Getting started with R

Vladan Devedzic

Introduction

Setup

Download R from https://cran.r-project.org/ (https://cran.r-project.org/) and install it.
Download RStudio from https://www.rstudio.com/products/rstudio/download/ (https://www.rstudio.com/products/rstudio/download/) and install it.

Open new project and new R script

Open new project in RStudio (File > New Project…).
Open new R script in the project (File > New File > R Script).

Working directory

Get working directory:

```
getwd()
```

Set working directory:
Session > Set Working Directory

Help

```
?setwd
help("setwd")
```

Install and load packages

```
# install.packages("ggplot2")
library(ggplot2)
# install.packages("caret")
library(caret)
```

Basic commands, operators and data types

Most of the sections in this document are structured as follows:

- pseudocode
- R code examples
- results or running the R code examples

In pseudocode throughout this document:

- syntax like `<something>` is used as a descriptive palceholder for an appropriate, context-specific identifier or another piece of code in lines of an actual R script
- syntax like `"<something>"` is used as a descriptive palceholder for an appropriate, context-specific *string* in lines of an actual R script

- leading '+' character in multi-line pseudocommands is used for indentation purposes only; it should be omitted in actual R code

Print a line of text in the console

```
print("Hi :)")
```

```
## [1] "Hi :)"
```

Naming and coding conventions

See https://google.github.io/styleguide/Rguide.xml (https://google.github.io/styleguide/Rguide.xml).

Assignment statement

```
x <- <something>
```

```
x <- 2
x
```

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

Manipulating objects in the workspace

```
ls()                      # list all objects in memory
rm(<o1>, <o2>, <o3>, ...) # remove one or more objects from memory by their names
rm(list = ls())           # remove all objects from memory (usually not recommended)
```

```
ls()
rm(x, y)
```

Operators

- + Add, 2 + 3 = 5
- - Subtract, 5 - 2 = 3
- * Multiply, 2 * 3 = 6
- / Divide, 6 / 2 = 3
- ^ Exponent, 2 ^ 3 = 8
- %% Modulus operator, 9%%2 = 1
- %/% Integer division, 9 %/% 2 = 4
- < Less than
- > Greater than
- = Equal to
- <= Less than or equal to
- >= Greater than or equal to
- != Not equal to
- ! Not
- | OR
- & And

Expressions

E.g., `x / y - z^2` etc.

```
3.4 / 2 + 7^2
```

```
## [1] 50.7
```

Vectors

```
<y> <- c(<something1>, <something2>, <something3>, ...)
<y> <- rep(<something>, <times>)
<y> <- <int1>:<int2>
<y> <- seq(<value1>, <value2>, by = <step>)
```

The index of the first element in a vector is 1, not 0.

```
y <- c(1, 2, 3)
z <- c(1.2, 3)
t <- 2:6
w <- seq(3.2, 4.7, by = 0.2)

w[3]
```

```
## [1] 3.6
```

```
w[]
```

```
## [1] 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6
```

```
w
```

```
## [1] 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6
```

Matrices

```
<m> <- matrix(c(3, 5, 7, 1, 9, 4), nrow = 3, ncol = 2, byrow = TRUE)
<m>.nrow <- nrow(<m>) # number of rows
<m>.ncol <- ncol(<m>) # number of columns
<m> <- t(<m>)         # transpose <m>
<m>[2,3]
<m>[2]
<m>[2, ]
```

```
a <- matrix(8:1, nrow = 2, ncol = 4, byrow = TRUE)
a
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    8    7    6    5
## [2,]    4    3    2    1
```

```
a.nrow <- nrow(a)
a.nrow
```

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

```
a <- t(a)
a
```

```
##      [,1] [,2]
## [1,]    8    4
## [2,]    7    3
## [3,]    6    2
## [4,]    5    1
```

```
a[1,2]
```

```
## [1] 4
```

```
a[2, ]
```

```
## [1] 7 3
```

```
a[2]
```

```
## [1] 7
```

```
a[]
```

```
##      [,1] [,2]
## [1,]    8    4
## [2,]    7    3
## [3,]    6    2
## [4,]    5    1
```

Lists

Ordered collections of elements of different types.

```
<list> <- list(<e1.name> = <e1>, <e2.name> = <e2>, <e3.name> = <e3>, ...)
<list>[[<index>]]     # accessing list element by index, showing value only
<list>[<index>]       # accessing list element by index, showing both name and value
<list>$<element.name> # accessing list element by its name
is.list(<something>)                                  # Is <something> a list?
<combined.list> <- c(<list1>, <list2>, <list3>, ...)  # list concatenation
names(<list>)                           # names of list elements
<list>[names(<list>) == <element.name>] # all elements of a list having the same name
unlist(<list>)                          # convert list into a named character vector
unlist(<list>, use.names = FALSE)       # convert list into a character vector
append(<list>,                          # insert new element into an existing list, after index <n>
+      list(<e1.name> = <e>),           # new element must be a list itself, that's why list(<e1.name> =  <e>)
+           <n>)                        # <n> is optional; if omitted, new element is appended at the end
<list>[[<n>]] <- NULL                   # remove <n>th element from <list>
```

```
traveler1 <- list(adult = TRUE, passport = "P212123", age = 34)
traveler1
```

```
## $adult
## [1] TRUE
##
## $passport
## [1] "P212123"
##
## $age
## [1] 34
```

```
traveler1[[3]]
```

```
## [1] 34
```

```
traveler2 <- list(adult = FALSE, passport = "P4567756", age = 14)
traveler2
```

```
## $adult
## [1] FALSE
##
## $passport
## [1] "P4567756"
##
## $age
## [1] 14
```

```
traveler2$age
```

```
## [1] 14
```

```
travelers <- c(traveler1, traveler2)
travelers
```

```
## $adult
## [1] TRUE
##
## $passport
## [1] "P212123"
##
## $age
## [1] 34
##
## $adult
## [1] FALSE
##
## $passport
## [1] "P4567756"
##
## $age
## [1] 14
```

```
travelers[[3]]
```

```
## [1] 34
```

```
travelers[[5]]
```

```
## [1] "P4567756"
```

```
travelers[5]
```

```
## $passport
## [1] "P4567756"
```

```
is.list(travelers)
```

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

```
is.vector(travelers)
```

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

```
names(travelers)
```

```
## [1] "adult"    "passport" "age"      "adult"    "passport" "age"
```

```
travelers[names(travelers) == "age"]
```

```
## $age
## [1] 34
##
## $age
## [1] 14
```

```
unlist(travelers)
```

```
##      adult   passport        age      adult   passport        age
##     "TRUE"  "P212123"       "34"    "FALSE" "P4567756"       "14"
```

```
unlist(travelers, use.names = FALSE)
```

```
## [1] "TRUE"     "P212123"  "34"       "FALSE"    "P4567756" "14"
```

```
age.of.travelers <- unlist(travelers[names(travelers) == "age"], use.names = FALSE)
age.of.travelers
```

```
## [1] 34 14
```

```
length(traveler1)
```

```
## [1] 3
```

```
traveler1 <- append(traveler1, list(country = "AUS"), 2)
length(traveler1)
```

```
## [1] 4
```

```
traveler1
```

```
## $adult
## [1] TRUE
##
## $passport
## [1] "P212123"
##
## $country
## [1] "AUS"
##
## $age
## [1] 34
```

```
traveler1[[3]] <- NULL
length(traveler1)
```

```
## [1] 3
```

```
traveler1
```

```
## $adult
## [1] TRUE
##
## $passport
## [1] "P212123"
##
## $age
## [1] 34
```

Data types

Vector, factor, numeric, character, logical, data.frame, matrix, list, …

```
class(<something>)                    # data type
mode(something), typeof(<something>)  # how a data item is internally stored in memory
```

```
class(a)
```

```
## [1] "matrix"
```

```
mode(a)
```

```
## [1] "numeric"
```

```
typeof(a)
```

```
## [1] "integer"
```

```
typeof(2.3)
```

```
## [1] "double"
```

Factors

```
<b> <- c(1, 2, 2, 2, 3, 1, 1, 4, 5, 4)
<f> <- as.factor(b)
levels(<f>)
<f> <- factor(c(1, 2, 3))
<f> <- gl(3,                                                    # gl() "generates levels" (here 3), i.e. factors
+         1, length = 10, labels = c("One", "Two", "Three"))    # each level replicated 1 time, length(<f>) = 10
```

```
b <- c(1, 1, 1, 2, 1, 1, 1, 1, 5, 4)
b.as.factor <- as.factor(b)
levels(b.as.factor)
```

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

```
f <- gl(3, 1, length = 10, labels = c("One", "Two", "Three"))
f
```

```
##  [1] One   Two   Three One   Two   Three One   Two   Three One
## Levels: One Two Three
```

```
f <- gl(3, 2, length = 10, labels = c("One", "Two", "Three"))
f
```

```
##  [1] One   One   Two   Two   Three Three One   One   Two   Two
## Levels: One Two Three
```

```
meal = factor(c("Lunch","Dinner"))
meal
```

```
## [1] Lunch  Dinner
## Levels: Dinner Lunch
```

```
meal = factor(c("Lunch","Dinner"), levels=c("Lunch","Dinner"))
meal
```

```
## [1] Lunch  Dinner
## Levels: Lunch Dinner
```

Dataframes

```
<dataframe> <- as.data.frame(<matrix>)
str(<dataframe>)
```

```
a.data.frame <- as.data.frame(a)
a.data.frame
```

```
##   V1 V2
## 1  8  4
## 2  7  3
## 3  6  2
## 4  5  1
```

```
str(a.data.frame)
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ V1: int  8 7 6 5
##  $ V2: int  4 3 2 1
```

Loops and branching

for, if, break, next

```
for (<i> in <int vector>) {
+  <line 1>
+  <line 2>
+  ...
+  if (<logical condition>) {
+    <line i1>
+    <line i2>
+    ...
+    break       # break: exit the loop; next: skip the remaining lines in this iteration
+  }
+  ...
+  <line n>
}
```

```
for (i in 1:10) {
  if (i == 3) {
    print("Done")
    break
  }
  s <- paste(i,"is current index", sep = " ")
  print(s)
}
```

```
## [1] "1 is current index"
## [1] "2 is current index"
## [1] "Done"
```

while, if-else, break, next

```
<i> <- <initial value>
while (logical condition involving <i>) {
+  <line 1>
+  <line 2>
+  ...
+  if (<logical condition>) {
+    <line i1>
+    <line i2>
+    ...
+    break       # break: exit the loop; next: skip the remaining lines in this iteration
+  } else {
+    <line j1>
+    <line j2>
+    ...
+  }
+  ...
+  <line n>
+  <i> <- <modify <i>>
}
```

```
i <- 1
while (i <= 10) {
  if (i == 5) {
    i <- i + 1
    next
  } else {
    print(paste(i, "is current index", sep = " "))
    i <- i + 1
  }
}
```

```
## [1] "1 is current index"
## [1] "2 is current index"
## [1] "3 is current index"
## [1] "4 is current index"
## [1] "6 is current index"
## [1] "7 is current index"
## [1] "8 is current index"
## [1] "9 is current index"
## [1] "10 is current index"
```

ifelse(, v1, v2)

Can return a vector (if involves another vector).

```
ifelse(1 < 6, TRUE, FALSE)
```

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

```
ifelse(1 < 6, "<", "Not <")
```

```
## [1] "<"
```

```
ifelse(1:10 < 6, 1, 2)
```

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

ggplot2

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

Data to plot (used in the examples below):

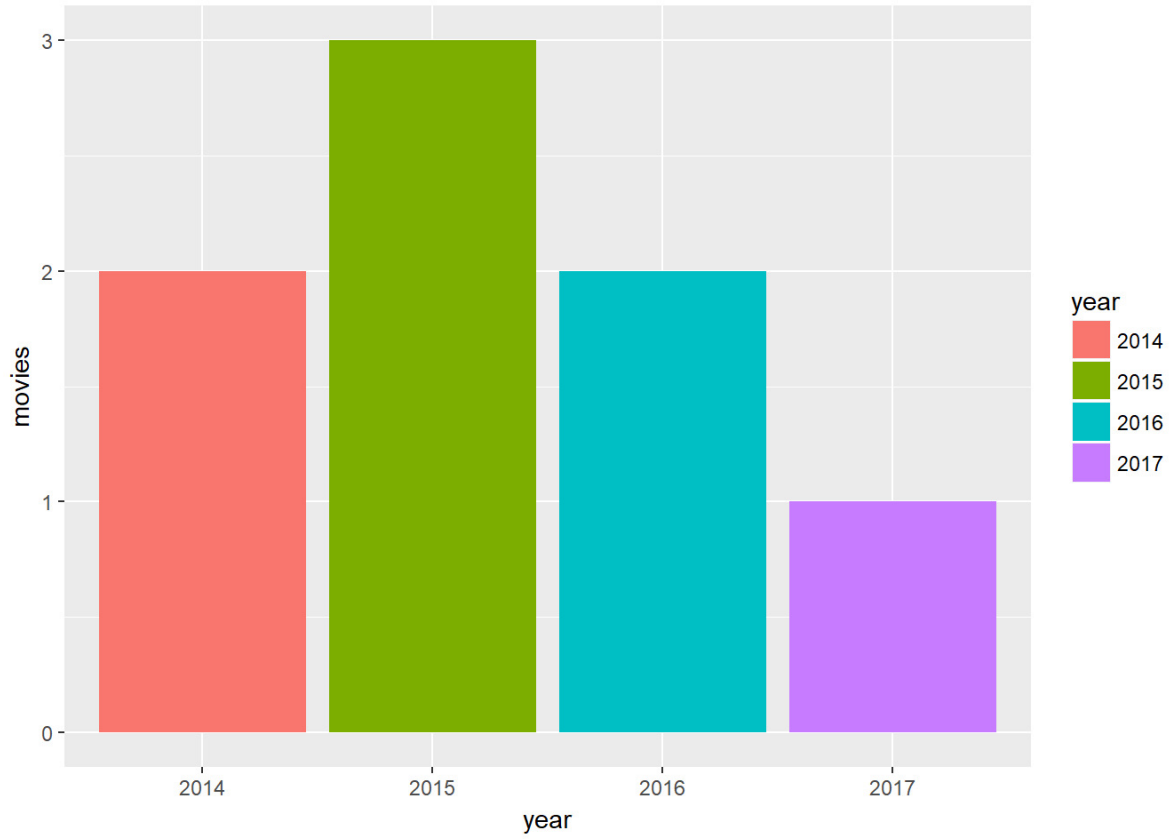
```
actor.xy <- data.frame(year = factor(c(2014, 2015, 2016, 2017)), movies = (c(2, 3, 2, 1)))
actor.xy
```

```
##   year movies
## 1 2014      2
## 2 2015      3
## 3 2016      2
## 4 2017      1
```

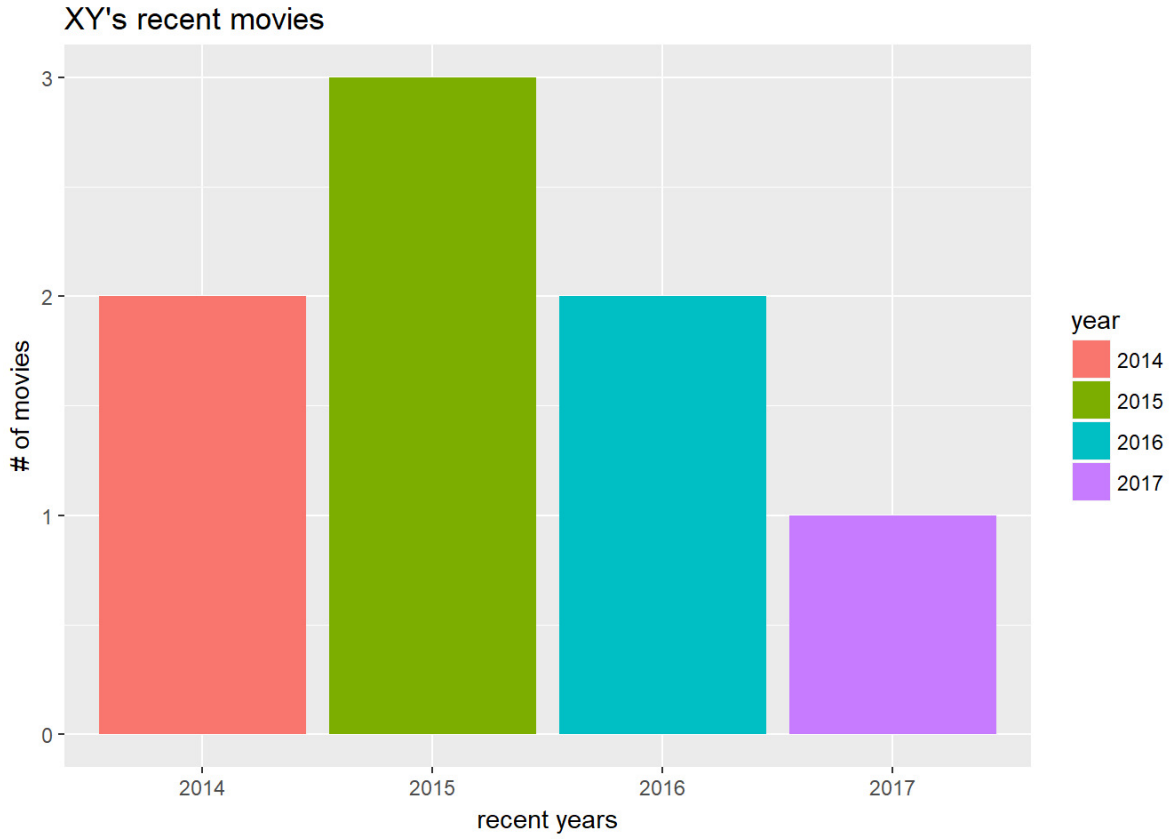
Bar graphs

```
ggplot(data = <dataframe>,
+      aes(x = <column 1>, y = <column 2>, fill = <column 1>)) +  # fill = <column 1> is optional; no y for counts
+ geom_bar(stat = "identity") +                                    # "identity" for values, "count" for counts
+ xlab("<x-axis label>") + ylab("<y-axis label>") +
+ ggtitle("<graph title>")
```

```
actor.plot.set <- ggplot(data = actor.xy, aes(x = year, y = movies, fill = year))
actor.plot.set + geom_bar(stat = "identity")
```



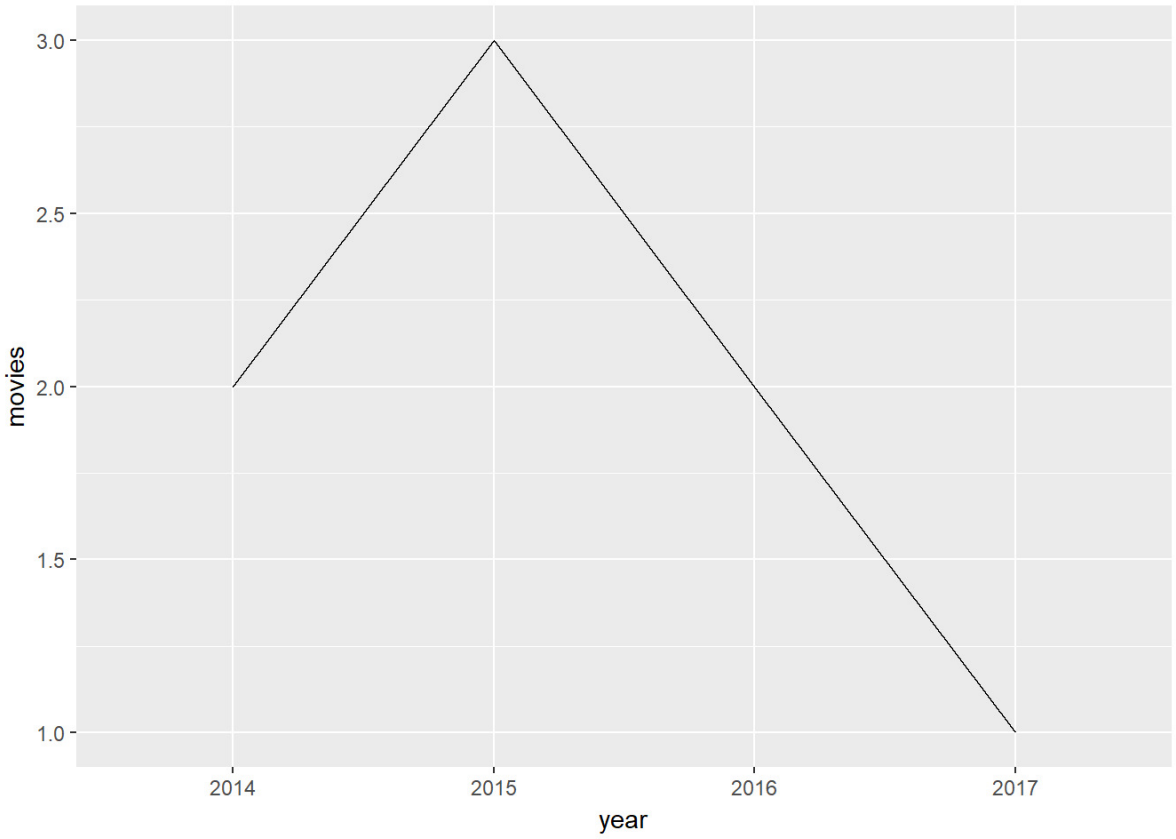

```
actor.plot.set +
  geom_bar(stat = "identity") +
  xlab("recent years") + ylab("# of movies") +
  ggtitle("XY's recent movies")
```



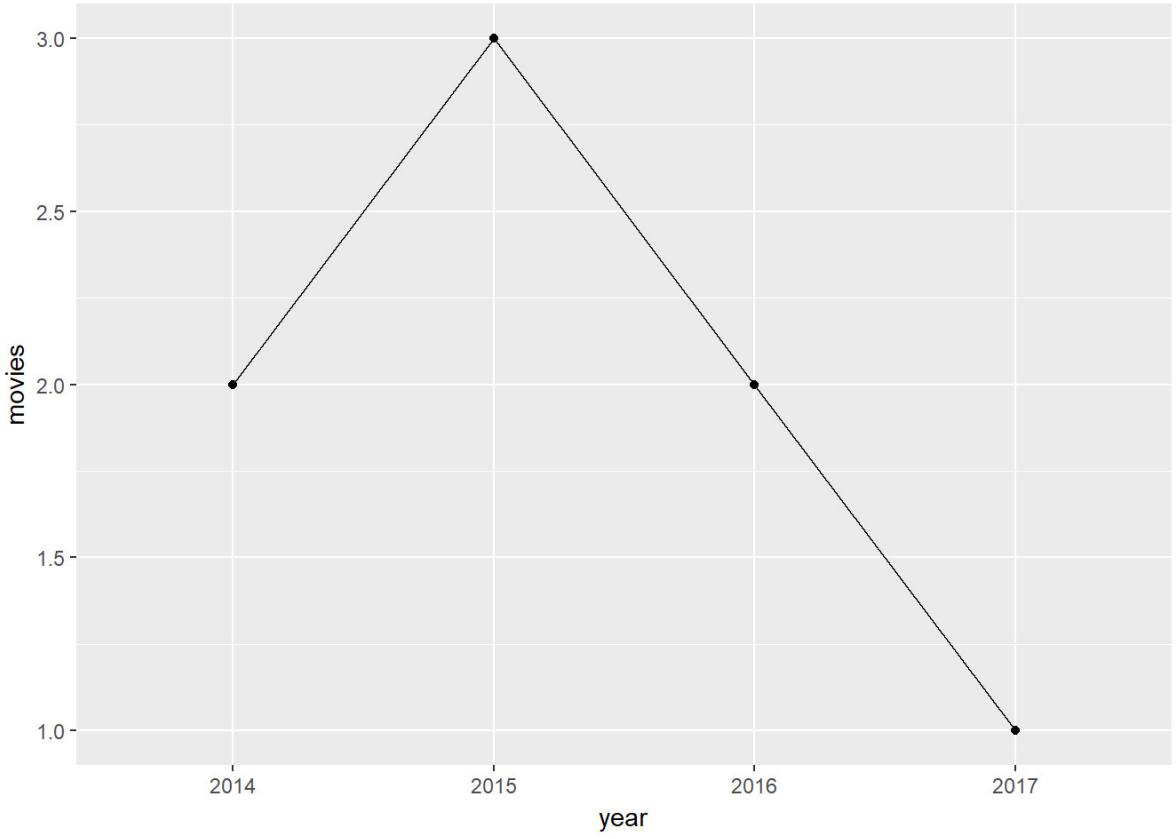
Line graphs

```
ggplot(data = <dataframe>,
+      aes(x = <column 1>, y = <column 2>, group = 1)) +  # group = 1: one line, all points connected
+ geom_line(colour = "<colour>", linetype = "<linetype>", size = <line thickness>) +
+ geom_point(colour="<colour>", size = <point size>, shape = <point shape>, fill = "<point fill colour>") +
+ xlab("<x-axis label>") + ylab("<y-axis label>") +
+ ggtitle("<graph title>")
All parameters in geom_line() and in geom_point() are optional.
The defaults are: colour = "black", linetype = "solid", size = 1, shape = 21 (circle), fill = "black"
See <http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/> for more information on colors.
See <http://www.cookbook-r.com/Graphs/Shapes_and_line_types/> for information on shapes and line types.
```

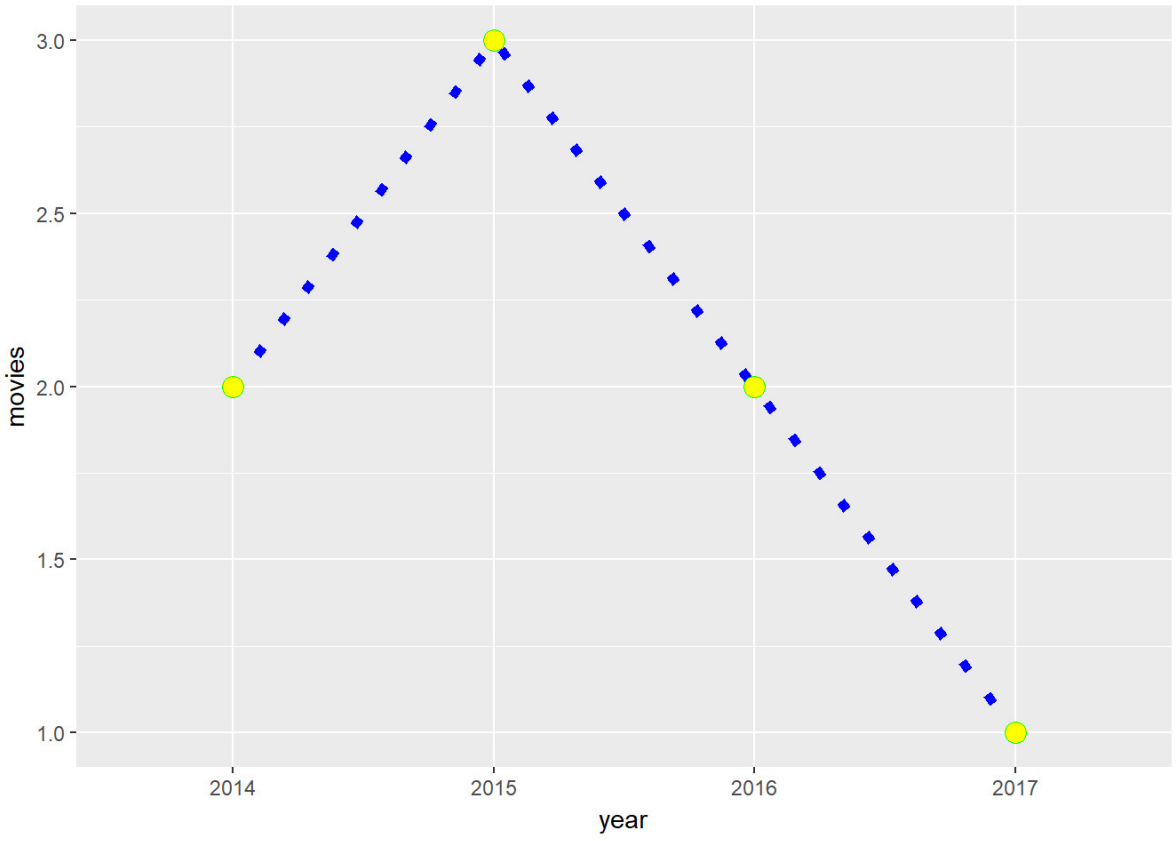
```
actor.plot.set <- ggplot(data = actor.xy, aes(x = year, y = movies, group = 1))
actor.plot.set + geom_line()
```




```
actor.plot.set + geom_line() + geom_point()
```




```
actor.plot.set +
  geom_line(colour = "blue", linetype = "dotted", size = 2) +
  geom_point(colour="green", size = 4, shape = 21, fill = "yellow")
```




```
# geom_point(color="green", size = 8, shape = 18, fill = "yellow")
```

Working with datasets / dataframes

Reading a dataset

```
<dataframe> <- read.csv("<filename>", stringsAsFactors = FALSE)
str(<dataframe>)  # structure of <dataframe>, all variables/columns
head(<dataframe>) # the first few rows
tail(<dataframe>) # the last few rows
```

```
the.beatles.songs <- read.csv("The Beatles songs dataset, v1.csv",
                              stringsAsFactors = FALSE)
```

Examining a dataframe

```
str(<dataframe>)              # structure of <dataframe>, all variables/columns
dim(<dataframe>)              # showing dimensions (numbers of rows and columns) of a dataframe
names(<dataframe>)            # showing column names
head(<dataframe>)             # the first few rows
tail(<dataframe>)             # the last few rows
<dataframe>[ , ]              # the entire dataframe
<dataframe>                   # the entire dataframe
<dataframe>[<m>, ]            # m-th row
<dataframe>[ ,<n>]            # n-th column
summary(<dataframe>$<column>) # summarizing a variable/column values
fix(<dataframe>)              # editing a dataframe
new.df <- edit(<dataframe>)   # editing a dataframe and assigning the modified dataframe to another datavrame
```

```
str(the.beatles.songs)
```

```
## 'data.frame':    310 obs. of  9 variables:
##  $ Title           : chr  "12-Bar Original" "A Day in the Life" "A Hard Day's Night" "A Shot of Rhythm and
Blues" ...
##  $ Year            : chr  "1965" "1967" "1964" "1963" ...
##  $ Album.debut     : chr  "Anthology 2" "Sgt. Pepper's Lonely Hearts Club Band" "UK: A Hard Day's Night U
S: 1962-1966" "Live at the BBC" ...
##  $ Duration        : int  174 335 152 104 163 230 139 NA 124 124 ...
##  $ Other.releases  : int  NA 12 35 NA 29 19 14 9 9 32 ...
##  $ Genre           : chr  "Blues" "Psychedelic Rock, Art Rock, Pop/Rock" "Rock, Electronic, Pop/Rock" "R&
B, Pop/Rock" ...
##  $ Songwriter      : chr  "Lennon, McCartney, Harrison and Starkey" "Lennon and McCartney" "Lennon" "Thomp
son" ...
##  $ Lead.vocal      : chr  "" "Lennon and McCartney" "Lennon, with McCartney" "Lennon" ...
##  $ Top.50.Billboard: int  NA NA 8 NA NA NA 50 41 NA NA ...
```

```
dim(the.beatles.songs)
```

```
## [1] 310   9
```

```
names(the.beatles.songs)
```

```
## [1] "Title"            "Year"             "Album.debut"
## [4] "Duration"         "Other.releases"   "Genre"
## [7] "Songwriter"       "Lead.vocal"       "Top.50.Billboard"
```

```
head(the.beatles.songs)
```

```
##                       Title Year
## 1            12-Bar Original 1965
## 2          A Day in the Life 1967
## 3         A Hard Day's Night 1964
## 4 A Shot of Rhythm and Blues 1963
## 5           A Taste of Honey 1963
## 6        Across the Universe 1968
##                                Album.debut Duration Other.releases
## 1                              Anthology 2      174             NA
## 2      Sgt. Pepper's Lonely Hearts Club Band      335             12
## 3       UK: A Hard Day's Night US: 1962-1966      152             35
## 4                           Live at the BBC      104             NA
## 5 UK: Please Please Me US: The Early Beatles      163             29
## 6                                Let It Be      230             19
##                                Genre
## 1                              Blues
## 2 Psychedelic Rock, Art Rock, Pop/Rock
## 3           Rock, Electronic, Pop/Rock
## 4                       R&B, Pop/Rock
## 5         Pop/Rock, Jazz, Stage&Screen
## 6           Psychedelic folk, Pop/Rock
##                             Songwriter             Lead.vocal
## 1 Lennon, McCartney, Harrison and Starkey
## 2                   Lennon and McCartney   Lennon and McCartney
## 3                             Lennon Lennon, with McCartney
## 4                             Thompson                 Lennon
## 5                        Scott, Marlow              McCartney
## 6                             Lennon                 Lennon
##   Top.50.Billboard
## 1               NA
## 2               NA
## 3                8
## 4               NA
## 5               NA
## 6               NA
```

```
tail(the.beatles.songs)
```

```
##                               Title Year
## 305                    You'll Be Mine 1960
## 306    You're Going to Lose That Girl 1965
## 307 You've Got to Hide Your Love Away 1965
## 308    You've Really Got a Hold on Me 1963
## 309                      Young Blood 1963
## 310           Your Mother Should Know 1967
##                                       Album.debut Duration
## 305                                   Anthology 1       98
## 306                                         Help!      140
## 307                                         Help!      131
## 308 UK: With the Beatles US: The Beatles Second Album      182
## 309                                Live at the BBC      116
## 310                            Magical Mystery Tour      149
##     Other.releases                                                Genre
## 305             NA                           R&B, Experimental, Pop/Rock
## 306              6                                       Rock, Pop/Rock
## 307             12                                         FolkPop/Rock
## 308              2                                       Soul, Pop/Rock
## 309             NA                                             Pop/Rock
## 310             13 Music Hall, Vaudeville Rock, Psychedelic Pop, Pop/Rock
##               Songwriter          Lead.vocal Top.50.Billboard
## 305 Lennon and McCartney           McCartney               NA
## 306               Lennon              Lennon               NA
## 307               Lennon              Lennon               NA
## 308             Robinson Lennon and Harrison               NA
## 309      Leiber, Stoller            Harrison               NA
## 310            McCartney           McCartney               NA
```

```
the.beatles.songs[4, ]
```

```
##                       Title Year     Album.debut Duration Other.releases
## 4 A Shot of Rhythm and Blues 1963 Live at the BBC      104             NA
##           Genre Songwriter Lead.vocal Top.50.Billboard
## 4 R&B, Pop/Rock   Thompson     Lennon               NA
```

```
the.beatles.songs[ ,2]
```

```
##   [1] "1965"      "1967"      "1964"      "1963"      "1963"
##   [6] "1968"      "1965"      "1961"      "1963"      "1963"
##  [11] "1969"      "1967"      "1967"      "1964"      "1966"
##  [16] "1963"      "1965"      "1964"      "1962"      "1963"
##  [21] "1964"      "1967"      "1968"      "1965"      "1963"
##  [26] "1963"      "1969"      "1969"      "1967"      "1968"
##  [31] "1968"      "1967"      "1963"      "1962"      "1964"
##  [36] "1963"      "1969"      "1962"      "1960"      "1963"
##  [41] "1968"      "1967"      "1968"      "1963"      "1969"
##  [46] "1969"      "1968"      "1961"      "1963"      "1965"
##  [51] "1968"      "1963"      "1969"      "1969"      "1965"
##  [56] "1963"      "1966"      "1963"      "1963"      "1969"
##  [61] "1968"      "1965"      "1964"      "1966"      "1968"
##  [66] "1964"      "1968"      "1964"      "196?"      "1967"
##  [71] "1967"      "1966"      "1969"      "1977/1994" "1963"
##  [76] "1963"      "1969"      "1967"      "1965"      "1963"
##  [81] "1968"      "1969"      "1966"      "1967"      "1968"
##  [86] "1969"      "1966"      "1960"      "1968"      "1968"
##  [91] "1962"      "1967"      "1965"      "1968"      "1969"
##  [96] "1969"      "1966"      "1968"      "1968"      "1963"
## [101] "1963"      "1964"      "1968"      "1962"      "1967"
## [106] "1964"      "1964"      "1964"      "1964"      "1963"
## [111] "1963"      "1963"      "1962"      "1970"      "1965"
## [116] "1963"      "1964"      "1963"      "1963"      "1966"
## [121] "1969"      "1968"      "1964"      "1963"      "1964"
## [126] "1964"      "1963"      "1963"      "1964"      "1965"
## [131] "1963"      "1964"      "1963"      "1965"      "1966"
## [136] "1968"      "1962"      "1963"      "1969"      "1965"
## [141] "1964"      "1965"      "1965"      "1965"      "1958"
## [146] "1963"      "1967"      "1965"      "1969"      "1967"
## [151] "1964"      "1968"      "1968"      "1964"      "1963"
## [156] "1964"      "1968"      "1964"      "1963"      "1969"
## [161] "1962"      "1963"      "1963"      "1964"      "1968"
## [166] "1962"      "1962"      "1962"      "1966"      "1967"
## [171] "1963"      "1967"      "1969"      "1969"      "1967"
## [176] "1969"      "1968"      "1964"      "1969"      "1969"
## [181] "1963"      "1965"      "1963"      "1963"      "1963"
## [186] "1968"      "1964"      "1961"      "1964"      "1965"
## [191] "1963"      "1968"      "1963"      "1965"      "1968"
## [196] "1969"      "1969"      "1969"      "1969"      "1964"
## [201] "1967"      "1963"      "1962"      "1966"      "1966"
## [206] "1968"      "1963"      "1962"      "1969"      "1966"
## [211] "1980/1995" "1968"      "1968"      "1968"      "1969"
## [216] "1964"      "1968"      "1963"      "1965"      "1968"
## [221] "1962"      "1962"      "1968"      "1967"      "1967"
## [226] "1969"      "1969"      "1963"      "1966"      "1964"
## [231] "1967"      "1964"      "1964"      "1964"      "1963"
## [236] "1963"      "1963"      "1969"      "1968"      "1968"
## [241] "1966"      "1969"      "1963"      "1963"      "1962"
## [246] "1969"      "1966"      "1969"      "1965"      "1964"
## [251] "1963"      "1965"      "1958"      "1963"      "1969"
## [256] "1968"      "1969"      "1967"      "1963"      "1968"
## [261] "1969"      "1965"      "1962"      "1962"      "1965"
## [266] "1963"      "1964"      "1965"      "1963"      "1962"
## [271] "1965"      "1963"      "1962"      "1963"      "1966"
## [276] "1963"      "1963"      "1969"      "1965"      "1969"
## [281] "1965"      "1965"      "1964"      "1968"      "1964"
## [286] "1966"      "1968"      "1968"      "1968"      "1960"
## [291] "1967"      "1967"      "1965"      "1964"      "1966"
## [296] "1968"      "1965"      "1965"      "1964"      "1967"
```

```
## [301] "1964"      "1965"      "1969"      "1965"      "1960"
## [306] "1965"      "1965"      "1963"      "1963"      "1967"
```

```
summary(the.beatles.songs$Duration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    23.0   130.0   149.0   160.6   176.0   502.0      29
```

```
summary(the.beatles.songs$Title)
```

```
##    Length     Class      Mode
##       310 character character
```

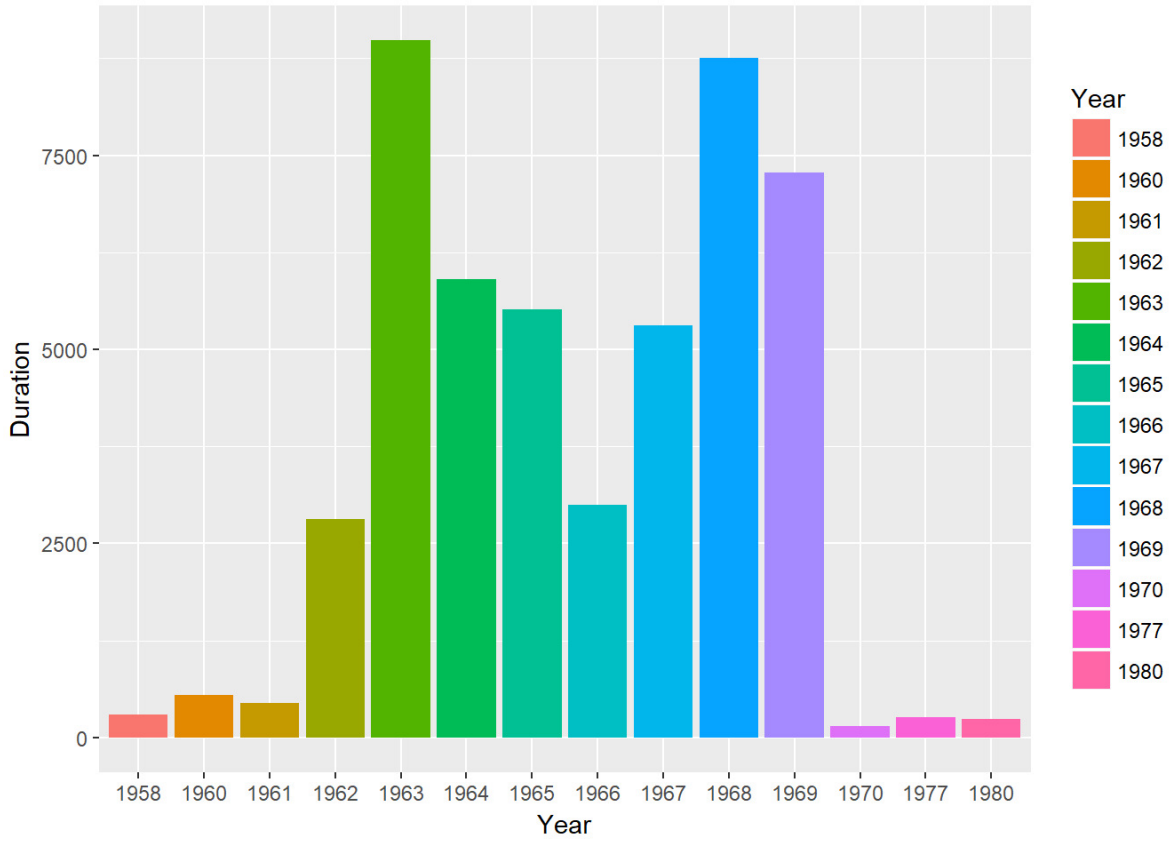
```
summary(the.beatles.songs$Year)
```

```
##    Length     Class      Mode
##       310 character character
```

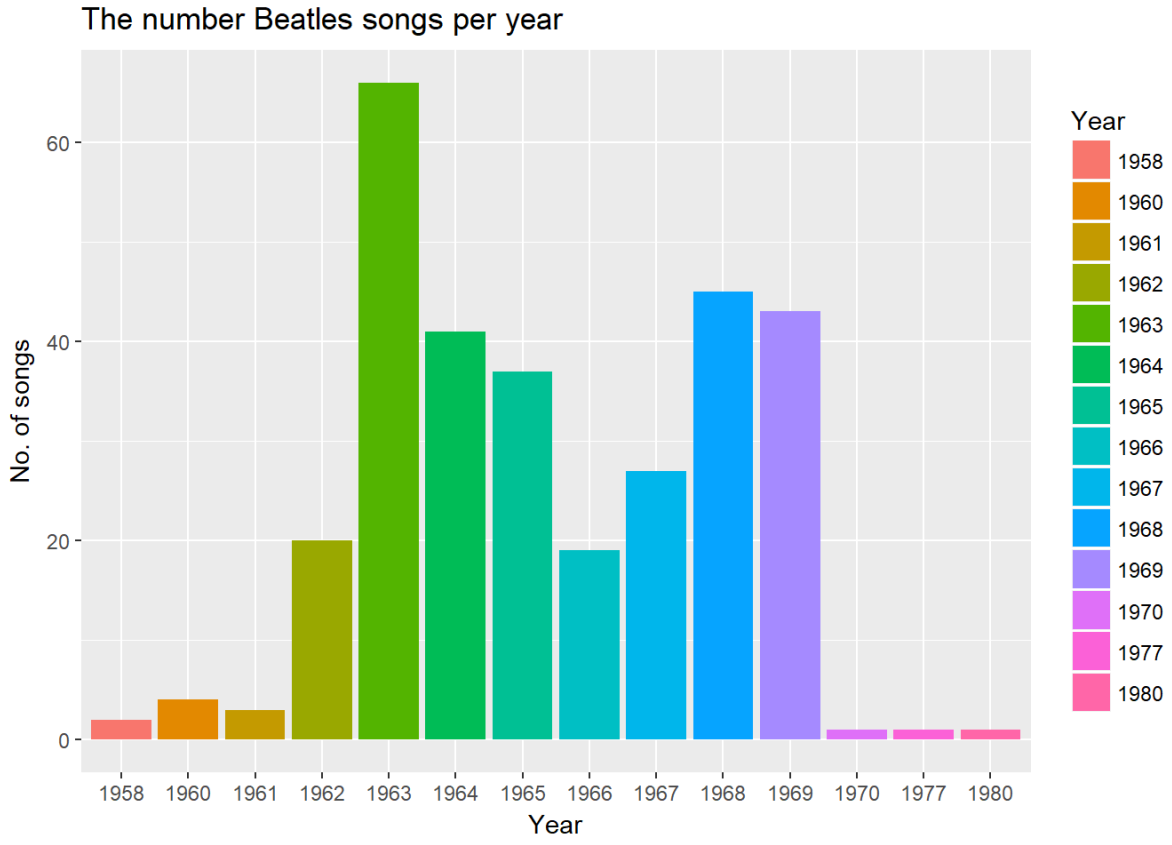
```
# fix(the.beatles.songs)
# a.data.frame.1 <- edit(a.data.frame)
```

Examining a dataframe visually, with ggplot()

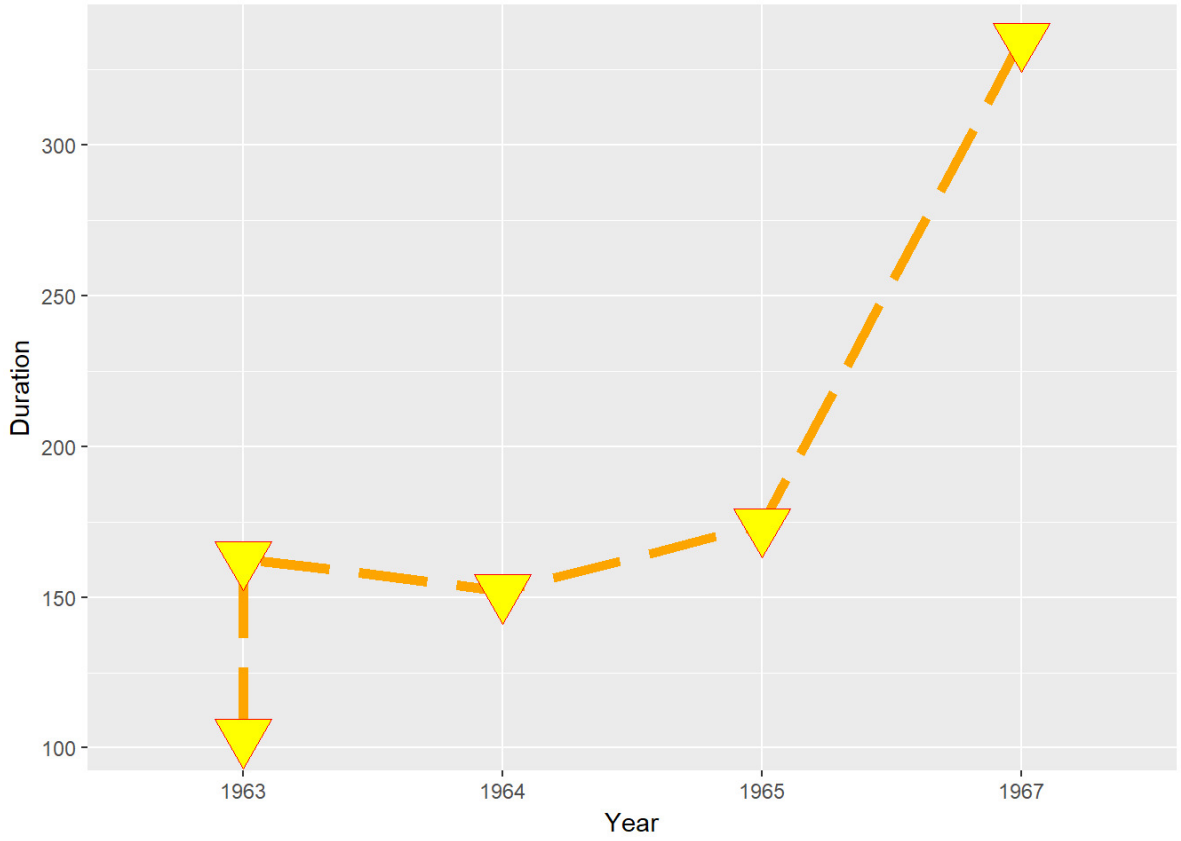
```
the.beatles.songs.clean <- read.csv("The Beatles songs dataset, v1, no NAs.csv",
                            stringsAsFactors = FALSE)
the.beatles.songs.clean$Year <- factor(the.beatles.songs.clean$Year)      # because write.csv/read.csv produc
es int's
g1 <- ggplot(data = the.beatles.songs.clean, aes(x = Year, y = Duration, fill = Year))
g1 + geom_bar(stat = "identity")
```




```
g2 <- ggplot(data = the.beatles.songs.clean, aes(x = Year, fill = Year))
g2 + geom_bar(stat = "count") +
  xlab("Year") + ylab("No. of songs") +
  ggtitle("The number Beatles songs per year")
```



```
g3 <- ggplot(the.beatles.songs.clean[1:5, ], aes(x = Year, y = Duration, group = 1))
g3 + geom_line(color = "orange", size = 2, linetype = "longdash") +
     geom_point(color = "red", shape = 25, size = 8, fill = "yellow")
```



Adding/Removing columns to/from a dataframe

```
<dataframe>$<new column name> <- <default value>  # adding a new column (default values)
<dataframe>$<column name> <- NULL                 # removing a column
```

```
the.beatles.songs$Not.on.album <- FALSE
the.beatles.songs$Not.on.album <- NULL
the.beatles.songs$On.album <- FALSE
the.beatles.songs$On.album[the.beatles.songs$Album.debut != ""] <- TRUE
```

Adding new rows to a dataframe

In case of adding one new row, it must be a 1-line dataframe with the same column names. It is also possible to add an entire dataframe to the existing one (with the same column names).

```
<new row> <- data.frame(<column name 1> = <value 1>, <column name 2> = <value 2>,...)
<new data frame> <- rbind(<dataframe>, <new row>) # append new row to the end of the existing dataframe
<new data frame> <- rbind(<dataframe>[1:i, ],     # insert new row in the middle
+                         <new row>,
+                         <dataframe>[(i + 1):nrow(<dataframe>), ])
```

```
new.song <- data.frame(the.beatles.songs[1, ])
the.beatles.songs <- rbind(the.beatles.songs, new.song)
the.beatles.songs <- rbind(the.beatles.songs[1:3, ],    # Rstudio keeps the original row numbers in View()
                           new.song,
                           the.beatles.songs[4:nrow(the.beatles.songs), ])
```

Removing rows from a dataframe

```
<dataframe>[-i, ]                           # show dataframe without i-th row
<dataframe>[-c(i, j, k), ]                  # show dataframe without rows i, j, k
<dataframe> <- <dataframe>[-i, ]            # remove i-th row from dataframe
<dataframe> <- <dataframe>[-c(i, j, k), ]   # remove rows i, j, k from dataframe
<dataframe> <- <dataframe>[-(i:k), ]        # remove rows i to k from dataframe
```

```
nrow(the.beatles.songs)
```

```
## [1] 312
```

```
the.beatles.songs <- the.beatles.songs[-nrow(the.beatles.songs), ]
the.beatles.songs1 <- the.beatles.songs[-(305:310), ]
the.beatles.songs <- the.beatles.songs[-(1:304), ]
the.beatles.songs <- rbind(the.beatles.songs1, the.beatles.songs)
```

Changing column names

```
colnames(<dataframe>)[i] <- "<new name>"
```

```
colnames(the.beatles.songs)
```

```
##  [1] "Title"            "Year"             "Album.debut"
##  [4] "Duration"         "Other.releases"   "Genre"
##  [7] "Songwriter"       "Lead.vocal"       "Top.50.Billboard"
## [10] "On.album"
```

```
which(colnames(the.beatles.songs) == "Genre")
```

```
## [1] 6
```

```
colnames(the.beatles.songs)[which(colnames(the.beatles.songs) == "Genre")] <- "Song.genre"
colnames(the.beatles.songs)[6] <- "Genre"
```

Changing row names

```
rownames(<dataframe>)[i] <- "<new name>"
rownames(<dataframe>) <- c("<new name 1>", "<new name 2>",...)
rownames(<dataframe>) <- c(1, 2,...)
rownames(<dataframe>) <- list("<new name 1>", <numeric 2>,...)
```

```
rownames(the.beatles.songs) <- paste("song", 1:nrow(the.beatles.songs))
rownames(the.beatles.songs) <- c(1:nrow(the.beatles.songs))
```

Slicing and dicing dataframes

```
<selection> <- <dataframe>[<some rows>, <some columns>]
<selection> <- <dataframe>[i:k, c("<column 1>", "<column 2>",...)]
<indexes> <- with(<dataframe>, which(<condition; can be complex>))  # a with()-which() selection, like an SQL query
<selection> <- <dataframe>[<indexes>, ]
<selection> <- subset(<dataframe>,                                  # subset() is much like SELECT... FROM... WHERE
```

```
+                     <logical condition for the rows to return>,
+                     <select statement for the columns to return>)   # can be omitted;
+                                                                    # column names not prefixed by <dataframe>$
library(dplyr)
<selection> <- filter(<dataframe>,                                  # filter() is from dplyr
+                     <logical condition for the rows to return>)    # can include column referencing,
+                                                                    # not-prefixed by <dataframe>$
```

```
selected.songs <- the.beatles.songs[1:5, c("Title", "Album.debut")]
# View(selected.songs)
indexes <- with(the.beatles.songs, which((Year == "1964") & (Lead.vocal != "McCartney")))
selected.songs <- the.beatles.songs[indexes, ]
songs.1958 <- subset(the.beatles.songs, Year == 1958, c("Title", "Album.debut"))
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
filter(the.beatles.songs,
       as.integer(rownames(the.beatles.songs)) < 33 & Title == "12-Bar Original")
```

```
##             Title Year Album.debut Duration Other.releases Genre
## 1 12-Bar Original 1965 Anthology 2      174             NA Blues
## 2 12-Bar Original 1965 Anthology 2      174             NA Blues
##                                Songwriter Lead.vocal Top.50.Billboard
## 1 Lennon, McCartney, Harrison and Starkey                          NA
## 2 Lennon, McCartney, Harrison and Starkey                          NA
##   On.album
## 1     TRUE
## 2     TRUE
```

Shuffling rows/columns

```
<dataframe> <- <dataframe>[sample(nrow(<dataframe>)), ]   # shuffle row-wise
<dataframe> <- <dataframe>[, sample(ncol(<dataframe>))]   # shuffle column-wise
```

```
the.beatles.songs <- the.beatles.songs[sample(nrow(the.beatles.songs)), ]
the.beatles.songs <- the.beatles.songs[, sample(ncol(the.beatles.songs))]
```

Replacing selected values in a column

```
<selected var name> <- <dataframe>$<column> == <selected value>
<dataframe>$<column>[<selected var name>] <- <new value>
```

```
empty.album.debut <- the.beatles.songs$Album.debut == ""
empty.album.debut
```

```
##   [1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [12] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [34]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [45] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [56] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [111] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [122] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
## [144] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [155] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [166] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE
## [177] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [188]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [199] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [210] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [221] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
## [232] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [243] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [254] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [276] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [287]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [298] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [309] FALSE FALSE FALSE FALSE
```

```
the.beatles.songs$Album.debut[empty.album.debut] <- "empty"
the.beatles.songs$Album.debut[empty.album.debut] <- ""
```

Applying functions to all elements in rows/columns of a dataframe

```
apply(<dataframe>, <1 | 2>, <function(x) {...}>)  # 1 | 2: apply function(x) by row | column
mapply(function(x, y, ...) {...}, <dataframe>$<column 1>, <dataframe>$<column 2>, ...)
+  # <dataframe>$<column 1> corresponds to x, <dataframe>$<column 2> corresponds to y, ...
+  # alternatively:  <f> <- function(x, y, ...) {...}
+                    mapply(<f>, <dataframe>$<column 1>, <dataframe>$<column 2>, ...)
+                      # <f> is just the function name (!)
+                      # <dataframe>$<column 1> corresponds to x, <dataframe>$<column 2> corresponds to y, ...,
+                      # or can be columns from different dataframes, "independent" vectors,... (of the same length)
sapply(<vector>, FUN = function(x) {...})   # function(x): function to be applied to each element of <vector>
```

```
apply(the.beatles.songs[1, ], 1, function(x) {print(x)})
```

```
##                              Top.50.Billboard
##                                            NA
##                                      Duration
##                                         "156"
##                                    Lead.vocal
##                                      "Lennon"
##                                          Year
##                                        "1963"
##                                         Title
##                     "Lonesome Tears in My Eyes"
##                                   Album.debut
##                              "Live at the BBC"
##                                 Other.releases
##                                            NA
##                                      On.album
##                                        "TRUE"
##                                         Genre
##                                    "Pop/Rock"
##                                    Songwriter
## "J. Burnette, D. Burnette, Burlison, Mortimer"
```

```
##                  164
## Top.50.Billboard NA
## Duration         "156"
## Lead.vocal       "Lennon"
## Year             "1963"
## Title            "Lonesome Tears in My Eyes"
## Album.debut      "Live at the BBC"
## Other.releases   NA
## On.album         "TRUE"
## Genre            "Pop/Rock"
## Songwriter       "J. Burnette, D. Burnette, Burlison, Mortimer"
```

```
apply(the.beatles.songs[1, ], 2, function(x) {print(x)})
```

```
## 164
##  NA
##   164
## "156"
##      164
## "Lennon"
##    164
## "1963"
##                         164
## "Lonesome Tears in My Eyes"
##               164
## "Live at the BBC"
## 164
##  NA
##    164
## "TRUE"
##        164
## "Pop/Rock"
##                                          164
## "J. Burnette, D. Burnette, Burlison, Mortimer"
```

```
##                               Top.50.Billboard
##                                             NA
##                                       Duration
##                                          "156"
##                                    Lead.vocal
##                                      "Lennon"
##                                           Year
##                                         "1963"
##                                          Title
##                     "Lonesome Tears in My Eyes"
##                                   Album.debut
##                              "Live at the BBC"
##                                 Other.releases
##                                             NA
##                                       On.album
##                                         "TRUE"
##                                          Genre
##                                      "Pop/Rock"
##                                     Songwriter
## "J. Burnette, D. Burnette, Burlison, Mortimer"
```

```
mapply(function(x, y) {print(x); print(y)},
       the.beatles.songs[111:113, ]$Title,
       the.beatles.songs[111:113, ]$Year)
```

```
## [1] "Step Inside Love/Los Paranoias"
## [1] "1968"
## [1] "It's Only Love"
## [1] "1965"
## [1] "You Know My Name (Look Up the Number)"
## [1] "1967"
```

```
##        Step Inside Love/Los Paranoias
##                                "1968"
##                        It's Only Love
##                                "1965"
## You Know My Name (Look Up the Number)
##                                "1967"
```

```
sapply(the.beatles.songs[1, ], FUN = function(x) {print(x)})
```

```
## [1] NA
## [1] 156
## [1] "Lennon"
## [1] "1963"
## [1] "Lonesome Tears in My Eyes"
## [1] "Live at the BBC"
## [1] NA
## [1] TRUE
## [1] "Pop/Rock"
## [1] "J. Burnette, D. Burnette, Burlison, Mortimer"
```

```
##                              Top.50.Billboard
##                                            NA
##                                      Duration
##                                         "156"
##                                    Lead.vocal
##                                      "Lennon"
##                                          Year
##                                        "1963"
##                                         Title
##                    "Lonesome Tears in My Eyes"
##                                   Album.debut
##                              "Live at the BBC"
##                                 Other.releases
##                                            NA
##                                     On.album
##                                       "TRUE"
##                                        Genre
##                                    "Pop/Rock"
##                                    Songwriter
## "J. Burnette, D. Burnette, Burlison, Mortimer"
```

Partitioning a dataframe

```
# install.packages('caret')
library(caret)
set.seed(<any specific int>)  # allows for repeating the randomization process exactly
<indexes> <- createDataPartition(<dataframe>$<column>, p = 0.8, list = FALSE)
<partition 1> <- <dataframe>[<indexes>, ]
<partition 2> <- <dataframe>[-<indexes>, ]
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.2
```

```
set.seed(222)
indexes <- createDataPartition(the.beatles.songs$Year, p = 0.8, list = FALSE)
```

```
## Warning in createDataPartition(the.beatles.songs$Year, p = 0.8, list
## = FALSE): Some classes have a single record ( 196?, 1970, 1977/1994,
## 1980/1995 ) and these will be selected for the sample
```

```
the.beatles.songs.p1 <- the.beatles.songs[indexes, ]
the.beatles.songs.p2 <- the.beatles.songs[-indexes, ]
```

Saving a dataset (modified or newly created dataset)

```
write.csv(x = <dataframe>, file = "<filename>", row.names = F)  # do not include the row names (row numbers) column
saveRDS(object = <dataframe or another R object>, file = "<filename>")  # save R object for the next session
<dataframe or another R object> <- readRDS(file = "<filename>")         # restore R object in the next session
```

```
write.csv(the.beatles.songs.p2, "p2.csv", row.names = F)
saveRDS(the.beatles.songs.p2, "p2.RData")
p2 <- readRDS("p2.RData")
```

Data type conversion

```
# Covered above:
# b <- c(1, 2, 2, 2, 3, 1, 1, 4, 5, 4)
# b.as.factor <- as.factor(b)
# levels(b.as.factor)
# e.g., <dataframe> <- as.data.frame(<matrix>)
# str(<dataframe>)
```

Difference between character and factor vectors

```
summary(<character vector>)
summary(as.factor(<character vector>))
```

```
class(the.beatles.songs$Year)
```

```
## [1] "character"
```

```
summary(the.beatles.songs$Year)
```

```
##    Length     Class      Mode
##       312 character character
```

```
summary(as.factor(the.beatles.songs$Year))
```

```
##      1958      196?      1960      1961      1962      1963      1964
##         2         1         4         3        20        66        41
##      1965      1966      1967      1968      1969      1970 1977/1994
##        38        19        28        45        42         1         1
## 1980/1995
##         1
```

Convert numeric to factor

```
<dataframe>$<numeric column with few different values> <-
+ factor(<dataframe>$<numeric column with few different values>,
+        levels = c(0, 1, ..., k), labels = c("<l1>", "<l2>", ..., "<lk>"))
```

```
the.beatles.songs1 <- the.beatles.songs
the.beatles.songs1$Billboard.hit <- 0
the.beatles.songs1$Billboard.hit[!is.na(the.beatles.songs1$Top.50.Billboard)] <- 1
the.beatles.songs1$Billboard.hit <-
  factor(the.beatles.songs1$Billboard.hit, levels = c(0,1), labels = c("N", "Y"))
class(the.beatles.songs1$Billboard.hit)
```

```
## [1] "factor"
```

```
summary(the.beatles.songs1$Billboard.hit)
```

```
##   N   Y
## 263  49
```

```
levels(the.beatles.songs1$Billboard.hit)
```

```
## [1] "N" "Y"
```

Examples

Fixing some values in the.beatles.songs$Year

```
summary(the.beatles.songs$Year)
```

```
##    Length     Class      Mode
##       312 character character
```

```
summary(as.factor(the.beatles.songs$Year))
```

```
##      1958      196?      1960      1961      1962      1963      1964
##         2         1         4         3        20        66        41
##      1965      1966      1967      1968      1969      1970 1977/1994
##        38        19        28        45        42         1         1
## 1980/1995
##         1
```

```
the.beatles.songs$Year[the.beatles.songs$Year == "196?"] <- "1969"
the.beatles.songs$Year[the.beatles.songs$Year == "1977/1994"] <- "1977"
the.beatles.songs$Year[the.beatles.songs$Year == "1980/1995"] <- "1980"
summary(as.factor(the.beatles.songs$Year))
```

```
## 1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1977 1980
##    2    4    3   20   66   41   38   19   28   45   43    1    1    1
```

Creating the.beatles.songs$Billboard.hit column as factor

```
the.beatles.songs$Billboard.hit <- 0
the.beatles.songs$Billboard.hit[!is.na(the.beatles.songs$Top.50.Billboard)] <- 1
the.beatles.songs$Billboard.hit <-
  factor(the.beatles.songs$Billboard.hit, levels = c(0,1), labels = c("N", "Y"))
```

Working with tables

The table() function

```
table(<var>)  # typically a factor or an integer var
```

```
table(the.beatles.songs1$Year)
```

```
##
##      1958      196?      1960      1961      1962      1963      1964
##         2         1         4         3        20        66        41
##      1965      1966      1967      1968      1969      1970 1977/1994
##        38        19        28        45        42         1         1
## 1980/1995
##         1
```

```
table(the.beatles.songs1$Top.50.Billboard)
```

```
##
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## 26 27 28 29 30 31 32 33 34 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

```
table(the.beatles.songs1$Billboard.hit)
```

```
##
##   N   Y
## 263  49
```

```
table(the.beatles.songs1$Billboard.hit)[1]
```

```
##   N
## 263
```

```
x <- table(the.beatles.songs1$Billboard.hit)[1]
x
```

```
##   N
## 263
```

```
y <- as.numeric(x)
y
```

```
## [1] 263
```

The prop.table() function

```
prop.table(table(<var>))
round(prop.table(table(<var>)), digits = <n>)
```

```
prop.table(table(the.beatles.songs1$Billboard.hit))
```

```
##
##         N         Y
## 0.8429487 0.1570513
```

```
round(prop.table(table(the.beatles.songs1$Billboard.hit)), digits = 2)
```

```
##
##    N    Y
## 0.84 0.16
```

Row and column margins

```
table(<var1>, <var2>)                                    # <var1>, <var2>: usually factors or integers
table(<rows title> = <var1>, <columns title> = <var2>)   # add common titles for rows/columns
prop.table(table(<var1>, <var2>), margin = 1)            # all row margins are 1.0
prop.table(table(<var1>, <var2>), margin = 2)            # all column margins are 1.0
```

```
table(the.beatles.songs$Billboard.hit, the.beatles.songs$Year)
```

```
##
##     1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1977 1980
##   N    2    4    1   17   60   31   31   12   25   42   37    1    0    0
##   Y    0    0    2    3    6   10    7    7    3    3    6    0    1    1
```

```
table(Hit = the.beatles.songs$Billboard.hit, Year = the.beatles.songs$Year)
```

```
##    Year
## Hit 1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1977 1980
##   N    2    4    1   17   60   31   31   12   25   42   37    1    0    0
##   Y    0    0    2    3    6   10    7    7    3    3    6    0    1    1
```

```
round(prop.table(table(the.beatles.songs$Billboard.hit, the.beatles.songs$Year), 1), digits = 2)
```

```
##
##     1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1977 1980
##   N 0.01 0.02 0.00 0.06 0.23 0.12 0.12 0.05 0.10 0.16 0.14 0.00 0.00 0.00
##   Y 0.00 0.00 0.04 0.06 0.12 0.20 0.14 0.14 0.06 0.06 0.12 0.00 0.02 0.02
```

```
round(prop.table(table(the.beatles.songs$Billboard.hit, the.beatles.songs$Year), 2), digits = 2)
```

```
##
##     1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1977 1980
##   N 1.00 1.00 0.33 0.85 0.91 0.76 0.82 0.63 0.89 0.93 0.86 1.00 0.00 0.00
##   Y 0.00 0.00 0.67 0.15 0.09 0.24 0.18 0.37 0.11 0.07 0.14 0.00 1.00 1.00
```

```
round(prop.table(table(the.beatles.songs$Billboard.hit, the.beatles.songs$Year)), digits = 2)
```

```
##
##     1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1977 1980
##   N 0.01 0.01 0.00 0.05 0.19 0.10 0.10 0.04 0.08 0.13 0.12 0.00 0.00 0.00
##   Y 0.00 0.00 0.01 0.01 0.02 0.03 0.02 0.02 0.01 0.01 0.02 0.00 0.00 0.00
```

Example: converting the.beatles.songs$Year to factor and showing it in tables

```
factor(the.beatles.songs$Year)
```

```
##   [1] 1963 1962 1965 1963 1962 1967 1964 1968 1967 1968 1969 1963 1969 1967
##  [15] 1965 1963 1968 1963 1968 1962 1969 1964 1963 1962 1964 1966 1963 1965
##  [29] 1964 1965 1968 1967 1963 1962 1968 1962 1965 1961 1969 1966 1962 1969
##  [43] 1967 1963 1966 1967 1967 1968 1965 1963 1963 1965 1964 1969 1965 1968
##  [57] 1961 1965 1965 1964 1969 1963 1965 1964 1965 1969 1963 1969 1964 1964
##  [71] 1963 1963 1964 1964 1964 1966 1969 1965 1968 1963 1963 1968 1963 1968
##  [85] 1964 1964 1964 1968 1965 1969 1963 1963 1964 1969 1963 1964 1965 1980
##  [99] 1961 1967 1967 1964 1969 1962 1967 1965 1963 1964 1969 1968 1968 1965
## [113] 1967 1964 1962 1964 1963 1964 1968 1965 1963 1969 1967 1967 1960 1966
## [127] 1963 1969 1963 1963 1964 1969 1969 1964 1963 1964 1963 1969 1968 1969
## [141] 1966 1969 1967 1965 1968 1968 1969 1969 1963 1963 1969 1964 1965 1963
## [155] 1963 1963 1965 1969 1968 1966 1967 1969 1964 1965 1962 1962 1968 1964
## [169] 1966 1968 1968 1965 1963 1969 1958 1963 1960 1968 1966 1963 1969 1960
## [183] 1963 1965 1967 1968 1966 1962 1963 1966 1963 1977 1969 1965 1963 1964
## [197] 1968 1960 1968 1965 1967 1964 1964 1962 1966 1964 1969 1968 1968 1968
## [211] 1958 1967 1967 1963 1965 1967 1963 1968 1965 1963 1969 1968 1963 1969
## [225] 1969 1968 1965 1965 1966 1962 1968 1965 1963 1963 1968 1967 1963 1963
## [239] 1963 1964 1963 1969 1968 1967 1969 1963 1965 1964 1968 1968 1965 1967
## [253] 1963 1966 1969 1968 1969 1963 1963 1968 1964 1969 1964 1968 1970 1963
## [267] 1963 1962 1969 1967 1969 1963 1962 1969 1968 1964 1963 1964 1968 1964
## [281] 1963 1963 1963 1966 1967 1963 1965 1965 1967 1962 1969 1964 1967 1962
## [295] 1965 1967 1968 1963 1966 1969 1965 1966 1964 1962 1968 1966 1966 1962
## [309] 1963 1968 1963 1965
## 14 Levels: 1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 ... 1980
```

```
the.beatles.songs$Year <- as.factor(the.beatles.songs$Year)
class(the.beatles.songs$Year)
```

```
## [1] "factor"
```

```
summary(the.beatles.songs$Year)
```

```
## 1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1977 1980
##    2    4    3   20   66   41   38   19   28   45   43    1    1    1
```

```
prop.table((table(the.beatles.songs$Year)))
```

```
##
##        1958        1960        1961        1962        1963        1964
## 0.006410256 0.012820513 0.009615385 0.064102564 0.211538462 0.131410256
##        1965        1966        1967        1968        1969        1970
## 0.121794872 0.060897436 0.089743590 0.144230769 0.137820513 0.003205128
##        1977        1980
## 0.003205128 0.003205128
```

```
round(prop.table((table(the.beatles.songs$Year))), digits = 2)
```

```
##
## 1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1977 1980
## 0.01 0.01 0.01 0.06 0.21 0.13 0.12 0.06 0.09 0.14 0.14 0.00 0.00 0.00
```

The xtabs() function

```
xtabs(~<column 1> + <column 2>, <dataframe>)
```

```
xtabs(~Billboard.hit + Year, the.beatles.songs)
```

```
##              Year
## Billboard.hit 1958 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970
##             N    2    4    1   17   60   31   31   12   25   42   37    1
##             Y    0    0    2    3    6   10    7    7    3    3    6    0
##              Year
## Billboard.hit 1977 1980
##             N    0    0
##             Y    1    1
```

Working with vectors

Differences in initializing vectors and dataframe columns

```
<vector> <- rep(<value>, <times>)
<vector> <- <value>
<dataframe>$<column> <- rep(<value>, <times>)
<dataframe>$<column> <- <value>
```

```
v <- rep(0, 5)
v
```

```
## [1] 0 0 0 0 0
```

```
v <- 2
v
```

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

```
df <- data.frame(a = c(1, 2, 3), b = c(4, 5, 6))
df
```

```
##   a b
## 1 1 4
## 2 2 5
## 3 3 6
```

```
df$a <- rep(1, 3)
df
```

```
##   a b
## 1 1 4
## 2 1 5
## 3 1 6
```

```
df$a <- 0
df
```

```
##   a b
## 1 0 4
## 2 0 5
## 3 0 6
```

Counting the number of elements with the values of <x> in a vector

1. <table> <- table(<vector>)
   <table>
   <table>[names(<table>) == <x>]
2. sum(<vector> == <x>)
3. length(which(<vector> == <x>))   # which() is like WHERE in SQL

```
v <- c(1, 2, 1, 3, 2, 4, 5, 1, 3, 1)
t <- table(v)
t
```

```
## v
## 1 2 3 4 5
## 4 2 2 1 1
```

```
t[names(t) == 1]
```

```
## 1
## 4
```

```
t[names(t) == "1"]
```

```
## 1
## 4
```

```
sum(v == 1)
```

```
## [1] 4
```

```
length(which(v == 1))
```

```
## [1] 4
```

Appending an element to a vector

```
<vector> <- append(<vector>, <element>)   # type conversion occurs if <element> is of different type than v[i]
<vector> <- append(<vector>, <element>, after = <n>)   # insert <=> append at a desired location
<vector> <- append(<vector>, NA)
```

```
v <- c(1, 2, 1, 3, 2, 4, 5, 1, 3, 1)
v
```

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

```
v <- append(v, NA)
v <- append(v, NA, after = 5)
v
```

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

```
v <- append(v, "s")
v
```

```
##  [1] "1" "2" "1" "3" "2" NA  "4" "5" "1" "3" "1" NA  "s"
```

Removing NAs from a vector in NA-sensitive functions

`<function>(<vector>, na.rm = TRUE)`

```
v <- c(1, 2, 1, 3, 2, 4, 5, 1, 3, 1)
v
```

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

```
v <- append(v, NA)
v <- append(v, NA)
mean(v)
```

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

```
mean(v, na.rm = TRUE)
```

```
## [1] 2.3
```

Selecting vector elements that match criteria, with controlling for NAs and NaNs

`<numeric vector> <- c(<n1>, <n2>, <n3>, ..., NA, ...NaN)`

`<selected> <- <numeric vector>[<logical criterion> & !is.na(<numeric vector>)]  # is.na() is TRUE for both NA and NaN`

Using `is.na()` is the only way to test if `<something>` is NA (`<something> == NA` does not work).

```
v <- c(1, 2, 1, 3, NA, 4, 5, 1, 3, NaN, 1)
v
```

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

```
v <- v[v > 1 & !is.na(v)]
v
```

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

Working with strings

Some of the basic string functions

```
# install.packages("stringr")
library(stringr)
nchar(<s>)                              # string length
str_length(<s>)                         # string length; str_length() is from stringr
substr(<s>, <start index>, <end index>) # substring
toupper(<s>)                            # to upper case letters
tolower(<s>)                            # to lower case letters
grepl(<s1>, <s2>)                       # contains; TRUE if <s2> contains <s1>
str_detect(<s1>, <s2>)                  # contains; TRUE if <s1> contains <s2>; str_detect() is from stringr
paste(<s1>, <s2>, sep = "")             # concatenate (result: <s1><s2>; <s1> <s2>, if sep = "" omitted)
sub(<s1>, <s2>, <s>)                    # substring replacement: replace <s1> in <s> with <s2>
strsplit(<s>, <regex>)                  # split (the type of the result is list)
```

```
# install.packages("stringr")
library(stringr)
title <- the.beatles.songs$Title[13]
title
```

```
## [1] "She Came in Through the Bathroom Window"
```

```
nchar(title)
```

```
## [1] 39
```

```
str_length(title)
```

```
## [1] 39
```

```
grepl("You", title)
```

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

```
str_detect(title, "You")
```

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

Splitting strings to words

```
strsplit(<s>, <regex>)                  # split (the type of the result is list)
```

```
title <- the.beatles.songs$Title[13]
words.in.title <- strsplit(title, " ")
words.in.title
```

```
## [[1]]
## [1] "She"      "Came"     "in"       "Through"  "the"      "Bathroom"
## [7] "Window"
```

```
words.in.title <- strsplit(title, " ")
words.in.title
```

```
## [[1]]
## [1] "She"      "Came"     "in"       "Through"  "the"      "Bathroom"
## [7] "Window"
```

```
words.in.title <- strsplit("All  My    Loving", " ")
words.in.title
```

```
## [[1]]
## [1] "All"    ""       "My"     ""       ""       ""       "Loving"
```

```
words.in.title <- unlist(words.in.title)
words.in.title <- words.in.title[words.in.title != ""]
words.in.title
```

```
## [1] "All"    "My"     "Loving"
```

```
title <- paste(words.in.title[1], words.in.title[2], words.in.title[3])
title
```

```
## [1] "All My Loving"
```

```
title <- paste(words.in.title[1], words.in.title[2], words.in.title[3], sep = "")
title
```

```
## [1] "AllMyLoving"
```

Resources, readings, references

R Tutorials, http://www.endmemo.com/program/R/ (http://www.endmemo.com/program/R/)
R: A Beginner's Guide (by Sharon Machlis), http://www.tfrec.wsu.edu/TFREConly/r4beginners_v3.pdf (http://www.tfrec.wsu.edu/TFREConly/r4beginners_v3.pdf)
Graphs with ggplot2, http://www.cookbook-r.com/Graphs/ (http://www.cookbook-r.com/Graphs/)