sb_rmd_ex_07

Sander Bossier

2024-01-17

1.

Consider the character vector x:

```
x <- c(
  "Swan swam over the pond, Swim swan swim!",
  "Swam swam back again - Well swum swan!"
)
```

Split the strings in `x` into words, removing any *commas* or *hyphens*. Call the result `y`.

```
y <- strsplit(
  x,
  split = ",? -? ?"
)
y
```

```
## [[1]]
## [1] "Swan"  "swam"  "over"  "the"   "pond"  "Swim"  "swan"  "swim!"
##
## [[2]]
## [1] "Swam"  "swam"  "back"  "again" "Well"  "swum"  "swan!"
```

2.

The .rda file `hafu` in the RStudio Project `Text` contains a data set `hafu` on manga characters. In the `Father` and `Mother` columns, some values have question marks after the country name, indicating that the author was uncertain about the nationality of the parent:

```
load("hafu.rda")
head(hafu[, -ncol(hafu)], n = 10)
```

```
##    Year                   Series                Character Gender        Father
## 1  1963            Yuki no Taiyou                   Sanae      F      Japanese
## 2  1964                Cyborg 009           Joe Shimamura      M      American
## 3  1967                 Lupin III               Lupin III      M       French?
## 4  1967              Nekome Kozou            Cat-Eyed Boy      M      Japanese
## 5  1972                 Gatchaman            Jun the Swan      F          <NA>
## 6  1974            Great Mazinger               Jun Hono      F      American
## 7  1975         Ichigo Monogatari                  Ichigo      F Scandinavian?
## 8  1976 Kochira Katsushika-ku... Reiko Katherine Akimoto      F     Japanese?
## 9  1977      Haikara-san ga Tooru         Shinobu Iijyuin      M      Japanese
## 10 1979        Mobile Suit Gundam               Amuro Ray      M      Japanese
##       Mother Eyes  Hair
## 1   American <NA>  <NA>
## 2   Japanese brown brown
## 3  Japanese? black black
## 4    Fantasy brown  <NA>
## 5       <NA> green green
## 6   Japanese black black
## 7  Japanese? brown brown
## 8    French?  blue blond
## 9     German  blue blond
## 10  Canadian  blue brown
```

Create two vectors `FatherUncertain` and `MotherUncertain` denoting whether or not there was a question mark in the `Father` or `Mother` column, respectively.

```r
FatherUncertainSB <- endsWith(
  x = as.character(hafu$Father),
  suffix = "?"
)
MotherUncertainSB <- endsWith(
  x = as.character(hafu$Mother),
  suffix = "?"
)

FatherUncertain <- with(hafu, grepl("\\?", Father))
MotherUncertain <- with(hafu, grepl("\\?", Mother))

identical(FatherUncertainSB, FatherUncertain)
```

```
## [1] FALSE
```

```r
identical(MotherUncertainSB, MotherUncertain)
```

```
## [1] FALSE
```

3.

Create a new data frame `hafu2` by removing the question marks from the `Father` and `Mother` columns in `hafu` (i.e. replace them by an empty string).

```r
hafu2 <- within(hafu, {
  Father <- gsub(pattern = "\\?", replacement = "", x = Father)
  Mother <- gsub(pattern = "\\?", replacement = "", x = Mother)
  }
)
```

Challenges

4.

The .rda file `Dataline` contains a vector `Dataline` that contains a single value. This value is the result of a webscraper. It contains 1500 fields, separated by a semicolon:

```r
load("Dataline.rda")
```

Create a vector `Fields` which contains the 1500 fields as separate values.

```r
Fields <- strsplit(Dataline, split = ";")[[1]]
```

5.

On the basis of the vector `Fields` create: * A character vector `Label` containing the labels of every field: possible values are "Low", "Middle" and "High". * A numeric vector `ID` containing the ID numbers of every field (i.e. the value following the underscore). * A character vector `newLabel` containing labels constructed as `ID_xxx_Label`, with `xxx` the value of `ID` for that field and `Label` the value of `Label` for that field.

```
head(Fields, n = 10)
```

```
##  [1] "MiddleID_54"  "HighID_115"   "LowID_109"    "HighID_346"   "MiddleID_298"
##  [6] "HighID_139"   "MiddleID_221" "MiddleID_447" "HighID_27"    "LowID_331"
```

```
Label <- gsub(
  pattern = "ID_\\d+", #KP solution is more general by no requiring the underscore followed by any numb
  replacement = "",
  x = Fields
)

ID <- as.numeric(
  gsub(
    pattern = "\\w+ID_", # again KP solution is more general as it matches a sequence of one or more no
    replacement = "",
    x = Fields
  )
)

newLabel <- paste(
  "ID",
  ID,
  Label,
  sep = "_"
)
```

```
# Solution 5:
LabelKP <- gsub("ID.*", "", Fields)
IDKP <- as.numeric(gsub("[^_]+_", "", Fields))
newLabelKP <- paste("ID", ID, Label, sep = "_")
```

```
identical(Label, LabelKP)
```

```
## [1] TRUE
```

```
identical(ID, IDKP)
```

```
## [1] TRUE
```

```
identical(newLabel, newLabelKP)
```

```
## [1] TRUE
```

6.

The file genes.rda contains a named vector **genes**. This vector contains the function descriptions of a set of genes. The names of this vector give the COG codes of these genes:

```
load("genes.rda")
head(genes, n = 10)
```

```
##                                                                  RCOG3478
## "Predicted nucleic-acid-binding protein containing a Zn-ribbon domain "
##                                                                 CCOG1031
##                                   "Uncharacterized Fe-S oxidoreductase "
##                                                                 SCOG3771
##                                            "Predicted membrane protein "
##                                                                 KCOG1191
##                 "DNA-directed RNA polymerase specialized sigma subunit "
##                                                                 SCOG5230
##                                     "Uncharacterized conserved protein "
##                                                                 JCOG0532
##                        "Translation initiation factor 2 (IF-2; GTPase) "
##                                                                 CCOG1905
##                          "NADH:ubiquinone oxidoreductase 24 kD subunit "
##                                                                 FCOG0461
##                                     "Orotate phosphoribosyltransferase "
##                                                                 SCOG2904
##                         "Uncharacterized protein conserved in bacteria "
##                                                                 ECOG0509
##                   "Glycine cleavage system H protein (lipoate-binding) "
```

Use this vector to create a character vector `geneclass` with the values "Predicted", "Uncharacterized" and "Described". These are defined as follows: * If the gene description contains "Uncharacterized", it is always called "Uncharacterized" * If the gene description contains "Predicted" but not "Uncharacterized", it is called "Predicted" * All other genes are called "Described"

```r
geneclass <- rep("Described", length(genes))
IDunc <- grep(pattern = "uncharacterized", x = genes, ignore.case = TRUE)
IDpred <- grep(pattern = "predicted", x = genes, ignore.case = TRUE)
geneclass[IDpred] <- "Predicted"
geneclass[IDunc] <- "Uncharacterized"
```

7.

Use the `genes` vector to find all COG codes for functions containing an enzyme name. An enzyme can be recognised by their name ending on "ase" or "ases". For this, you need to: * Extract the COG code from the names: this consists of the numbers following the letters COG in the name * Find at which positions in the vector genes an enzyme is described * Use the information from the previous two steps to find only the COG codes of the enzymes

```r
COGcodes <- gsub(
  pattern = ".*COG", # Again KP solutions is likely more general as it matches any non-digit characters
  replacement = "",
  x = names(genes)
)
postions <- grepl(
  pattern = ".*ases?\\b", # The questionmark only affects the 's' and thus the pattern can match 'ase'
  x = genes
)
COGcodes_recognized <- COGcodes[postions]
```

```r
COG <- gsub("\\D", "", names(genes))
isEnzyme <- grep("ases?\\b", genes)
enzymCOG <- COG[isEnzyme]
```

```r
identical(COGcodes_recognized, enzymCOG)
```

```
## [1] TRUE
```

8.

The RStudio Project `Text` also contains two external files appletweets.txt and microsofttweets.txt. Each contains a selection of tweets scraped from Twitter/X and addressed to the helpdesk of either Apple or Microsoft. Use the function `readLines()` to read in both files. Store the tweets for Apple in a vector `AppleTweets` and those for Microsoft in a vector `MSTweets`.

```r
AppleTweets <- readLines(
  con = "appletweets.txt",
  encoding = "UTF-8"
)
MSTweets <- readLines(
  con = "microsofttweets.txt",
  encoding = "UTF-8"
)
```

9.

Make sure that there are no empty character values (“ “) in either vector.

```r
AppleTweets <- readLines(
  con = "appletweets.txt",
  encoding = "UTF-8"
)
MSTweets <- readLines(
  con = "microsofttweets.txt",
  encoding = "UTF-8"
)
```

```r
AppleTweets <- AppleTweets[AppleTweets != ""]
MSTweets <- MSTweets[MSTweets != ""]
```

10.

Count the number of tweets that talk about iOS10. Think about possible spaces and capitalization.

```r
HASiOS10 <- sum(
  grepl(
    pattern = "\\bios\\s?10",
    x = AppleTweets,
    ignore.case = TRUE
  )
)
```

```
## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Mein iPad verliert st<e4>ndig seinen WLAN-Router
## seit iOS10-Update. Immer wieder neu einzurichten.  this sucks.' to a wide
## string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): input string 37 is invalid

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' para cobrarme y para no pagar 10.000M a la UE,
## est<e1>n mas listos...' to a wide string
## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): input string 107 is invalid

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Me gustar<ed>a que me dijese porque mi iPhone 5 no
## se enciende siquiera con cargador.' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): input string 117 is invalid

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' el nuevo IOS 10 se come la bater<ed>a m<e1>s
## r<e1>pido que antes o es mi imaginaci<f3>n?' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): input string 142 is invalid

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'RT #IOS10 va fatal. Jodi<f3> la bater<ed>a.  lo ha
## hecho fatal. Y no tengo cita en el Genius Bar hasta el 21S!  ' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): input string 152 is invalid

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' nes i ei diddymu, mae yna'n ail ymddangos yn y
## rhestr ieithoedd ail osod ag aw<e9>' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate '#IOS10 va fatal. Jodi<f3> la bater<ed>a.  lo ha
## hecho fatal. Y no tengo cita en el Genius Bar hasta el 21S!  ' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' Bonjour, gros probl<e8>me pour les c<e2>bles
## certifi<e9> Apple d'une autre marque sous IOS 10 c'est tr<e8>s <e9>nervant' to
## a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'RT Bonsoir , je suis <e9>norm<e9>ment d<e9><e7>ue
## d'iOS10, agissez au plus vite pour rectifier ce drame, merci.' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Bonsoir , je suis <e9>norm<e9>ment d<e9><e7>ue
## d'iOS10, agissez au plus vite pour rectifier ce drame, merci.' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'TEM DEZ MINUTOS QUE TO TENTANDO ABRIR O APP DE
## TELEFONE E N<c3>O ABRE QUE CU ' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' Iphone DESCART<c1>VEL 1ano e 4meses de uso e para
## de funcionar e unica op<e7><e3>o que e Apple d<e1> <e9> comprar novo com
## desconto INDGNADA' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' ce genre de bug n'est plus possible pour iOS 10
## d<e9>j<e0> qu'ily <e9>tait sur la b<ea>ta s'il vous pla<ee>t r<e9>gl<e9> le '
## to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Eita, , atualizei meu IOS e agora minhas notas
## n<e3>o funcionam, fecham repentinamente. O que eu fa<e7>o?' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' merci de m'avoir bloqu<e9> mon t<e9>l<e9>phone
## avec votre fdp de mise <e0> jour de IOS10 ramenez moi un nouvel iphone' to a
## wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' a tu<e9> mon iPhone. Merci iOS 10. ' to a wide
## string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' It's hooked up to a PC for <fe> sake of <fe> iOS10
## update via iTunes. It charges my iPhone just fine.' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' Iphone6 DESCART<c1>VEL parou de funcionar em 1ano
## e 4meses, e a unica solucao <e9> desconto para aquisi<e7><e3>o de um novo..
## lament<e1>vel' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Alguien que se haya descargado #ios10 y tenga
## problema con la App Notas? Se cierra sola en cuanto entras. Sres. de qu<e9>
## hago?' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Ei cria logo uma outra atualiza<e7><e3>o porque a
## bateria t<e1> acabando muito r<e1>pido com esse #iOS10' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' I need 1 call on genius bar in S<e3>o Paulo and is
## impossible schedule at site,this always crowded.It is a mobile exchange' to a
## wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Me sacaron los fondos de pantalla de las plumas, a
## m<ed> me gustaban #iOS10 ' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' <de> very same. When I unplug it, it does show
## more battery life, but charges slowly while saying "not charging".' to a wide
## string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Just me or is it unacceptable that iPhones costing
## <a3>700 have screens that smash so easily! ' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' Abi bana l<fc>tfen OS 10 g<fc>ncelletirmesini
## g<f6>ndermeyin.Laf olsun diye g<fc>ncelletirme g<f6>nderiyorsunuz.' to a wide
## string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Bonjour quand j'envoie les animations iMessage
## <e7>a fonctionne mais pas quand je les re<e7>ois ... une id<e9>e ? #ios10' to a
## wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' why can some people see the new features in I
## message but my fianc<e9> keeps getting a [sent with invisible link]' to a wide
## string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'RT Con el iOS10 se va a da<f1>ar el bot<f3>n del
## iPhone m<e1>s r<e1>pido de lo normal ' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'Con el iOS10 se va a da<f1>ar el bot<f3>n del
## iPhone m<e1>s r<e1>pido de lo normal ' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate 'So you going to help me get Beyonc<e9>'s profile
## picture or nah' to a wide string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' how can you make a product that is in effect glass
## and you can't replace the part ?? <a3><a3><a3><a3><a3><a3>&amp;' to a wide
## string

## Warning in grepl(pattern = "\\bios\\s?10", x = AppleTweets, ignore.case =
## TRUE): unable to translate ' beautifully designed products that when they break
## epithet cost the earth to fix sons I pad has a small crack <a3>220 to fix it'
## to a wide string
```

```
HASiOS10
```

```
## [1] 360
```

Extra exercises on dates and date-time objects

11.

Dates and Date-time objects in R are based on strings. Create the following objects (think about which type to use): * A vector called time1: 2011-01-30 * A vector called time2: 30/01/11 * A vector called time3: Jan 30, 2011 14:45 UTC

You can you these strings as your starting point:

```
text1 <- "2011-01-30"
text2 <- "30/01/11"
text3 <- "Jan 30, 2011 14:45 UTC"
```

```
time1 <- as.Date(text1)
time2 <- strptime(text2, format = "%d/%m/%y")
time3 <- strptime(text3, format = "%b %d, %Y %H:%M", tz = "UTC")
time1
```

```
## [1] "2011-01-30"
```

```
time2
```

```
## [1] "2011-01-30 CET"
```

```
time3
```

```
## [1] "2011-01-30 14:45:00 UTC"
```

12.

Create a vector called `dateSeq` containing the dates: * Starting from July, 1st 1989 * Ending at the last valid date before today * With a periodicity of 3 months

```r
dateSeq <- seq(
  from = as.Date("July, 1st 1989", format = "%B, %dst %Y"),
  to = Sys.Date()-1,
  by = "3 months"
)
```

13.

Find in `dateSeq`: * All dates that are later than January 2nd, 2000 * All dates in dateSeq that fall on a Monday

```r
dateSeq[dateSeq > as.Date("2000-01-02")]
```

```
##  [1] "2000-04-01" "2000-07-01" "2000-10-01" "2001-01-01" "2001-04-01"
##  [6] "2001-07-01" "2001-10-01" "2002-01-01" "2002-04-01" "2002-07-01"
## [11] "2002-10-01" "2003-01-01" "2003-04-01" "2003-07-01" "2003-10-01"
## [16] "2004-01-01" "2004-04-01" "2004-07-01" "2004-10-01" "2005-01-01"
## [21] "2005-04-01" "2005-07-01" "2005-10-01" "2006-01-01" "2006-04-01"
## [26] "2006-07-01" "2006-10-01" "2007-01-01" "2007-04-01" "2007-07-01"
## [31] "2007-10-01" "2008-01-01" "2008-04-01" "2008-07-01" "2008-10-01"
## [36] "2009-01-01" "2009-04-01" "2009-07-01" "2009-10-01" "2010-01-01"
## [41] "2010-04-01" "2010-07-01" "2010-10-01" "2011-01-01" "2011-04-01"
## [46] "2011-07-01" "2011-10-01" "2012-01-01" "2012-04-01" "2012-07-01"
## [51] "2012-10-01" "2013-01-01" "2013-04-01" "2013-07-01" "2013-10-01"
## [56] "2014-01-01" "2014-04-01" "2014-07-01" "2014-10-01" "2015-01-01"
## [61] "2015-04-01" "2015-07-01" "2015-10-01" "2016-01-01" "2016-04-01"
## [66] "2016-07-01" "2016-10-01" "2017-01-01" "2017-04-01" "2017-07-01"
## [71] "2017-10-01" "2018-01-01" "2018-04-01" "2018-07-01" "2018-10-01"
## [76] "2019-01-01" "2019-04-01" "2019-07-01" "2019-10-01" "2020-01-01"
## [81] "2020-04-01" "2020-07-01" "2020-10-01" "2021-01-01" "2021-04-01"
## [86] "2021-07-01" "2021-10-01" "2022-01-01" "2022-04-01" "2022-07-01"
## [91] "2022-10-01" "2023-01-01" "2023-04-01" "2023-07-01" "2023-10-01"
## [96] "2024-01-01"
```

```r
dateSeq[weekdays(dateSeq) == "Monday"]
```

```
##  [1] "1990-01-01" "1990-10-01" "1991-04-01" "1991-07-01" "1996-01-01"
##  [6] "1996-04-01" "1996-07-01" "2001-01-01" "2001-10-01" "2002-04-01"
## [11] "2002-07-01" "2007-01-01" "2007-10-01" "2012-10-01" "2013-04-01"
## [16] "2013-07-01" "2018-01-01" "2018-10-01" "2019-04-01" "2019-07-01"
## [21] "2024-01-01"
```

```r
identical(dateSeq[dateSeq > as.Date("2000-01-02")], dateSeq[which(dateSeq > as.Date("2000-01-02"))])
```

```
## [1] TRUE
```

```r
library(microbenchmark)
```

```
## Warning: package 'microbenchmark' was built under R version 4.3.2
```

```r
microbenchmark(
  dateSeq[dateSeq > as.Date("2000-01-02")],
  times = 10000,
  unit = "us"
)
```

```
## Unit: microseconds
##                                      expr  min     lq     mean median     uq
##  dateSeq[dateSeq > as.Date("2000-01-02")] 50.7 55.701 81.69688 58.201 67.701
##       max neval
##  9252.501 10000
```

```
microbenchmark(
  dateSeq[which(dateSeq > as.Date("2000-01-02"))],
  times = 10000,
  unit = "us"
)
```

```
## Unit: microseconds
##                                             expr    min      lq     mean median
##  dateSeq[which(dateSeq > as.Date("2000-01-02"))] 51.701 57.4015 83.83047 59.702
##      uq     max neval
##  66.501 14107.1 10000
```