3 3 5 74 ## 4 4 5 62 ## 5 5 5 56 ## 6 6 5 66 ## 7 7 5 65 ## 8 8 5 59 ## 9 9 5 61 ## 10 10 5 69 ## 11 11 5 74 ## 12 12 5 69 ## 13 13 5 66 ## 14 14 5 68 ## 15 15 5 58 ## 16 16 5 64 ## 17 17 5 66 ## 18 18 5

57 ## 19 19 5 68 ## 20 20 5 62 ## 21 21 5 59 ## 22 22 5 73 ## 23 23 5 61 ## 24 24 5 61 ## 25 25 5 57 ## 26 26 5 58 ## 27 27 5 57 ## 28 28 5 67 ## 29 29 5 81 ## 30 30 5 79 ## 31 31 5 76 ## 32 1 6 78 ## 33 2 6 74 ## 34 3 6 67 ## 35 4 6 84 ## 36 5 6 85 ## 37 6 6 79 ## 38 7 6 82 ## 39 8 6 87 ## 40 9 6 90 ## 41 10 6 87 ## 42 11 6 93 ## 43 12 6 92 ## 44 13 6 82 ## 45 14 6 80 ## 46 15 6 79 ## 47 16 6 77 ## 48 17 6 72 ## 49 18 6 65 ## 50 19 6 73 ## 51 20 6 76 ## 52 21 6 77 ## 53 22 6 76 ## 54 23 6 76 ## 55 24 6 76 ## 56 25 6 75 ## 57 26 6 78 ## 58 27 6 73 ## 59 28 6 80 ## 60 29 6 77 ## 61 30 6 83 ## 62 31 6 NA ## 63 1 7 84 ## 64 2 7 85 ## 65 3 7 81 ## 66 4 7 84 ## 67 5 7 83 ## 68 6 7 83 ## 69 7 7 88 ## 70 8 7 92 ## 71 9 7 92 ## 72 10 7 89 ## 73 11 7 82 ## 74 12 7 73 ## 75 13 7 81 ## 76 14 7 91 ## 77 15 7 80 ## 78 16 7 81 ## 79 17 7 82 ## 80 18 7 84 ## 81 19 7 87 ## 82 20 7 85 ## 83 21 7 74 ## 84 22 7 81 ## 85 23 7 82 ## 86 24 7 86 ## 87 25 7 85 ## 88 26 7 82 ## 89 27 7 86 ## 90 28 7 88 ## 91 29 7 86 ## 92 30 7 83 ## 93 31 7 81 ## 94 1 8 81 ## 95 2 8 81 ## 96 3 8 82 ## 97 4 8 86 ## 98 5 8 85 ## 99 6 8 87 ## 100 7 8 89 ## 101 8 8 90 ## 102 9 8 90 ## 103 10 8 92 ## 104 11 8 86 ## 105 12 8 86 ## 106 13 8 82 ## 107 14 8 80 ## 108 15 8 79 ## 109 16 8 77 ## 110 17 8 79 ## 111 18 8 76 ## 112 19 8 78 ## 113 20 8 78 ## 114 21 8 77 ## 115 22 8 72 ## 116 23 8 75 ## 117 24 8 79 ## 118 25 8 81 ## 119 26 8 86 ## 120 27 8 88 ## 121 28 8 97 ## 122 29 8 94 ## 123 30 8 96 ## 124 31 8 94 ## 125 1 9 91 ## 126 2 9 92 ## 127 3 9 93 ## 128 4 9 93 ## 129 5 9 87 ## 130 6 9 84 ## 131 7 9 80 ## 132 8 9 78 ## 133 9 9 75 ## 134 10 9 73 ## 135 11 9 81 ## 136 12 9 76 ## 137 13 9 77 ## 138 14 9 71 ## 139 15 9 71 ## 140 16 9 78 ## 141 17 9 67 ## 142 18 9 76 ## 143 19 9 68 ## 144 20 9 82 ## 145 21 9 64 ## 146 22 9 71 ## 147 23 9 81 ## 148 24 9 69 ## 149 25 9 63 ## 150 26 9 70 ## 151 27 9 77 ## 152 28 9 75 ## 153 29 9 76 ## 154 30 9 68 ## 155 31 9 NA

```
##################### # # # Exercise 8 # # # #####################
back2long_na <- back2long %>% replace_na(replace = list(Temp = "unknown")) ; back2long_na
```

Day Month Temp ## 1 1 5 67 ## 2 2 5 72 ## 3 3 5 74 ## 4 4 5 62 ## 5 5 5 56 ## 6 6 5 66 ## 7 7 5 65 ## 8 8 5 59 ## 9 9 5 61 ## 10 10 5 69 ## 11 11 5 74 ## 12 12 5 69 ## 13 13 5 66 ## 14 14 5 68 ## 15 15 5 58 ## 16 16 5 64 ## 17 17 5 66 ## 18 18 5

57 ## 19 19 5 68 ## 20 20 5 62 ## 21 21 5 59 ## 22 22 5 73 ## 23 23 5 61 ## 24 24 5 61 ## 25 25 5 57 ## 26 26 5 58 ## 27 27 5 57 ## 28 28 5 67 ## 29 29 5 81 ## 30 30 5 79 ## 31 31 5 76 ## 32 1 6 78 ## 33 2 6 74 ## 34 3 6 67 ## 35 4 6 84 ## 36 5 6 85 ## 37 6 6 79 ## 38 7 6 82 ## 39 8 6 87 ## 40 9 6 90 ## 41 10 6 87 ## 42 11 6 93 ## 43 12 6 92 ## 44 13 6 82 ## 45 14 6 80 ## 46 15 6 79 ## 47 16 6 77 ## 48 17 6 72 ## 49 18 6 65 ## 50 19 6 73 ## 51 20 6 76 ## 52 21 6 77 ## 53 22 6 76 ## 54 23 6 76 ## 55 24 6 76 ## 56 25 6 75 ## 57 26 6 78 ## 58 27 6 73 ## 59 28 6 80 ## 60 29 6 77 ## 61 30 6 83 ## 62 31 6 unknown ## 63 1 7 84 ## 64 2 7 85 ## 65 3 7 81 ## 66 4 7 84 ## 67 5 7 83 ## 68 6 7 83 ## 69 7 7 88 ## 70 8 7 92 ## 71 9 7 92 ## 72 10 7 89 ## 73 11 7 82 ## 74 12 7 73 ## 75 13 7 81 ## 76 14 7 91 ## 77 15 7 80 ## 78 16 7 81 ## 79 17 7 82 ## 80 18 7 84 ## 81 19 7 87 ## 82 20 7 85 ## 83 21 7 74 ## 84 22 7 81 ## 85 23 7 82 ## 86 24 7 86 ## 87 25 7 85 ## 88 26 7 82 ## 89 27 7 86 ## 90 28 7 88 ## 91 29 7 86 ## 92 30 7 83 ## 93 31 7 81 ## 94 1 8 81 ## 95 2 8 81 ## 96 3 8 82 ## 97 4 8 86 ## 98 5 8 85 ## 99 6 8 87 ## 100 7 8 89 ## 101 8 8 90 ## 102 9 8 90 ## 103 10 8 92 ## 104 11 8 86 ## 105 12 8 86 ## 106 13 8 82 ## 107 14 8 80 ## 108 15 8 79 ## 109 16 8 77 ## 110 17 8 79 ## 111 18 8 76 ## 112 19 8 78 ## 113 20 8 78 ## 114 21 8 77 ## 115 22 8 72 ## 116 23 8 75 ## 117 24 8 79 ## 118 25 8 81 ## 119 26 8 86 ## 120 27 8 88 ## 121 28 8 97 ## 122 29 8 94 ## 123 30 8 96 ## 124 31 8 94 ## 125 1 9 91 ## 126 2 9 92 ## 127 3 9 93 ## 128 4 9 93 ## 129 5 9 87 ## 130 6 9 84 ## 131 7 9 80 ## 132 8 9 78 ## 133 9 9 75 ## 134 10 9 73 ## 135 11 9 81 ## 136 12 9 76 ## 137 13 9 77 ## 138 14 9 71 ## 139 15 9 71 ## 140 16 9 78 ## 141 17 9 67 ## 142 18 9 76 ## 143 19 9 68 ## 144 20 9 82 ## 145 21 9 64 ## 146 22 9 71 ## 147 23 9 81 ## 148 24 9 69 ## 149 25 9 63 ## 150 26 9 70 ## 151 27 9 77 ## 152 28 9 75 ## 153 29 9 76 ## 154 30 9 68 ## 155 31 9 unknown

```
#####################  #  #  #  Exercise  9  #  #  #  ##################### back2long_na <- back2long_na %>% fill(year) ; back2long_na
```

Day Month Temp year ## 1 1 5 67 2015 ## 2 2 5 72 2015 ## 3 3 5 74 2015 ## 4 4 5 62 2015 ## 5 5 5 56 2015 ## 6 6 5 66 2015 ## 7 7 5 65 2015 ## 8 8 5 59 2015 ## 9 9 5 61 2015 ## 10 10 5 69 2015 ## 11 11 5 74 2015 ## 12 12 5 69 2015 ## 13 13 5 66

2015 ## 14 14 5 68 2015 ## 15 15 5 58 2015 ## 16 16 5 64 2015 ## 17 17 5 66 2015 ## 18 18 5 57 2015 ## 19 19 5 68 2015 ## 20 20 5 62 2015 ## 21 21 5 59 2015 ## 22 22 5 73 2015 ## 23 23 5 61 2015 ## 24 24 5 61 2015 ## 25 25 5 57 2015 ## 26 26 5 58 2015 ## 27 27 5 57 2015 ## 28 28 5 67 2015 ## 29 29 5 81 2015 ## 30 30 5 79 2015 ## 31 31 5 76 2015 ## 32 1 6 78 2015 ## 33 2 6 74 2015 ## 34 3 6 67 2015 ## 35 4 6 84 2015 ## 36 5 6 85 2015 ## 37 6 6 79 2015 ## 38 7 6 82 2015 ## 39 8 6 87 2015 ## 40 9 6 90 2015 ## 41 10 6 87 2015 ## 42 11 6 93 2015 ## 43 12 6 92 2015 ## 44 13 6 82 2015 ## 45 14 6 80 2015 ## 46 15 6 79 2015 ## 47 16 6 77 2015 ## 48 17 6 72 2015 ## 49 18 6 65 2015 ## 50 19 6 73 2015 ## 51 20 6 76 2016 ## 52 21 6 77 2016 ## 53 22 6 76 2016 ## 54 23 6 76 2016 ## 55 24 6 76 2016 ## 56 25 6 75 2016 ## 57 26 6 78 2016 ## 58 27 6 73 2016 ## 59 28 6 80 2016 ## 60 29 6 77 2016 ## 61 30 6 83 2016 ## 62 31 6 unknown 2016 ## 63 1 7 84 2016 ## 64 2 7 85 2016 ## 65 3 7 81 2016 ## 66 4 7 84 2016 ## 67 5 7 83 2016 ## 68 6 7 83 2016 ## 69 7 7 88 2016 ## 70 8 7 92 2016 ## 71 9 7 92 2016 ## 72 10 7 89 2016 ## 73 11 7 82 2016 ## 74 12 7 73 2016 ## 75 13 7 81 2016 ## 76 14 7 91 2016 ## 77 15 7 80 2016 ## 78 16 7 81 2016 ## 79 17 7 82 2016 ## 80 18 7 84 2016 ## 81 19 7 87 2016 ## 82 20 7 85 2016 ## 83 21 7 74 2016 ## 84 22 7 81 2016 ## 85 23 7 82 2016 ## 86 24 7 86 2016 ## 87 25 7 85 2016 ## 88 26 7 82 2016 ## 89 27 7 86 2016 ## 90 28 7 88 2016 ## 91 29 7 86 2016 ## 92 30 7 83 2016 ## 93 31 7 81 2016 ## 94 1 8 81 2016 ## 95 2 8 81 2016 ## 96 3 8 82 2016 ## 97 4 8 86 2016 ## 98 5 8 85 2016 ## 99 6 8 87 2016 ## 100 7 8 89 2016 ## 101 8 8 90 2016 ## 102 9 8 90 2016 ## 103 10 8 92 2017 ## 104 11 8 86 2017 ## 105 12 8 86 2017 ## 106 13 8 82 2017 ## 107 14 8 80 2017 ## 108 15 8 79 2017 ## 109 16 8 77 2017 ## 110 17 8 79 2017 ## 111 18 8 76 2017 ## 112 19 8 78 2017 ## 113 20 8 78 2017 ## 114 21 8 77 2017 ## 115 22 8 72 2017 ## 116 23 8 75 2017 ## 117 24 8 79 2017 ## 118 25 8 81 2017 ## 119 26 8 86 2017 ## 120 27 8 88 2017 ## 121 28 8 97 2017 ## 122 29 8 94 2017 ## 123 30 8 96 2017 ## 124 31 8 94 2017 ## 125 1 9 91 2017 ## 126 2 9 92 2017 ## 127 3 9 93 2017 ## 128 4 9 93 2017 ## 129 5 9 87 2017 ## 130 6 9 84 2017 ## 131 7 9 80 2017 ## 132 8 9 78 2017 ## 133 9 9 75 2017 ## 134 10 9 73 2017 ## 135 11 9 81 2017 ## 136 12 9 76 2017 ## 137 13 9 77 2017 ## 138 14 9 71 2017 ## 139 15 9 71 2017 ## 140 16 9 78 2017 ## 141 17 9 67 2017 ## 142 18 9 76 2017 ## 143 19 9 68 2017 ## 144 20 9 82 2017 ## 145 21 9 64

```
2017 ## 146 22 9 71 2017 ## 147 23 9 81 2017 ## 148 24 9 69
2017 ## 149 25 9 63 2017 ## 150 26 9 70 2017 ## 151 27 9 77
2017 ## 152 28 9 75 2017 ## 153 29 9 76 2017 ## 154 30 9 68
2017 ## 155 31 9 unknown 2017

#####################  #  #  #  Exercise  10  #  #  #
#################### back2long_na %$% extract_numeric(Temp)

## [1] 67 72 74 62 56 66 65 59 61 69 74 69 66 68 58 64 66 57
68 62 59 73 61 ## [24] 61 57 58 57 67 81 79 76 78 74 67 84 85
79 82 87 90 87 93 92 82 80 79 ## [47] 77 72 65 73 76 77 76 76
76 75 78 73 80 77 83 NA 84 85 81 84 83 83 88 ## [70] 92 92 89
82 73 81 91 80 81 82 84 87 85 74 81 82 86 85 82 86 88 86 83 ##
[93] 81 81 81 82 86 85 87 89 90 90 92 86 86 82 80 79 77 79 76
78 78 77 72 ## [116] 75 79 81 86 88 97 94 96 94 91 92 93 93 87
84 80 78 75 73 81 76 77 71 ## [139] 71 78 67 76 68 82 64 71 81
69 63 70 77 75 76 68 NA
```

Data wrangling : I/O (Part-2)-Solutions

Below are the solutions to these exercises on data importing and exporting.

```
#####################  #  #  #  Exercise  1  #  #  #
#####################  url  <-
getURL("https://raw.githubusercontent.com/VasTsak/r-exercises-
dw/master/part-1/data.csv")  ###################  #  #  #
Exercise  2  #  #  #  ###################  csv_file  <-
read.csv(text = url) ################## # # # Exercise 3 # #
#  ###################  url  <-
getURL("https://raw.githubusercontent.com/VasTsak/r-exercises-
dw/master/part-2/data.txt") txt_file <- read.table(text = url)
####################  #  #  #  Exercise  4  #  #  #
```

```
#####################                url                <-
getURL("https://raw.githubusercontent.com/VasTsak/r-exercises-
dw/master/part-2/data.json")  json_file  <-  fromJSON(url)
#####################  #  #  #  Exercise  5  #  #  #
#####################                url                <-
getURL("https://raw.githubusercontent.com/VasTsak/r-exercises-
dw/master/part-2/data.xml") xml_file <- ldply(xmlToList(url),
data.frame)

#####################  #  #  #  Exercise  6  #  #  #
#####################                url                <-
read_html("http://www.worldatlas.com/articles/largest-cities-i
n-europe-by-population.html") ##################### # # #
Exercise 7 # # # ##################### tbls <- html_nodes(url,
"table") ##################### # # # Exercise 8 # # #
##################### tbls_read <- url %>% html_nodes("table")
%>% html_table(fill = TRUE) ##################### # # #
Exercise  9  #  #  #  ######################  url  <-
"http://www.worldatlas.com/articles/largest-cities-in-europe-b
y-population.html"  tbls_xml  <-  readHTMLTable(url)
#####################  #  #  #  Exercise  10  #  #  #
##################### df_pop <- htmltab(doc = url, which =
"//th[text() = 'Rank']/ancestor::table")
```

Data wrangling : I/O (Part-2)

Data wrangling is a task of great importance in data analysis. Data wrangling, is the process of importing, cleaning and transforming raw data into actionable information for analysis. It is a time-consuming

process which is estimated to take about 60-80% of analyst's time. In this series we will go through this process. It will be a brief series with goal to craft the reader's skills on the data wrangling task. This is the first part of this series and it aims to cover the importing of data from the web. In many cases, downloading data in order to process them can be time consuming, therefore being able to import the data straight from the web is a 'nice-to-have' skill. Moreover, data isn't always not saved in structured files, but they are on the web in forms of text and tables, in this set of exercise we will go through the latter case. In case you want me to go through the former case as well, please let me know at the comment section.

Before proceeding, it might be helpful to look over the help pages for the getURL, fromJSON, ldply, xmlToList, read_html, html_nodes, html_table, readHTMLTable, htmltab.

Moreover please load the following libraries.

```
install.packages("RCurl")
library(RCurl)
install.packages("rjson")
library(rjson)
install.packages("XML")
library(XML)
install.packages("plyr")
library(plyr)
install.packages("rvest")
library(rvest)
install.packages("htmltab")
library(htmltab)
```

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

Retrieve the source of the web page "https://raw.githubusercontent.com/VasTsak/r-exercises-dw/master/part-1/data.csv" and assign it to the object "url"

Exercise 2

Read the csv file and assign it to the "csv_file" object.

Exercise 3

Do the same as exercise 1, but with the url: "https://raw.githubusercontent.com/VasTsak/r-exercises-dw/master/part-2/data.txt" and then assign it to the "txt_file" object.
Note: it is a txt file, so you should use the adequate function in order to import it.

Exercise 4

Do the same as exercise 1, but with the url: "https://raw.githubusercontent.com/VasTsak/r-exercises-dw/master/part-2/data.json" and then assign it to the "json_file" object.
Note: it is a json file, so you should use the adequate function in order to import it.

Exercise 5

Do the same as exercise 1, but with the url: “https://raw.githubusercontent.com/VasTsak/r-exercises-dw/master/part-2/data.xml” and then assign it to the “xml_file” object.
Note: it is a xml file, so you should use the adequate function in order to import it.

Exercise 6

We will go through web scraping now. Read the html file “http://www.worldatlas.com/articles/largest-cities-in-europe-by-population.html” and assign it to the object “url”.
hint: consider using read_html

Exercise 7

Select the “table” nodes from the html document you retrieved before.
hint: consider using html_nodes

Exercise 8

Convert the node you retrieved at exercise 7, to an actionable list for processing.
hint: consider using html_table

Exercise 9

Let’s go to a faster and more straight forward function, retrieve the html document like you did at exercise 6 and make it an actionable list using the function readHTMLTable.

Exercise 10

This may be a bit tricky, but give it a try. Retrieve the html document like you did at exercise 6 and make it an actionable data frame using the function htmltab.

Data wrangling : I/O (Part-1) – Solutions

Below are the solutions to these exercises on data importing and exporting.

```
#####################  #  #  #  Exercise  1  #  #  #
#################### csv_file <- read.csv("data.csv", header =
TRUE) ###################### # # # Exercise 2 # # #
#################### txt_file <- read.table("data.txt")
#####################  #  #  #  Exercise  3  #  #  #
#################### xls_file <- read_excel("data.xls", sheet
=  1)  #OR    xls_file  <-  read.xlsx("data.xls",1)
#####################  #  #  #  Exercise  4  #  #  #
#################### xlsx_file <- read_excel("data.xlsx",
sheet = 1) #OR  xlsx_file <- read.xlsx("data.xlsx",1)
#####################  #  #  #  Exercise  5  #  #  #
#################### json_file <- fromJSON(file="data.json") #
It returns a list, you will probably require a data frame,
therefore use  # the command below in order to transform it.
json_file <- as.data.frame(json_file) ################### # #
# Exercise 6 # # # ##################### xml_file <-
ldply(xmlToList("data.xml"), data.frame) ###################
#  #  #  Exercise  7  #  #  #  ####################
write.csv(csv_file,  file  =  "data.csv")  #OR
write.table(csv_file, file = "data.csv", sep = ",")
write.table(txt_file,  file  =  "data.txt",  sep="\t")
#####################  #  #  #  Exercise  8  #  #  #
####################  write.xlsx(xls_file,  "data.xls")
write.xlsx(xlsx_file, "data.xlsx") ################### # # #
Exercise 9 # # # ##################### data_json <-
toJSON(json_file)  write(data_json,  "data.json")
#####################  #  #  #  Exercise  10  #  #  #
#################### write.xml(xml_file, file="data.xml")
```

Data wrangling : I/O (Part-1)

Data wrangling is a task of great importance in data analysis. Data wrangling, is the process of importing, cleaning and transforming raw data into actionable information for analysis. It is a time-consuming process which is estimated to take about 60-80% of analyst's time. In this series we will go through this process. It will be a brief series with goal to craft the reader's skills on the data wrangling task. This is the first part of this series and it aims to cover the importing and exporting of data locally.
Please download the data from here

Before proceeding, it might be helpful to look over the help pages for the read.csv, read.table, read_excel, fromJSON, as.data.frame, xmlParse, xmlToList, write.csv, write.table, write.xlsx, toJSON, write, write.xml.

Moreover please load the following libraries.

```
install.packages("readxl")
library(readxl)
install.packages("xlsx")
library(xlsx)
install.packages("rjson")
library(rjson)
install.packages("plyr")
library(plyr)
install.packages("XML")
library(XML)
install.packages("kulife")
```

```
library(kulife)
```

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

Import the data.csv file to the csv_file object.

Exercise 2

Import the data.txt file to the txt_file object.

Exercise 3

Import the data.xls file to the xls_file object.

Exercise 4

Import the data.xlsx file to the xlsx_file object.

Exercise 5

Import the data.json file to the json_file object.

Exercise 6

Import the data.xml file to the xml_file object.

Exercise 7

Export the csv_file as “data.csv” and txt_file as “data.txt”.

Exercise 8

Export the xls_file as “data.xls” and xlsx_file as “data.xlsx”.

Exercise 9

Export the json_file as “data.json”.

Exercise 10

Export the xml_file as “data.xml”.

Data science for Doctors: Variable importance Exercises

Data science enhances people’s decision making. Doctors and researchers are making critical decisions every day. Therefore, it is absolutely necessary for those people to have some basic knowledge of data science. This series aims to help people that are around medical field to enhance their data science skills.

We will work with a health related database the famous “Pima Indians Diabetes Database”. It was generously donated by Vincent Sigillito from Johns Hopkins University. Please find further information regarding the dataset here.

This is the tenth part of the series and it aims to cover the

very basics of the subject of principal correlation coefficient and components analysis, those two methods illustrate how variables are related.
In my opinion, it is necessary for researchers to know how to have a notion of the relationships between variables, in order to be able to find potential cause and effect relation – however this relation is hypothetical, you can't claim that there is a cause-effect relation only because the correlation is high between those two variables-,remove unecessary variables etc. In particular we will go through Pearson correlation coefficient and Confidence interval by the bootstrap and (Principal component analysis.

Before proceeding, it might be helpful to look over the help pages for the ggplot, cor, cor.tes, boot.cor, quantile, eigen, princomp, summary, plot, autoplot.

Moreover please load the following libraries.

```
install.packages("ggplot2")
library(ggplot2)
install.packages("ggfortify")
library(ggfortify)
```

Please run the code below in order to load the data set and transform it into a proper data frame format:

```
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"
data <- read.table(url, fileEncoding="UTF-8", sep=",")
names <- c('preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class')
colnames(data) <- names
data <- data[-which(data$mass ==0),]
```

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

Compute the value of the correlation coefficient for the variables age and preg.

Exercise 2

Construct the scatterplot for the variables age and preg.

Exercise 3

Apply a correlation test for the variables age and preg with null hypothesis to be the correlation is zero and the alternative to be different from zero.
hint: cor.test

Exercise 4

Construct a 95% confidence interval is by the bootstrap. First find the correlation by bootstrap.
hint: mean

Exercise 5

Now that you have found the correlation, find the 95% confidence interval.

Exercise 6

Find the eigen values and eigen vectors for the data set(exclude the class.fac variable).

Exercise 7

Compute the principal components for the dataset used above.

Exercise 8

Show the importance of each principal component.

Exercise 9

Plot the principal components using an elbow graph.

Exercise 10

Constract a scatterplot with x-axis to be the first component and the y-axis to be the second component. Moreover if possible draw the eigen vectors on the plot.
hint: autoplot

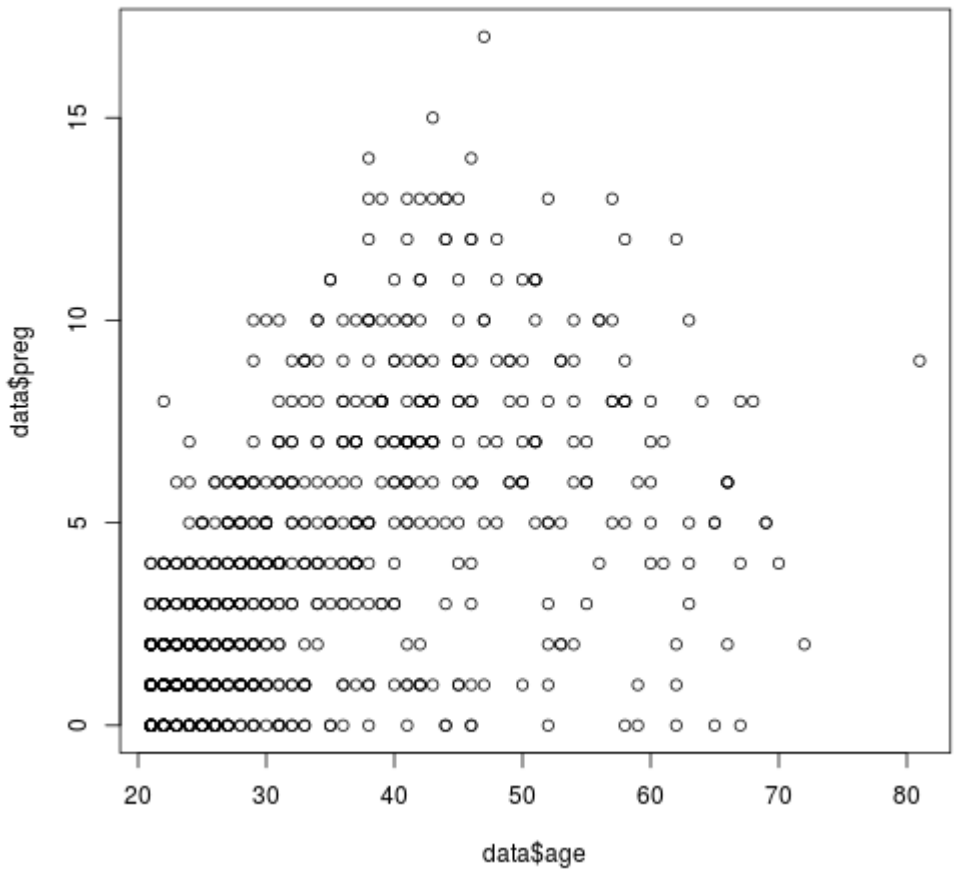
Data science for Doctors: Variable importance Solutions

Below are the solutions to these exercises on principal component analysis.

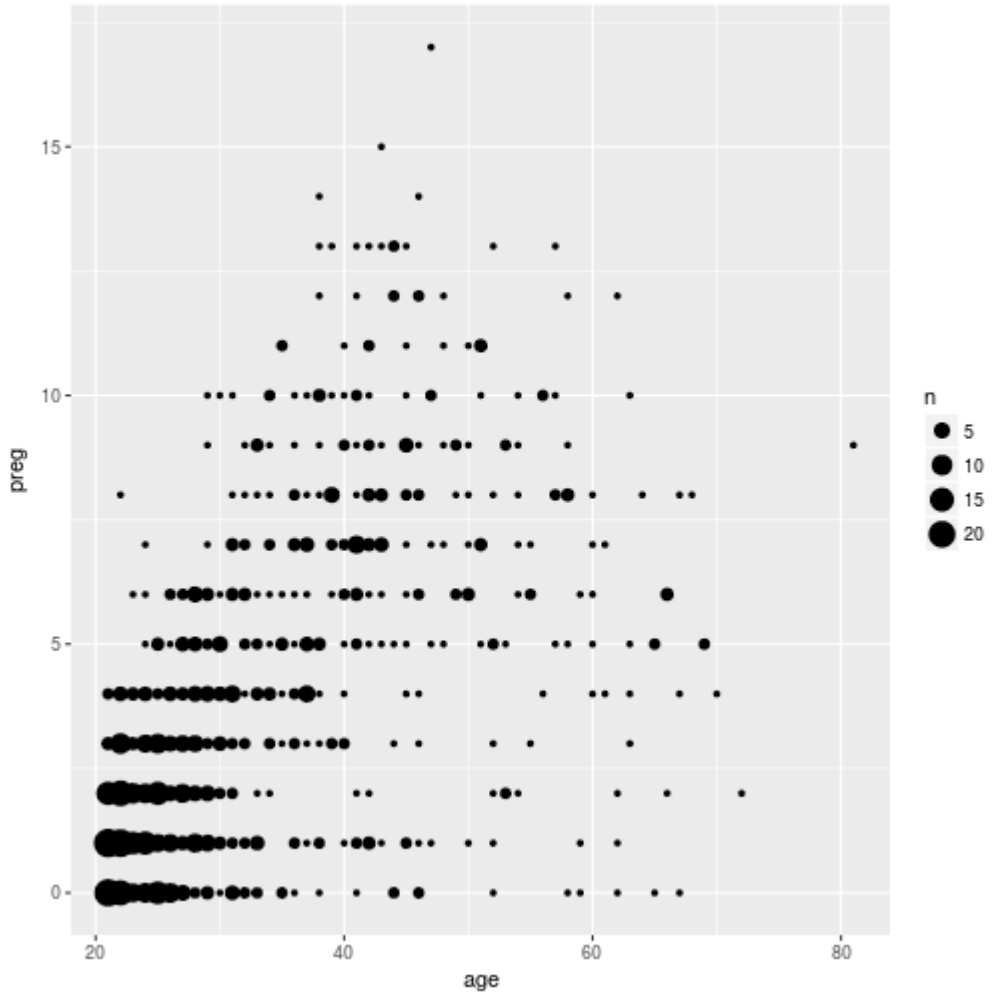
```
#####################  #  #  #  Exercise  1  #  #  #
#####################
cor(data$age,data$preg)
```

```
## [1] 0.5443412
```

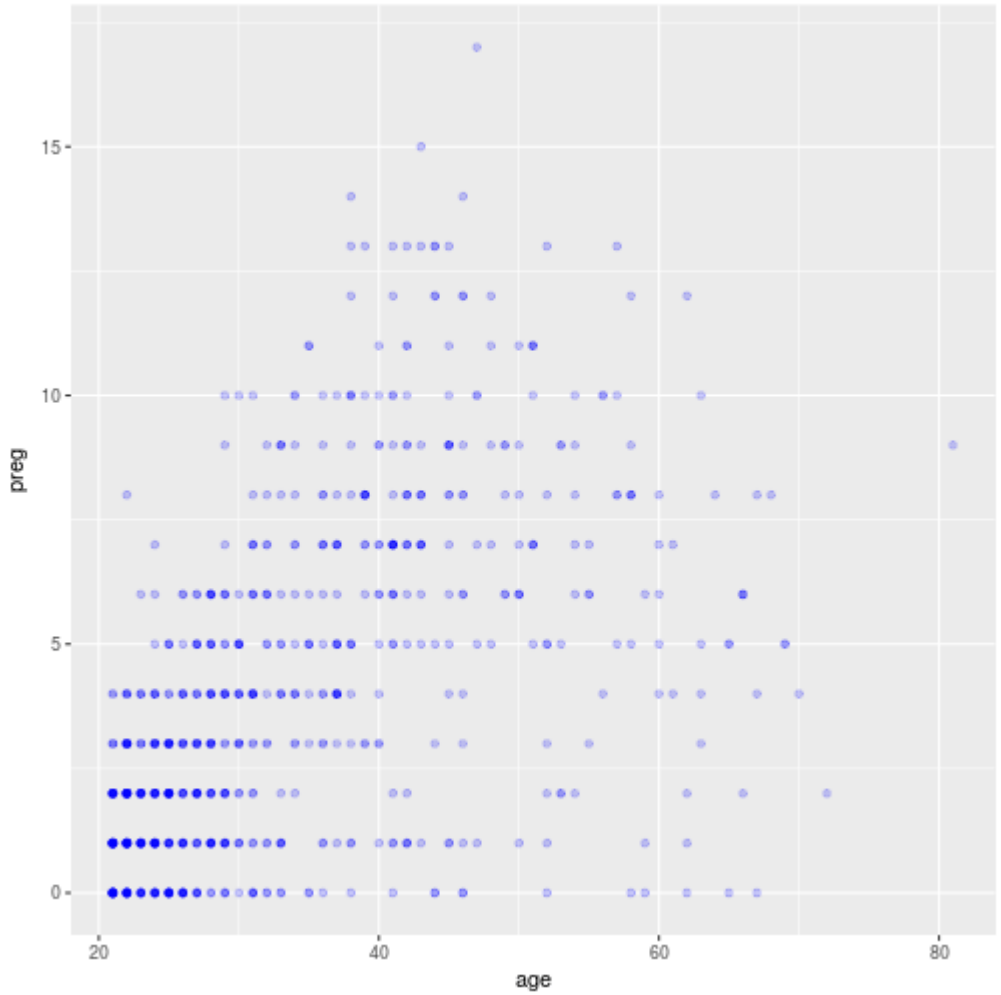
```
#####################  #  #  #  Exercise  2  #  #  #
#####################
plot(data$age,data$preg)
```




```
#OR ggplot(data,aes(x = age, y = preg))+ geom_count() # we use
geom_count() to capture the frequency.
```




```
#OR ggplot(data,aes(x = age, y = preg))+ geom_point(colour = "blue", alpha = 0.2) # we use transparent points to capture the frequency. I personaslly prefer the this scatterplot because it captures the frequency but in a cleaner way.
```




```
######################  #  #  #  Exercise  3  #  #  #
#################### cor.test(data$age,data$preg)
```

Pearson's product-moment correlation ## ## data: data$age and data$preg ## t = 17.959, df = 766, p-value < 2.2e-16 ## alternative hypothesis: true correlation is not equal to 0 ## 95 percent confidence interval: ## 0.4925652 0.5922775 ## sample estimates: ## cor ## 0.5443412

```
######################  #  #  #  Exercise  4  #  #  #
#####################  nboot  <-  1000;  boot.cor  <-
matrix(,nrow=nboot,ncol  =  1)  data_mat  <-
matrix(c(data$age,data$preg),ncol=2,byrow=FALSE) for (i in
1:nboot){                  dat.star                  <-
data_mat[sample(1:nrow(data_mat),replace=TRUE),] boot.cor[i,]
<- cor(dat.star)[2,1]} mean(boot.cor)
```

[1] 0.5437089

```
#####################  #  #  #  Exercise  5  #  #  #
#################### quantile(boot.cor[,1],c(0.025,0.975))
```

```
## 2.5% 97.5% ## 0.4891308 0.5950294
```

```
#####################  #  #  #  Exercise  6  #  #  #
#################### Z <- data.matrix(data[-ncol(data)]) K <-
eigen(cor(Z)) K$values; K$vectors
```

```
## [1] 2.0943799 1.7312101 1.0296299 0.8755290 0.7623444
0.6826284 0.4198162 ## [8] 0.4044620
```

```
## [,1] [,2] [,3] [,4] [,5] [,6] ## [1,] -0.1284321 -0.5937858
0.01308692  0.08069115  0.4756057  -0.193598168  ##  [2,]
-0.3930826  -0.1740291  -0.46792282  -0.40432871  -0.4663280
-0.094161756  ##  [3,]  -0.3600026  -0.1838921  0.53549442
0.05598649 -0.3279531 0.634115895 ## [4,] -0.4398243 0.3319653
0.23767380  0.03797608  0.4878621  -0.009589438  ##  [5,]
-0.4350262  0.2507811  -0.33670893  -0.34994376  0.3469348
0.270650609 ## [6,] -0.4519413 0.1009598 0.36186463 0.05364595
-0.2532038  -0.685372179  ##  [7,]  -0.2706114  0.1220690
-0.43318905  0.83368010  -0.1198105  0.085784088  ##  [8,]
-0.1980271  -0.6205885  -0.07524755  0.07120060  0.1092900
0.033357170 ## [,7] [,8] ## [1,] 0.58879003 0.117840984 ##
[2,] 0.06015291 0.450355256 ## [3,] 0.19211793 -0.011295538 ##
[4,] -0.28221253 0.566283799 ## [5,] 0.13200992 -0.548621381
##  [6,]  0.03536644  -0.341517637  ##  [7,]  0.08609107
-0.008258731 ## [8,] -0.71208542 -0.211661979
```

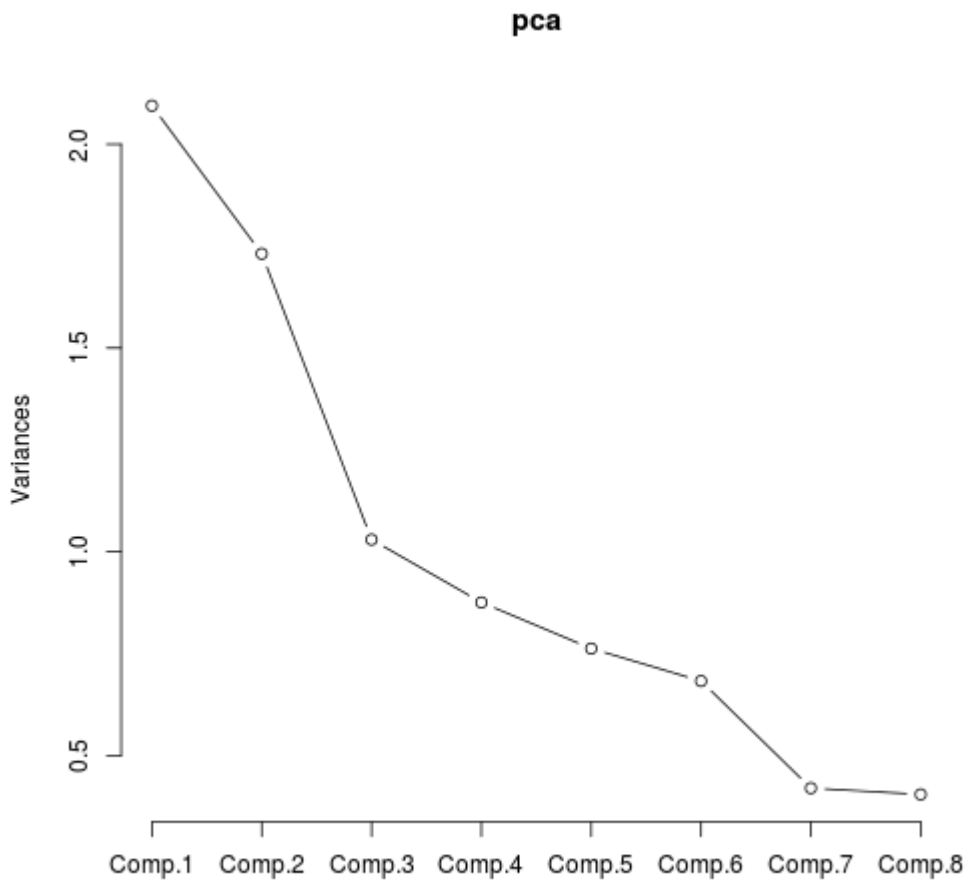
```
#####################  #  #  #  Exercise  7  #  #  #
#################### pca <- princomp(Z, center = TRUE,
cor=TRUE, scores=TRUE)
```

```
## Warning: In princomp.default(Z, center = TRUE, cor = TRUE,
scores = TRUE) : ## extra argument 'center' will be
disregarded
```

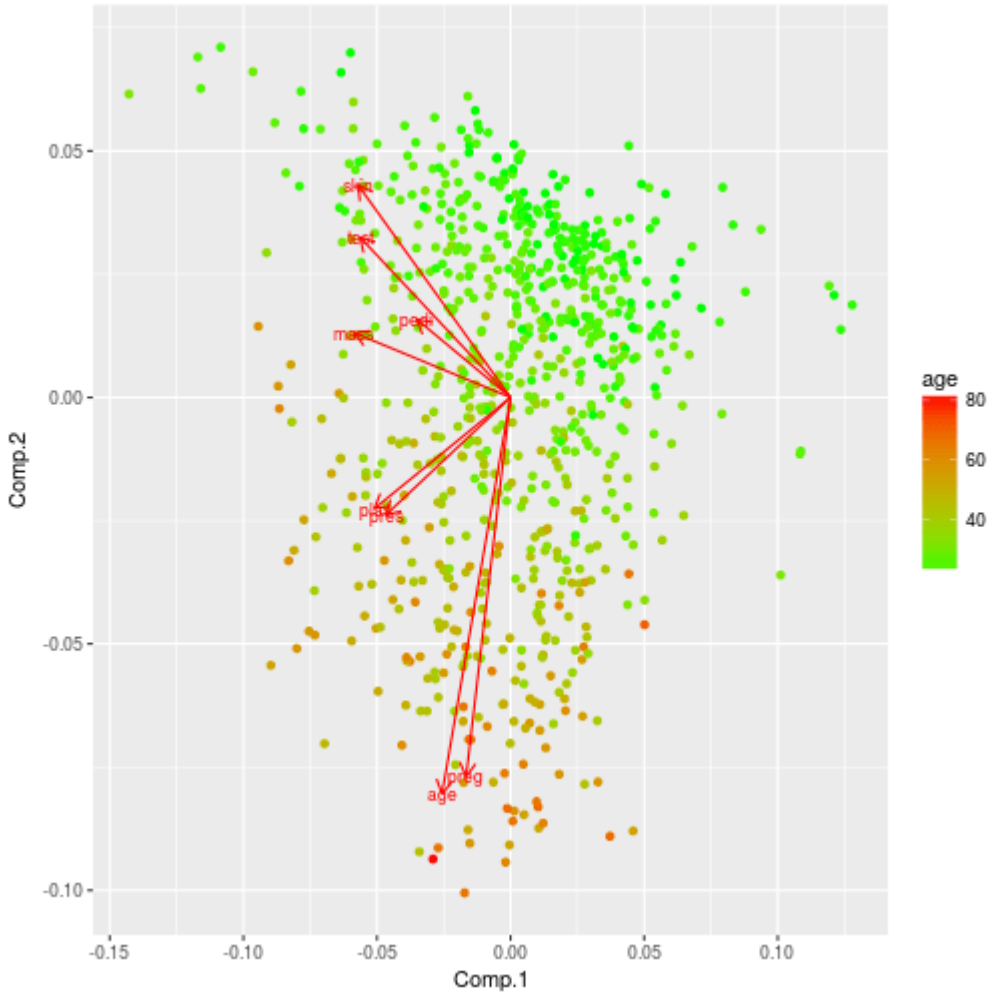
```
#####################  #  #  #  Exercise  8  #  #  #
#################### summary(pca)
```

Importance of components: ## Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 ## Standard deviation 1.4471973 1.3157546 1.0147068 0.9356971 0.87312335 ## Proportion of Variance 0.2617975 0.2164013 0.1287037 0.1094411 0.09529305 ## Cumulative Proportion 0.2617975 0.4781988 0.6069025 0.7163436 0.81163667 ## Comp.6 Comp.7 Comp.8 ## Standard deviation 0.82621328 0.64793223 0.63597331 ## Proportion of Variance 0.08532855 0.05247702 0.05055776 ## Cumulative Proportion 0.89696522 0.94944224 1.00000000

########## # # # Exercise 9 # # # ########## plot(pca,type="l")



########## # # # Exercise 10 # # # ########## autoplot(pca, data = data, colour = 'age', loadings = TRUE, loadings.label = TRUE, loadings.label.size = 3) + scale_color_gradient(low="green", high="red")



Data science for Doctors: Cluster Analysis Exercises

Data science enhances people's decision making. Doctors and researchers are making critical decisions every day. Therefore, it is absolutely necessary for those people to have some basic knowledge of data science. This series aims to help people that are around medical field to enhance their data science skills.

We will work with a health related database the famous "Pima

Indians Diabetes Database". It was generously donated by Vincent Sigillito from Johns Hopkins University. Please find further information regarding the dataset here.

This is the ninth part of the series and it aims to cover the very basics of the subject of cluster analysis.
In my opinion, it is necessary for researchers to know how to discover relationships between patients and diseases. Therefore in this set of exercises we will go through the basics of cluster analysis relationship discovery. In particular we will use hierarchical clustering and centroid-based clustering , k-means clustering and k-median clustering.

Before proceeding, it might be helpful to look over the help pages for the ggplot, geom_point, dist, hclust, cutree, stats::rect.hclust, multiplot, kmeans, kGmedian.

Moreover please load the following libraries.

```
install.packages("ggplot2")
library(ggplot2)
install.packages("Gmedian")
library(Gmedian)
```

Please run the code below in order to load the data set and transform it into a proper data frame format:

```
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"
data <- read.table(url, fileEncoding="UTF-8", sep=",")
names <- c('preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class')
colnames(data) <- names
data <- data[-which(data$mass ==0),]
```

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

Construct a scatterplot with x-axis to be the mass variable and y-axis to be the age variable. Moreover, determine the colour of the points based on the class of the candidate (0 or 1).

Exercise 2

Create a distance matrix for the data.

Exercise 3

Make an hierarchical clustering analysis using the single linkage method. Then create an object that contains only two clusters.

Exercise 4

Make an hierarchical clustering analysis using the complete linkage method(default). Then create an object that contains only two clusters.

Exercise 5

Construct the trees that are produced by exercises 2 and 3 and draw the two clusters(at the plots).
hint: rect.hclust

Exercise 6

Construct two scatterplot with x-axis to be the mass variable and y-axis to be the age variable. Moreover, determine the

colour of the points based on the cluster that those points belong to. Each scatterplot is for different clustering method.

If possible illustrate those scatterplots (each one at a time) next to the plot of exercise 1, to see whether the clustering can discriminate the positive classified from the negative classified patients. In case you didn’t do that, find it at the solution’s section, I highly encourage you to check it out.

Exercise 7

Run the following in order to create dummy variables data_mat <- model.matrix(~.+0, data = data).
Make a centroid-based cluster analysis using the k-means method with k to be 2. Apply the k-mean clustering on the data_mat data frame.

Exercise 8

Construct a scatterplot with x-axis to be the mass variable and y-axis to be the age variable. Moreover, determine the colour of the points based on the cluster (retrieved from k-mean method) that those points belong to.

If possible illustrate those scatterplot next to the plot of exercise 1.

Exercise 9

Make a centroid-based cluster analysis using the k-median method with k to be 2. Apply the k-median clustering on the data_mat data frame.

Exercise 10

Construct a scatterplot with x-axis to be the mass variable and y-axis to be the age variable. Moreover, determine the colour of the points based on the cluster (retrieved from k-

median method) that those points belong to.

If possible illustrate those scatterplot next to the plot of exercise 1.

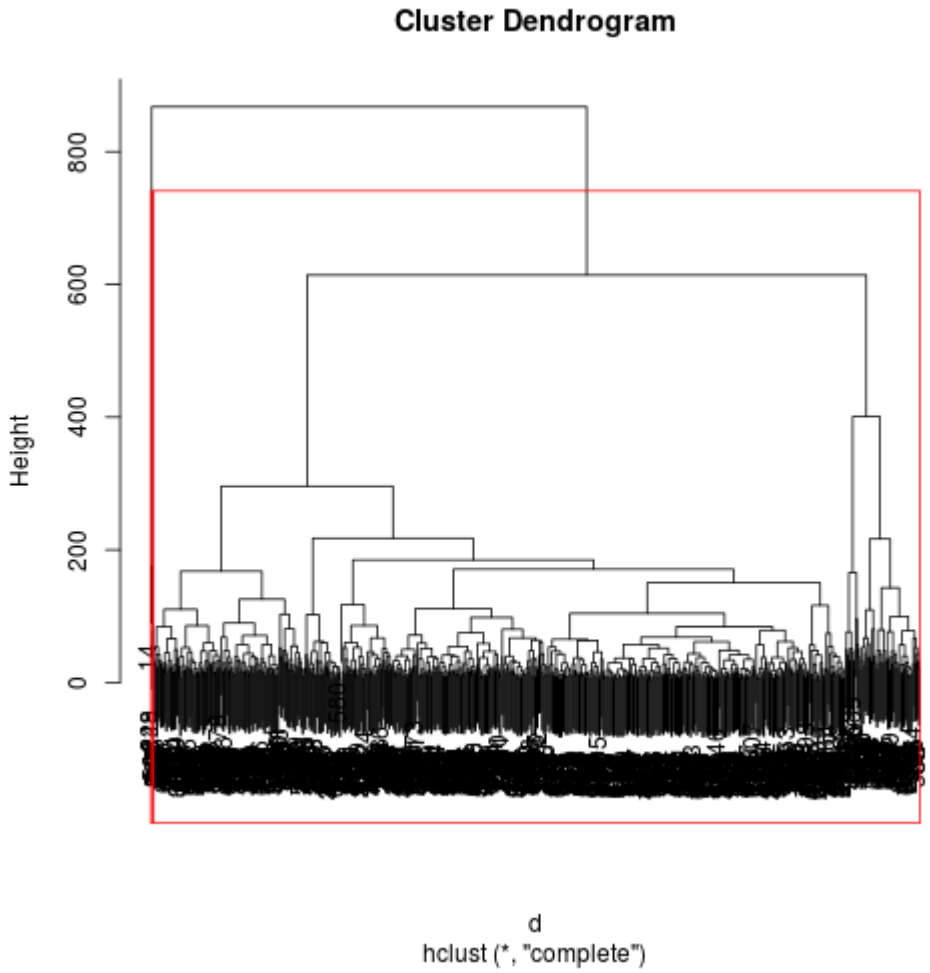
Data science for Doctors: Cluster Analysis Exercises Solutions

Below are the solutions to these exercises on cluster analysis.

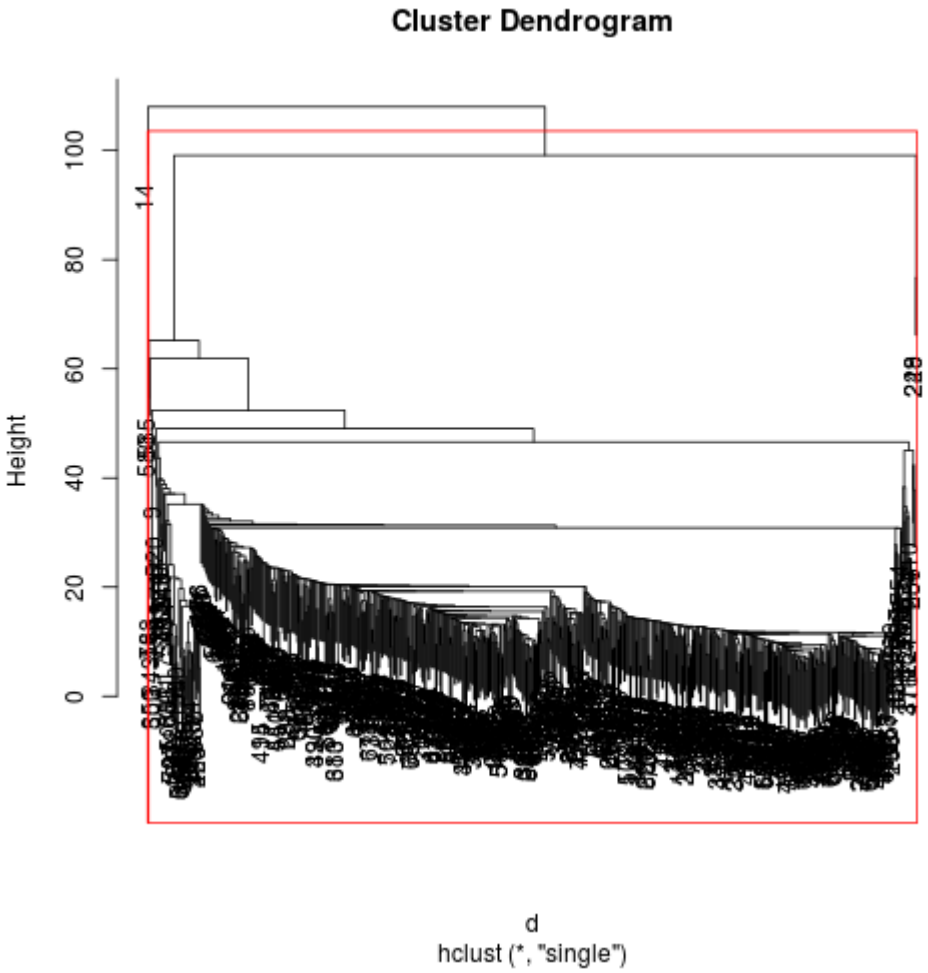
In order to run the multiplot function , run this script first

```
#####################  #  #  #  Exercise  1  #  #  #  ##################### p1 <- ggplot(data, aes(mass, age, color = class))  +  geom_point()+  scale_colour_gradient(low="green", high="red") p1
```

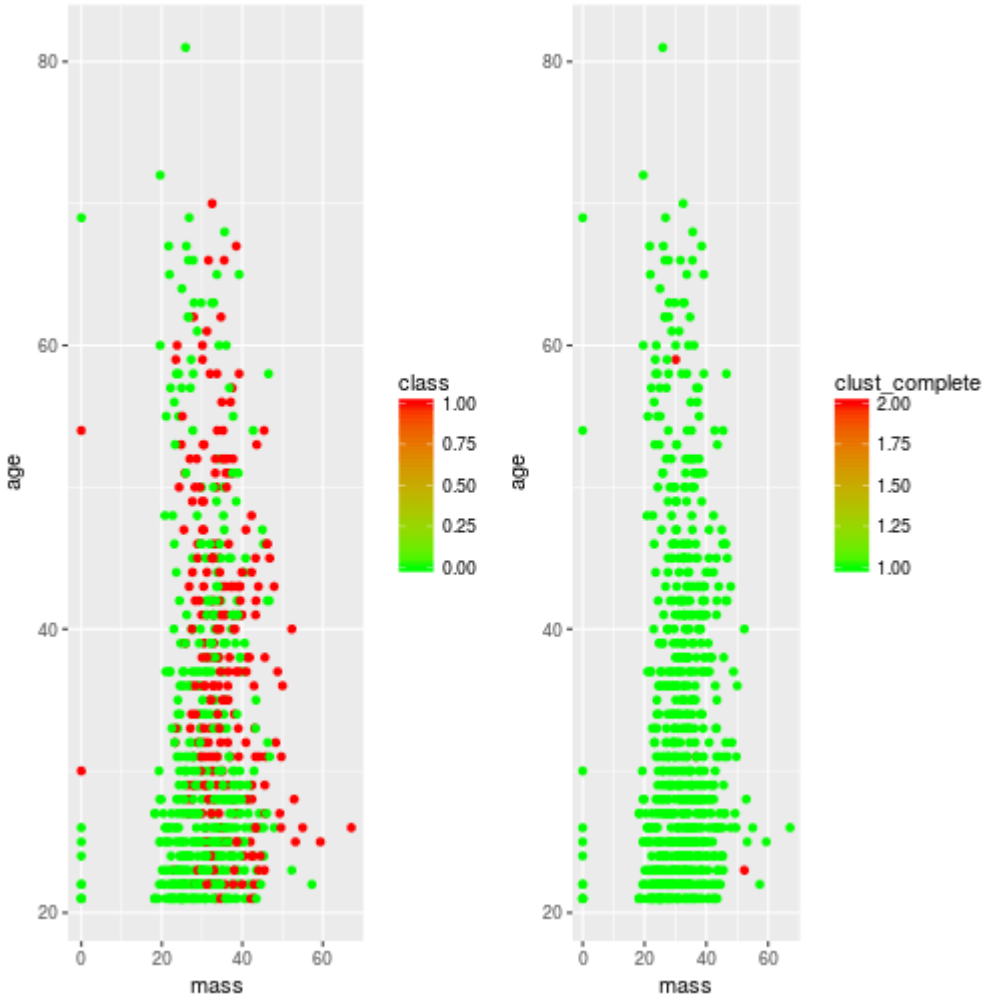

```
######################  #  #  #  Exercise  2  #  #  #
######################  d  <-  dist(as.matrix(data))
######################  #  #  #  Exercise  3  #  #  #
#################### single_link <- hclust(d,method="single")
clust_sing <- cutree(single_link, k = 2) ####################
# # # Exercise 4 # # # #################### complete_link <-
hclust(d,method="complete")  clust_complete  <-
cutree(complete_link, k = 2) ####################  # # #
Exercise 5 # # # #################### plot(complete_link)
stats::rect.hclust(complete_link, 2)
```



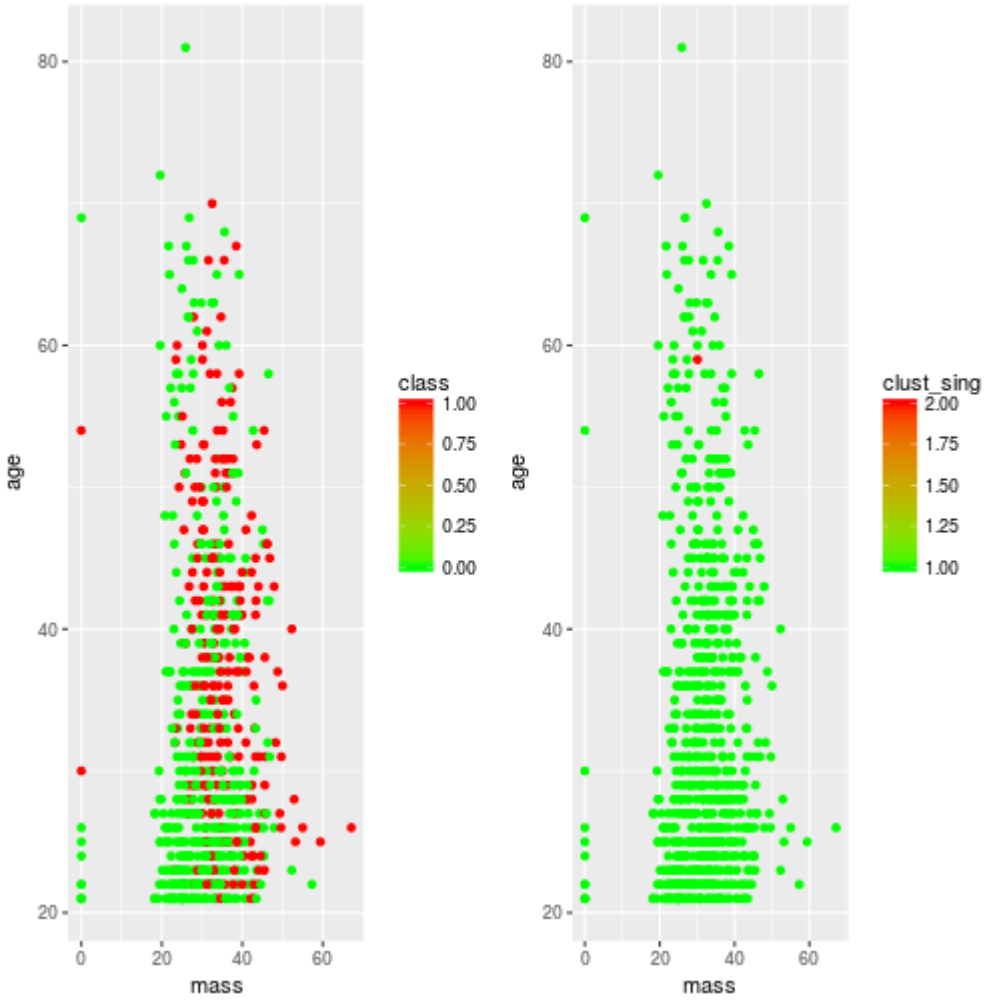

```
plot(single_link) stats::rect.hclust(single_link, 2)
```



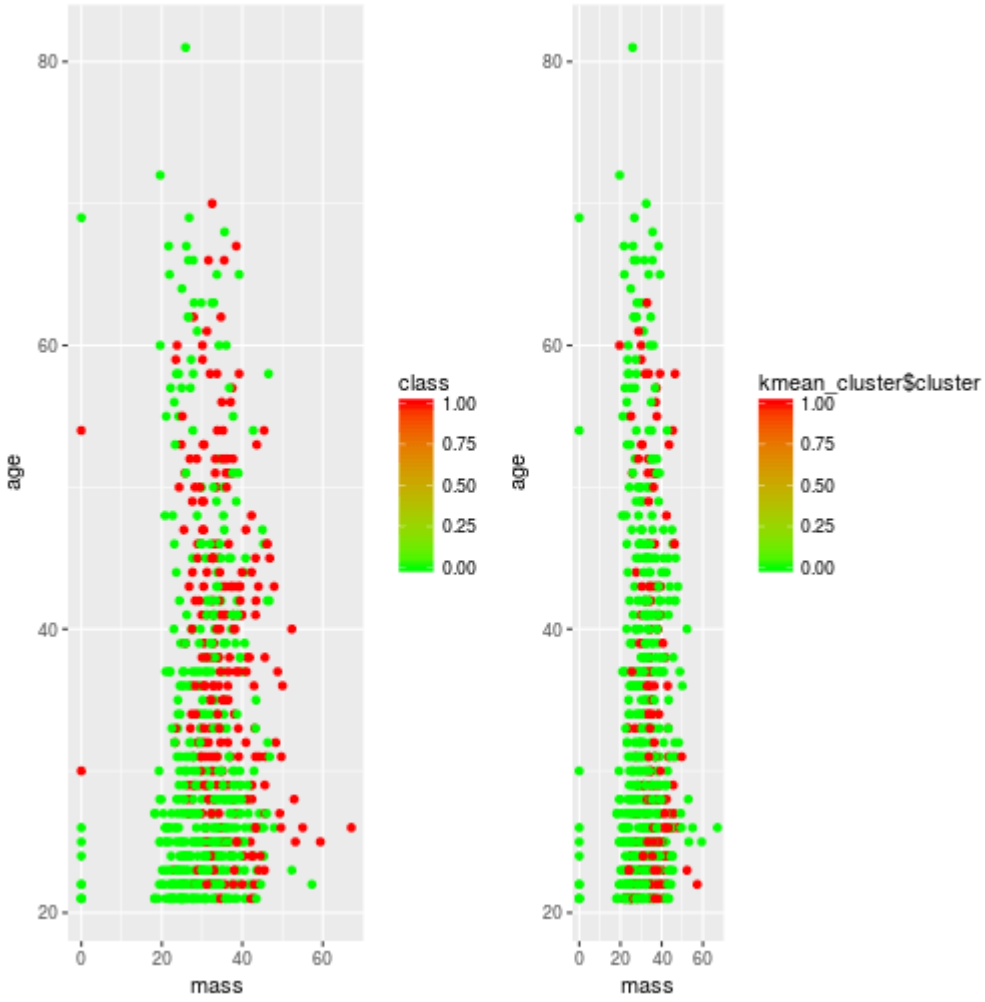

```
######################  #  #  #  Exercise  6  #  #  #
###################### p2 <-ggplot(data, aes(mass, age, color =
clust_complete))          +          geom_point()+
scale_color_gradient(low="green",  high="red")  p3  <-
ggplot(data,  aes(mass,  age,  color  =  clust_sing))  +
geom_point()+  scale_color_gradient(low="green",  high="red")
multiplot(p1,p2, cols = 2)
```



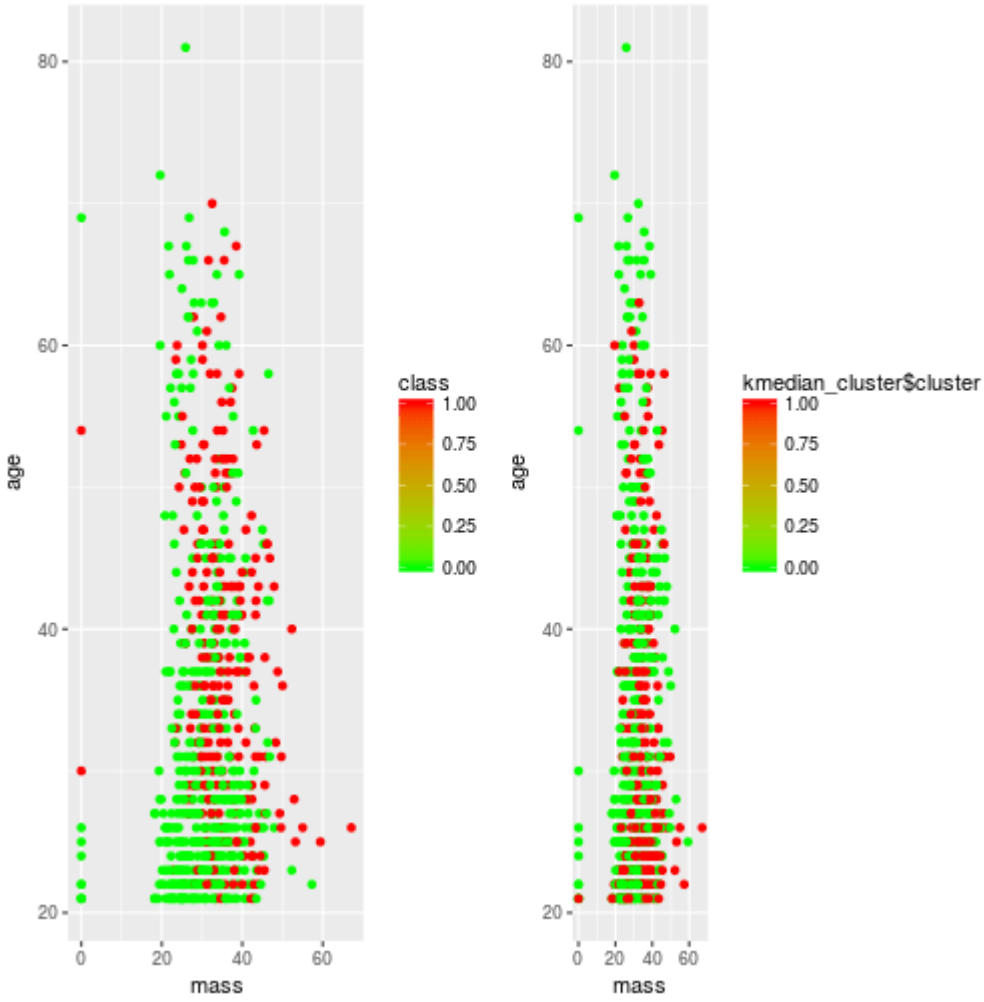

```
multiplot(p1,p3, cols = 2)
```




```
###################### # # # Exercise 7 # # # ###################### kmean_cluster <- kmeans(data_mat, 2, nstart = 20) # initializes the centroids 20 times and selects the one with the lowest within cluster variation ###################### # # # Exercise 8 # # # ###################### kmean_cluster$cluster <- kmean_cluster$cluster -1 # You are not required to do it, but I did that for scaling purposes. p4 <- ggplot(data, aes(mass, age, color = kmean_cluster$cluster)) + geom_point()+ scale_color_gradient(low="green", high="red") multiplot(p1,p4, cols = 2)
```




```
###################### # # # Exercise 9 # # # ###################### kmedian_cluster <-kGmedian(data_mat, ncenters=2, nstart = 20) ###################### # # # Exercise 10 # # # ###################### kmedian_cluster$cluster <- kmedian_cluster$cluster -1 p5 <- ggplot(data, aes(mass, age, color = kmedian_cluster$cluster)) + geom_point()+ scale_color_gradient(low="green", high="red") multiplot(p1,p5, cols = 2)
```


age
mass
class
1.00
0.75
0.50
0.25
0.00
kmedian_cluster$cluster