# Stat Computing - Exercises 02 – Gradebook

Nick Gembs – ng334

There is a dataset in datasets/grade\_book.csv containing simulated grades for Statistical Computing. From the exercises directory, you can read in the dataset by running

dat <- read.csv("../grade\_book.csv")  
str(dat)

## 'data.frame': 50 obs. of 26 variables:  
## $ lab08 : int 8 8 6 6 8 5 10 8 8 8 ...  
## $ lab03 : int NA 5 6 4 7 5 7 8 8 8 ...  
## $ lab01 : int 10 9 7 9 5 9 8 9 9 7 ...  
## $ ex08 : int NA 19 19 18 17 16 19 19 20 20 ...  
## $ lab09 : int 9 9 8 9 7 7 8 4 9 8 ...  
## $ ex09 : int 18 20 18 20 NA 19 19 20 18 18 ...  
## $ ex04 : int 17 14 13 10 10 11 17 14 13 16 ...  
## $ lab06 : int 7 9 7 8 6 10 9 7 7 8 ...  
## $ project: int 82 83 NA 71 71 59 87 80 70 84 ...  
## $ ex07 : int 19 18 17 19 16 12 17 17 14 19 ...  
## $ ex03 : int 16 11 11 13 13 14 15 15 13 14 ...  
## $ lab07 : int 9 7 6 8 10 6 8 9 6 10 ...  
## $ ex12 : int 19 18 16 19 19 16 18 16 19 20 ...  
## $ netID : chr "gnz598" "hpl316" "ivp353" "iau101" ...  
## $ lab10 : int 9 8 6 7 4 5 10 9 6 9 ...  
## $ ex10 : int 15 16 NA 15 10 14 15 18 15 17 ...  
## $ lab12 : int 9 8 6 9 7 6 9 7 9 7 ...  
## $ ex02 : int 15 17 NA 15 15 16 15 19 17 19 ...  
## $ ex05 : int 17 19 NA NA NA 17 18 17 13 17 ...  
## $ lab05 : int 5 6 4 4 4 6 9 4 4 6 ...  
## $ ex11 : int 16 15 15 18 18 18 19 NA 17 18 ...  
## $ ex01 : int 12 15 11 16 13 12 14 16 11 11 ...  
## $ ex06 : int 19 18 13 18 18 15 15 19 19 20 ...  
## $ lab11 : int 9 9 8 8 10 7 10 10 8 8 ...  
## $ lab04 : int 8 7 7 9 8 7 8 8 9 10 ...  
## $ lab02 : int 8 9 6 10 10 10 9 10 10 10 ...

Complete the following exercises related to the grade book.

1. Randomly generate with replacement a birth date for each student between 2001-01-01 and 2005-12-31. Print out how many unique birth dates there are (year-month-date) and how many unique birthdays there are (month-date). Is this surprising?

dim(dat)

## [1] 50 26

bday = (as.Date(runif(50, min = 0, max = 1825), origin = as.Date("2001-01-01")))  
bdayshort = format(bday, format = "%m/%d")  
length(unique(bday))

## [1] 50

b = length(unique(bdayshort))  
b

## [1] 46

There are 50 unique (year-month-date) birthdays and 46 unique (month-date) birthdays. This makes sense because there are 1825 unique birthday options with year while only 365 without year.

1. Add the birthdate column to the data frame in 3 different ways: using cbind, $, and [[]].

way1 = cbind(dat,birthdate = bday)  
  
#way2  
dat$birthdate = bday  
  
#way3  
dat[["birthdate"]] = bday

1. Print out the column names.

names(dat)

## [1] "lab08" "lab03" "lab01" "ex08" "lab09" "ex09"   
## [7] "ex04" "lab06" "project" "ex07" "ex03" "lab07"   
## [13] "ex12" "netID" "lab10" "ex10" "lab12" "ex02"   
## [19] "ex05" "lab05" "ex11" "ex01" "ex06" "lab11"   
## [25] "lab04" "lab02" "birthdate"

1. Remove the birthdate column and show that it’s gone.

dat$birthdate = NULL  
names(dat)

## [1] "lab08" "lab03" "lab01" "ex08" "lab09" "ex09" "ex04"   
## [8] "lab06" "project" "ex07" "ex03" "lab07" "ex12" "netID"   
## [15] "lab10" "ex10" "lab12" "ex02" "ex05" "lab05" "ex11"   
## [22] "ex01" "ex06" "lab11" "lab04" "lab02"

1. Print out which column number has the netIDs.

which( names(dat)=="netID" )

## [1] 14

1. Print out the column numbers that contain lab grades. You might find the grep function useful.

grep('lab', names(dat))

## [1] 1 2 3 5 8 12 15 17 20 24 25 26

1. Print out the column numbers that contain exercise grades.

grep('ex', names(dat))

## [1] 4 6 7 10 11 13 16 18 19 21 22 23

1. What happens when you try to convert the data frame to a matrix with as.matrix?

matdat <- as.matrix(dat)

It automatically converts all of the data types to string because a matrix has to have uniform data type.

1. Extract the exercise columns and convert to a matrix. Why does this work as intended?

datmat = as.matrix(dat[grep('ex', names(dat))])

This works as intended because the rule of uniform data type of a matrix is not broken, so R does not auto convert.

1. Add a column to the data frame containing each student’s average exercise grade. Treat missing values as a grade of 0. You can do this in a couple of lines with rowSums or rowMeans. Exercises are out of 20. Print out the average exercise grades for the first 10 students.

datmat <- replace(datmat, is.na(datmat), 0)  
dat$exAVG = 5\*rowMeans(datmat, na.rm =TRUE)  
5\*rowMeans(datmat)[1:10]

## [1] 76.25000 83.33333 55.41667 75.41667 62.08333 75.00000 83.75000 79.16667  
## [9] 78.75000 87.08333

1. Calculate each student’s exercise average again, this time using the average of the non-missing values. Print out the average exercise grades for the first 10 students.

datmat = as.matrix(dat[grep('ex', names(dat))])  
5\*rowMeans(datmat, na.rm =TRUE)[1:10]

## [1] 108.02083 108.97436 94.20833 106.84028 95.94697 98.07692 109.51923  
## [8] 112.15278 102.98077 113.87821

1. Print out the number of missing exercises for each exercise.

sum(is.na(datmat[,1]))

## [1] 1

sum(is.na(datmat[,2]))

## [1] 3

sum(is.na(datmat[,3]))

## [1] 1

sum(is.na(datmat[,4]))

## [1] 2

sum(is.na(datmat[,5]))

## [1] 1

sum(is.na(datmat[,6]))

## [1] 2

sum(is.na(datmat[,7]))

## [1] 1

sum(is.na(datmat[,8]))

## [1] 1

sum(is.na(datmat[,9]))

## [1] 5

sum(is.na(datmat[,10]))

## [1] 4

1. Calculate each student’s lab average, and add to the data frame. Labs are out of 10. Print out the average lab grades for the first 10 students.

datmat = as.matrix(dat[grep('lab', names(dat))])  
  
10\*rowMeans(datmat, na.rm =TRUE)[1:10]

## [1] 82.72727 78.33333 64.16667 75.83333 71.66667 69.16667 87.50000 77.50000  
## [9] 77.50000 82.50000

#datmat <- replace(datmat, is.na(datmat), 0)  
dat$labAVG = 10\*rowMeans(datmat, na.rm =TRUE)

1. Using the formula in the syllabus, add a column containing each student’s overall numeric grade. Treat missing assignments as 0. Project is out of 100.

dat <- replace(dat, is.na(dat), 0)  
  
dat$numericgrade = dat$labAVG\*.2 + dat$exAVG\*.6 + dat$project\*.2

1. Using the guidelines in the syllabus, add a column containing each student’s letter grade.

lam = c()  
for(i in 1:50){  
 if(dat[i,"numericgrade"] >= 93){  
 lam[i] = "A"  
 } else if (dat[i,"numericgrade"] >= 90){  
 lam[i] = "A-"  
 } else if (dat[i,"numericgrade"] >= 87){  
 lam[i] = "B+"  
 } else if (dat[i,"numericgrade"] >= 83){  
 lam[i] = "B"  
 } else if (dat[i,"numericgrade"] >= 80){  
 lam[i] = "B-"  
 } else if (dat[i,"numericgrade"] >= 77){  
 lam[i] = "C+"  
 } else if (dat[i,"numericgrade"] >= 73){  
 lam[i] = "C"  
 } else if (dat[i,"numericgrade"] >= 70){  
 lam[i] = "C-"  
 } else if (dat[i,"numericgrade"] >= 67){  
 lam[i] = "D+"  
 } else if (dat[i,"numericgrade"] >= 63){  
 lam[i] = "D"  
 } else if (dat[i,"numericgrade"] >= 60){  
 lam[i] = "D-"  
 } else {  
 lam[i] = "F"  
 }  
}  
  
dat$lettergrade = lam  
  
head(dat)

## lab08 lab03 lab01 ex08 lab09 ex09 ex04 lab06 project ex07 ex03 lab07 ex12  
## 1 8 0 10 0 9 18 17 7 82 19 16 9 19  
## 2 8 5 9 19 9 20 14 9 83 18 11 7 18  
## 3 6 6 7 19 8 18 13 7 0 17 11 6 16  
## 4 6 4 9 18 9 20 10 8 71 19 13 8 19  
## 5 8 7 5 17 7 0 10 6 71 16 13 10 19  
## 6 5 5 9 16 7 19 11 10 59 12 14 6 16  
## netID lab10 ex10 lab12 ex02 ex05 lab05 ex11 ex01 ex06 lab11 lab04 lab02  
## 1 gnz598 9 15 9 15 17 5 16 12 19 9 8 8  
## 2 hpl316 8 16 8 17 19 6 15 15 18 9 7 9  
## 3 ivp353 6 0 6 0 0 4 15 11 13 8 7 6  
## 4 iau101 7 15 9 15 0 4 18 16 18 8 9 10  
## 5 nue991 4 10 7 15 0 4 18 13 18 10 8 10  
## 6 yky774 5 14 6 16 17 6 18 12 15 7 7 10  
## exAVG labAVG numericgrade lettergrade  
## 1 76.25000 82.72727 78.69545 C+  
## 2 83.33333 78.33333 82.26667 B-  
## 3 55.41667 64.16667 46.08333 F  
## 4 75.41667 75.83333 74.61667 C  
## 5 62.08333 71.66667 65.78333 D  
## 6 75.00000 69.16667 70.63333 C-

1. Print out the netID, numeric average, and letter grade for the top 10 scorers. You may want to look at the order function.

tops = dat[order(dat$numericgrade, decreasing=TRUE),c("netID","numericgrade","lettergrade")]  
tops[1:10,]

## netID numericgrade lettergrade  
## 22 esy224 92.26667 A-  
## 24 waq733 89.78333 B+  
## 36 hbz284 89.76667 B+  
## 48 ayp949 89.35000 B+  
## 42 bor334 86.86667 B  
## 21 ujq876 86.61667 B  
## 34 rtq675 86.10000 B  
## 43 sxz212 85.83333 B  
## 10 kap440 85.55000 B  
## 7 pmc842 85.15000 B