Advanced Data Analysis 2, Stat 428/528 Homework 18 Name: Answer Key
Spring 2015 Prof. Erik B. Erhardt

Part I. (90 points) Do all calculations in R. All R code for the assignment should be included with the part of the problem it addresses (for code and output use a fixed-width font, such as Courier). Code is used to calculate result. Text is used to report and interpret results. Do not report or interpret results in the code.

(20pts) **1. Type conversions**
Type the following code in your R terminal.

```
v <- factor(c("2", "3", "5", "7", "11"))
```

(a) (5 pts) Describe the variable type/class of `v`.

Solution: The variable `v` is a factor variable with levels 1, 2, 3, 4, 5, mapped alphabetically to labels "11", "2", "3", "5", and "7", respectively.

```
v

## [1] 2  3  5  7  11
## Levels: 11 2 3 5 7

class(v)

## [1] "factor"

str(v)

##  Factor w/ 5 levels "11","2","3","5",..: 2 3 4 5 1
```

(b) (5 pts) Convert `v` to character with `as.character()`. Explain what just happened.

Solution: The factor values were converted to their labels and returned in a character string.

```
as.character(v)

## [1] "2"   "3"   "5"   "7"   "11"
```

(c) (5 pts) Convert `v` to numeric with `as.numeric()`. Explain what just happened.

Solution: The factor levels were returned in a numeric string.

```
as.numeric(v)

## [1] 2 3 4 5 1
```

(d) (5 pts) How would you convert the values of `v` to integers? Do it.

Solution: First return the character labels, then coerce to integers.

```
as.integer(as.character(v))

## [1]  2  3  5  7 11
```

(70pts) **2. From raw to technically correct data**
In this exercise we'll use `readLines()` to read in an irregular textfile: `http://statacumen.com/teach/ADA2/ADA2_HW_18_example.txt`. The file looks like this.

```
// Survey data. Created : 22 April 2015
// Field 1: Gender
// Field 2: Age (in years)
// Field 3: Weight (in kg)
M;28;81.3
male;45;
Female; 17 ;57,2
fem.;64;62.8
```

```
Ma.;16;55.3
m;;50,1
w;20.4;55
Fm;;
;55;55
```

First, we will read the data, work with the commented lines, then put the data lines into a matrix with column labels. Then, we will coerce the columns of the data to a structured data set.

Imagine that these three fields and four data rows are the first of potentially dozens of fields and 10,000 rows of data, so your strategy should be general and handle unseen but reasonable data values (for the columns in this dataset).

This is the type of coding that demands detailed comments for why you're writing each line of code and what your strategy is. Therefore, before each line of code, indicate the why and how of your code.

(a) (5 pts) Read the complete file using readLines.

Solution:

```
# Read data with readLines() so each line is a character string in a vector
fn.data <- "http://statacumen.com/teach/ADA2/ADA2_HW_18_example.txt"
example.txt <- readLines(fn.data)
example.txt

##  [1] "// Survey data. Created : 22 April 2014"
##  [2] "// Field 1: Gender"
##  [3] "// Field 2: Age (in years)"
##  [4] "// Field 3: Weight (in kg)"
##  [5] "M;28;81.3"
##  [6] "male;45;"
##  [7] "Female; 17 ;57,2"
##  [8] "fem.;64;62.8"
##  [9] "Ma.;16;55.3"
## [10] "m;;50,1"
## [11] "w;20.4;55"
## [12] "Fm;;"
## [13] ";55;55"
```

(b) (10 pts) Separate the vector of lines into a vector containing comments and a vector containing the data. Hint: use `grepl()`.

Solution:

```
# detect lines starting (^) with a double slash (//)
ind.header <- grepl("^//", example.txt)
ind.header

##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE

# create an object with only header
example.head <- example.txt[ind.header]
example.head

## [1] "// Survey data. Created : 22 April 2014"
## [2] "// Field 1: Gender"
## [3] "// Field 2: Age (in years)"
## [4] "// Field 3: Weight (in kg)"

# create an object with only data
example.dat <- example.txt[!ind.header]
example.dat

## [1] "M;28;81.3"        "male;45;"         "Female; 17 ;57,2"
## [4] "fem.;64;62.8"     "Ma.;16;55.3"      "m;;50,1"
## [7] "w;20.4;55"        "Fm;;"             ";55;55"
```

(c) (5 pts) Extract the date from the first comment line and print to the console with the text "This data was created YYYY-MM-DD.", where the date is entirely numeric.

Solution:

```
# read date from first line of the comments
library(lubridate)

##
## Attaching package:  'lubridate'
##
## The following object is masked from 'package:plyr':
##
##     here

example.date <- dmy(example.head[1])
# print comment about when data were created
format(example.date, format = "This data was created %Y-%m-%d.")

## [1] "This data was created 2014-04-22."
```

(d) (15 pts) Read the data into a matrix as follows.

1. Split the character vectors in the vector containing data lines by semicolon (;) using `strsplit()`.
2. Find the maximum number of fields retrieved by `split()`. Append rows that are shorter with `NA`'s.
3. Use `unlist()` and `matrix()` to transform the data to row-column format.

Solution:

```
## (1)
# remove whitespace by substituting nothing where spaces appear
example.dat2 <- gsub(" ", "", example.dat)
# split strings by semicolon
example.fieldList <- strsplit(example.dat2, split = ";")
example.fieldList

## [[1]]
## [1] "M"      "28"     "81.3"
##
## [[2]]
## [1] "male" "45"
##
## [[3]]
## [1] "Female" "17"     "57,2"
##
## [[4]]
## [1] "fem." "64"   "62.8"
##
## [[5]]
## [1] "Ma."  "16"   "55.3"
##
## [[6]]
## [1] "m"    ""     "50,1"
##
## [[7]]
## [1] "w"    "20.4" "55"
##
## [[8]]
## [1] "Fm" ""
##
## [[9]]
## [1] ""   "55" "55"

## (2)
# append short rows with NA
  # for each field list calculate the length
  # unlist these lengths into a single vector
  # determine the max length
  # assign the max length to num.fields
```

```r
num.fields <- max(unlist(lapply(example.fieldList, length)))

# loop through each list element
for (i.dat in 1:length(example.dat)) {
  # if shorter than num.fields
  if (length(example.fieldList[[i.dat]]) < num.fields) {
    # then append the necessary number of NAs to the vector
    example.fieldList[[i.dat]] <-
      c(example.fieldList[[i.dat]]
        , rep(NA, num.fields - length(example.fieldList[[i.dat]]))
        )
  }
}
# There are now \lst{NA}s completing the length of each data row.
#example.fieldList

## (3)
# Convert the data to a matrix
example.mat <- matrix(unlist(example.fieldList)
                    , nrow = length(example.fieldList)
                    , byrow = TRUE)
example.mat
```

```
##       [,1]     [,2]   [,3]
##  [1,] "M"      "28"   "81.3"
##  [2,] "male"   "45"   NA
##  [3,] "Female" "17"   "57,2"
##  [4,] "fem."   "64"   "62.8"
##  [5,] "Ma."    "16"   "55.3"
##  [6,] "m"      ""     "50,1"
##  [7,] "w"      "20.4" "55"
##  [8,] "Fm"     ""     NA
##  [9,] ""       "55"   "55"
```

(e) (5 pts) From comment lines 2-4, extract the names of the fields. Set these as `colnames` for the `matrix` you just created.
Solution:

```r
# remove the first row with the date, the rest have the field names
example.head.names <- example.head[2:length(example.head)]
# unlist into a matrix, then only keep the second column with the field names
example.colnames <- matrix(unlist(strsplit(example.head.names, split = ":"))
                                  , nrow = length(example.head.names)
                                  , byrow = TRUE)[,2]
example.colnames
```

```
## [1] " Gender"         " Age (in years)" " Weight (in kg)"
```

```r
# trim the white space from around the names
library(stringr)
example.colnames <- str_trim(example.colnames)
# replace spaces with underscores
example.colnames <- gsub(" ", "_", example.colnames)
example.colnames
```

```
## [1] "Gender"         "Age_(in_years)" "Weight_(in_kg)"
```

```r
# assign column names to the data matrix
colnames(example.mat) <- example.colnames
example.mat
```

```
##       Gender   Age_(in_years) Weight_(in_kg)
##  [1,] "M"      "28"           "81.3"
##  [2,] "male"   "45"           NA
##  [3,] "Female" "17"           "57,2"
##  [4,] "fem."   "64"           "62.8"
##  [5,] "Ma."    "16"           "55.3"
##  [6,] "m"      ""             "50,1"
##  [7,] "w"      "20.4"         "55"
##  [8,] "Fm"     ""             NA
##  [9,] ""       "55"           "55"
```

(f) (5 pts) Coerce the `matrix` to a `data.frame`, making sure all columns are `character` columns.
Solution:

```
# coerce the matrix to a data.frame, keep values as character (by not converting to factors)
example.df <- as.data.frame(example.mat, stringsAsFactors = FALSE)
example.df
```

```
##   Gender Age_(in_years) Weight_(in_kg)
## 1      M             28           81.3
## 2   male             45           <NA>
## 3 Female             17           57,2
## 4   fem.             64           62.8
## 5    Ma.             16           55.3
## 6      m                          50,1
## 7      w           20.4             55
## 8     Fm                          <NA>
## 9                    55             55
```

```
str(example.df)
```

```
## 'data.frame': 9 obs. of  3 variables:
##  $ Gender        : chr  "M" "male" "Female" "fem." ...
##  $ Age_(in_years): chr  "28" "45" "17" "64" ...
##  $ Weight_(in_kg): chr  "81.3" NA "57,2" "62.8" ...
```

(g) (10 pts) Use a string distance technique to transform the `Gender` column into a `factor` variable with labels `man` and `woman`.
Solution:

```
# matching names
gender.match <- c("male", "female")
# labels for Gender to assign later
gender.labels <- c("man", "woman")

# first, make some obvious changes based on first characters
# use substr to pull out the first character, and make capital
gender.first.char <- toupper(substr(example.df[,1], 1, 1))

library(stringr)
# if the Gender string starts with "m" or "M", change to "Male"
gender.first.char <- str_replace(gender.first.char, "M", "Male")
# if the Gender string starts with "f" or "F" or "w" or "W", change to "Female"
gender.first.char <- str_replace(gender.first.char, "F", "Female")
gender.first.char <- str_replace(gender.first.char, "W", "Female")
# if the Gender is blank, change to NA
gender.first.char[(str_length(gender.first.char) == 0)] <- NA
example.df[,1] <- gender.first.char

# use the stringdist function to find most closely matching label
library(stringdist)
gender.dist <- stringdistmatrix(example.df[,1], gender.match, method = "osa")
# reassign Gender column with gender.labels most closely matching
  # max.col() chooses the column with the maximum distance
  #   so I take the negative of the gender.dist so we find the minimum distance
  # then assign that label to the data.frame
example.df[,1] <- gender.labels[max.col(-gender.dist)]
# set as factor variable
example.df[,1] <- factor(example.df[,1])
as.numeric(factor(example.df[,1]))
```

```
## [1]  1  1  2  2  1  1  2  2 NA
```

```
example.df
```

```
##   Gender Age_(in_years) Weight_(in_kg)
## 1    man             28           81.3
## 2    man             45           <NA>
## 3  woman             17           57,2
## 4  woman             64           62.8
## 5    man             16           55.3
## 6    man                          50,1
## 7  woman           20.4             55
## 8  woman                          <NA>
## 9   <NA>             55             55
```

(h) (5 pts) Coerce the `Age` column to `integer`.
Solution:

```
# determine which column number is Age
colnum.Age <- which.min(stringdist(substr(example.colnames, 1, 3), "Age", method = "lcs"))
# replace commas with periods
example.df[, colnum.Age] <- gsub(",", ".", example.df[, colnum.Age], fixed = TRUE)
# set type to integer
example.df[, colnum.Age] <- as.integer(example.df[, colnum.Age])
```

(i) (5 pts) Coerce the `weight` column to `numeric`. Hint: use `gsub()` to replace comma's with a period.
Solution:

```
# determine which column number is Weight
colnum.Weight <- which.min(stringdist(substr(example.colnames, 1, 6), "Weight", method = "lcs"))
# replace commas with periods
example.df[, colnum.Weight] <- gsub(",", ".", example.df[, colnum.Weight], fixed = TRUE)
# set type to numeric
example.df[, colnum.Weight] <- as.numeric(example.df[, colnum.Weight])
```

(j) (5 pts) Show off your beautiful dataset!
Solution: Looks great! Ready for analysis.

```
str(example.df)
```

```
## 'data.frame': 9 obs. of  3 variables:
##  $ Gender        : Factor w/ 2 levels "man","woman": 1 1 2 2 1 1 2 2 NA
##  $ Age_(in_years): int  28 45 17 64 16 NA 20 NA 55
##  $ Weight_(in_kg): num  81.3 NA 57.2 62.8 55.3 50.1 55 NA 55
```

```
example.df
```

```
##   Gender Age_(in_years) Weight_(in_kg)
## 1    man             28           81.3
## 2    man             45             NA
## 3  woman             17           57.2
## 4  woman             64           62.8
## 5    man             16           55.3
## 6    man             NA           50.1
## 7  woman             20           55.0
## 8  woman             NA             NA
## 9   <NA>             55           55.0
```