Assignment 4 - CT5102 Functionals (apply) and Matrices

Chin Zhe Jing 22221970

r1615473

2022-10-06

The goal of this assignment is to create a matrix of examinations results, and use the apply() function to: (1) clean the data; (2) impute missing values; and (3) generate summary statistics for each student.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

First, the synthetic data must be setup as follows (five subjects (CX101:CX108) and 20 students:

```
set.seed(100)
CX101 <- rnorm(20,45,8)
CX102 <- rnorm(20,65,8)
CX103 <- rnorm(20,85,10)
CX104 <- rnorm(20,45,10)
CX105 <- rnorm(20,60,5)
```

Create a matrix from this raw data, and confirm that it has the following summaries, and also that the row names related to a student (Student_1 through to Student_20)

```
res <- c(CX101, CX102, CX103, CX104, CX105) %>% matrix(nrow = 20, ncol=5,
             dimnames = list(c(paste0("Student_", 1:20)),
                             c("CX101","CX102","CX103", "CX104", "CX105")))

res %>% summary()
```

```
##      CX101           CX102           CX103            CX104
##  Min.   :37.69   Min.   :55.74   Min.   : 62.28   Min.   :24.26
##  1st Qu.:42.06   1st Qu.:61.49   1st Qu.: 75.21   1st Qu.:36.99
##  Median :45.74   Median :65.37   Median : 84.71   Median :44.31
##  Mean   :45.86   Mean   :65.74   Mean   : 84.82   Mean   :43.95
##  3rd Qu.:47.93   3rd Qu.:69.03   3rd Qu.: 98.20   3rd Qu.:48.15
##  Max.   :63.48   Max.   :79.06   Max.   :103.97   Max.   :70.82
##      CX105
##  Min.   :50.34
##  1st Qu.:55.20
##  Median :59.35
```

```
##  Mean   :59.69
##  3rd Qu.:63.68
##  Max.   :72.23
```

Notice that subject CX103 has a number of invalid values (> 100)

```r
res[res[,"CX103"] > 100,]
```

```
##               CX101    CX102    CX103    CX104    CX105
## Student_14 50.91872 64.11045 103.9747 61.48522 54.21214
## Student_18 49.08685 68.33859 103.2487 47.70539 62.06760
```

Using the apply() functional to iterate through each column, convert any outliers (< 0 or > 100) to the symbol NA, and store the result in a new matrix res1.

```r
in_column <- function(x){
  if (x<0 || x>100){
    x <- NA
  }
  else {
    x
  }
}
res1 <- matrix(unlist(apply(res, 2, in_column)), nrow = 20, ncol=5,
               dimnames = list(c(paste0("Student_", 1:20)),
                               c("CX101","CX102","CX103", "CX104", "CX105")))
```

```
## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'

## Warning in x < 0 || x > 100: 'length(x) = 20 > 1' in coercion to 'logical(1)'
```

```r
res1
```

```
##               CX101    CX102     CX103    CX104    CX105
## Student_1  40.98246 61.49528  83.98371 42.38004 64.48411
## Student_2  46.05225 71.11248  99.03203 44.31156 59.75002
## Student_3  44.36866 67.09569  67.23224 41.21116 53.27325
## Student_4  52.09428 71.18724  91.22867 70.81959 50.34394
## Student_5  45.93577 58.48497  79.77717 46.29834 63.54791
## Student_6  47.54904 61.49240  98.22231 37.86975 59.21047
## Student_7  40.34567 59.23823  81.36560 51.37994 61.08184
```

```
## Student_8  50.71626 66.84756  98.19066 47.01692 64.08681
## Student_9  38.39792 55.73816  85.43779 44.30083 68.63588
## Student_10 42.12110 66.97661  66.21344 44.07510 59.48115
## Student_11 45.71909 64.27109  80.52938 49.48903 57.21439
## Student_12 45.77020 79.05900  67.61402 34.35644 67.14151
## Student_13 43.38693 63.89656  86.78865 33.37581 55.53521
## Student_14 50.91872 64.11045 103.97466 61.48522 54.21214
## Student_15 45.98704 59.47989  62.28075 24.37904 57.34852
## Student_16 44.76547 63.22565  94.80464 45.12750 72.22841
## Student_17 41.88917 66.46326  71.01174 34.12472 55.83752
## Student_18 49.08685 68.33859 103.24872 47.70539 62.06760
## Student_19 37.68949 73.52322  98.81299 55.08452 54.10658
## Student_20 63.48237 72.76162  76.61148 24.25595 54.12983
```

```
all_column_mean <- apply(res1, 2, mean, na.rm=TRUE)
all_column_mean[['CX103']]
```

```
## [1] 84.81803
```

Use apply() to replace the NA values (result sstored in res2) with mean of all other results for that subject (simple imputation)

```
cols <- colnames(res1)
impute_mean <- function(x){
  lapply(cols, function(col){
    ifelse(is.na(x[[col]]), all_column_mean[[col]], x[col])
  })
}
res2 <- matrix(unlist(apply(res1, 1, impute_mean)),
               nrow = 20,
               ncol=5,
               byrow = TRUE,
               dimnames = list(c(paste0("Student_", 1:20)),
              c("CX101","CX102","CX103", "CX104", "CX105")))
res2
```

```
##               CX101    CX102     CX103    CX104    CX105
## Student_1  40.98246 61.49528  83.98371 42.38004 64.48411
## Student_2  46.05225 71.11248  99.03203 44.31156 59.75002
## Student_3  44.36866 67.09569  67.23224 41.21116 53.27325
## Student_4  52.09428 71.18724  91.22867 70.81959 50.34394
## Student_5  45.93577 58.48497  79.77717 46.29834 63.54791
## Student_6  47.54904 61.49240  98.22231 37.86975 59.21047
## Student_7  40.34567 59.23823  81.36560 51.37994 61.08184
## Student_8  50.71626 66.84756  98.19066 47.01692 64.08681
## Student_9  38.39792 55.73816  85.43779 44.30083 68.63588
## Student_10 42.12110 66.97661  66.21344 44.07510 59.48115
## Student_11 45.71909 64.27109  80.52938 49.48903 57.21439
## Student_12 45.77020 79.05900  67.61402 34.35644 67.14151
## Student_13 43.38693 63.89656  86.78865 33.37581 55.53521
## Student_14 50.91872 64.11045 103.97466 61.48522 54.21214
## Student_15 45.98704 59.47989  62.28075 24.37904 57.34852
## Student_16 44.76547 63.22565  94.80464 45.12750 72.22841
## Student_17 41.88917 66.46326  71.01174 34.12472 55.83752
## Student_18 49.08685 68.33859 103.24872 47.70539 62.06760
## Student_19 37.68949 73.52322  98.81299 55.08452 54.10658
```

```
## Student_20 63.48237 72.76162  76.61148 24.25595 54.12983
```

For each student, then calculate the average and the range, and bind these to new columns into a new matrix

```
student_average <- apply(res2, 1, mean)
student_range <- apply(res2, 1, range)
range <- student_range[2,] - student_range[1,]
res3 <- cbind(res2, Mean=student_average, Range=range)
res3
```

```
##               CX101    CX102     CX103    CX104    CX105     Mean    Range
## Student_1  40.98246 61.49528  83.98371 42.38004 64.48411 58.66512 43.00125
## Student_2  46.05225 71.11248  99.03203 44.31156 59.75002 64.05167 54.72048
## Student_3  44.36866 67.09569  67.23224 41.21116 53.27325 54.63620 26.02108
## Student_4  52.09428 71.18724  91.22867 70.81959 50.34394 67.13474 40.88473
## Student_5  45.93577 58.48497  79.77717 46.29834 63.54791 58.80883 33.84140
## Student_6  47.54904 61.49240  98.22231 37.86975 59.21047 60.86879 60.35256
## Student_7  40.34567 59.23823  81.36560 51.37994 61.08184 58.68226 41.01992
## Student_8  50.71626 66.84756  98.19066 47.01692 64.08681 65.37164 51.17374
## Student_9  38.39792 55.73816  85.43779 44.30083 68.63588 58.50212 47.03987
## Student_10 42.12110 66.97661  66.21344 44.07510 59.48115 55.77348 24.85550
## Student_11 45.71909 64.27109  80.52938 49.48903 57.21439 59.44460 34.81029
## Student_12 45.77020 79.05900  67.61402 34.35644 67.14151 58.78823 44.70256
## Student_13 43.38693 63.89656  86.78865 33.37581 55.53521 56.59663 53.41284
## Student_14 50.91872 64.11045 103.97466 61.48522 54.21214 66.94024 53.05593
## Student_15 45.98704 59.47989  62.28075 24.37904 57.34852 49.89504 37.90171
## Student_16 44.76547 63.22565  94.80464 45.12750 72.22841 64.03033 50.03918
## Student_17 41.88917 66.46326  71.01174 34.12472 55.83752 53.86528 36.88703
## Student_18 49.08685 68.33859 103.24872 47.70539 62.06760 66.08943 55.54333
## Student_19 37.68949 73.52322  98.81299 55.08452 54.10658 63.84336 61.12350
## Student_20 63.48237 72.76162  76.61148 24.25595 54.12983 58.24825 52.35553
```

Write a filter query to display the student with the highest average. Note that the student number (row name) should also be displayed.

```
max(res3[,"Mean"])
```

```
## [1] 67.13474
```

```
res3[res3[,"Mean"] == max(res3[,"Mean"]),,drop=FALSE]
```

```
##              CX101    CX102    CX103    CX104    CX105     Mean    Range
## Student_4 52.09428 71.18724 91.22867 70.81959 50.34394 67.13474 40.88473
```