Data analysis and visualization in R

UC Merced R curriculum

Andrea Sánchez-Tapia
Rio de Janeiro Botanical Garden - ¡liibre! - RLadies+

2020-10-22

last time

- we setup a project and its file structure
- we started using R inside RStudio
- we created numerical, character and logical vectors with `c()`
- we learned to subset vectors with brackets `[]` and other functions: `length()`, `:`, `seq(from, to, interval)`
- vector subset can be done via numeric or logical indexes

data structures in R

- **vector**: lineal arrays (one dimension: only length)
- **matrices**: arrays of vectors of the same type (all numeric or all character, for instance) (two dimensions: width and length)
- **data frames**: two-dimensional structures ("rectangular") but might be of combined types (i.e., column 1 with names, column 2 with numbers)
- **factors**: vectors (one-dimensional) representing **categorical variables** and thus having **levels**
- **lists**: literally lists, of objects that can be of any type (a list of data frames, or different objects)
- **arrays** are similar to matrices and dataframes but may be three-dimensional ("layered" data frames)

matrices

- data have to be of the same type

```
?matrix
matrix(nrow = 4, ncol = 3)
```

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

matrices

```
nums <- 1:12
matrix(data = nums, nrow = 3)
matrix(data = nums, nrow = 3, byrow = TRUE)
m <- matrix(data = nums, nrow = 3, byrow = TRUE,
       dimnames = list(c(LETTERS[1:3]), letters[1:4]))
m
```

- matrix algebra

starting with data

the survey dataset

- Data frames: one row per sampling unit (individual), one column per variable

```
knitr::include_graphics("./docs/figs/columns.png")
```

downloading the dataset

We are going to download the file to our `data/raw` sub folder:

```
download.file(url = "https://ndownloader.figshare.com/files/2292169",
              destfile = "data/raw/portal_data_joined.csv")
```

reading files into R

Functions to read data are key to any project. for data frames: `read.csv()`, `read.delim()`

```
surveys <- read.csv("data/raw/portal_data_joined.csv")
surveys_check <- read.table(file = "data/raw/portal_data_joined.csv",
                            sep = ",",
                            header = TRUE)
identical(surveys, surveys_check)
```

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

reading files into R

- Package **readr**
- Package **data.table** (`data.table::fread()`) when you need to open a large file
- Excel spreadsheets: `readxl::read_excel()`
- **Graphic interface**

There are **many other ways** to read data into R, some are specific for the type of data (GIS shapefiles or raster, and specific packages may come with their own reader functions)

basic functions to explore dataframes

```
str(surveys)
dim(surveys)
nrow(surveys)
ncol(surveys)
head(surveys) # 6 rows by default
head(surveys)[,1:3]
tail(surveys)
names(surveys)
rownames(surveys)
length(surveys) # number of columns
summary(surveys)
```

inspecting `data.frame` objects

Based on the output of `str(surveys)`, can you answer the following questions?

- What is the class of the object surveys?
- How many rows and how many columns are in this object?
- What is the type of data of the columns?

indexing and subsetting data frames

- a vector has only one dimension, so:
  - `length()` refers to number of **elements**
  - `dim()`
  - selection between brackets `[]`
- a data.frame has **two** dimensions, so `dim()`, `ncol()`, `nrow()` selection between brackets `[]` **BUT with the two dimensions separated by a comma**: `[rows, columns]`

indexing and subsetting data frames

```
names(surveys)
surveys[, 6]
surveys[1, ]
surveys[ , 13]
surveys[4, 13]
```

indexing and subsetting data frames

```
sub <- surveys[1:10,]
# first element in the first column of the data frame

# first element in the 6th column

# first column of the data frame

# first three elements in the 7th column

# the 3rd row of the data frame
```

indexing and subsetting data frames

```
# first element in the first column of the data frame
sub[1, 1]
# first element in the 6th column
sub[1, 6]
# first column of the data frame
sub[, 1]
# first column of the data frame
sub[1]
# first three elements in the 7th column (as a vector)
sub[1:3, 7]
# the 3rd row of the data frame
sub[3, ]
# equivalent to head_surveys <- head(surveys)
head_surveys <- surveys[1:6, ]
```

indexing and subsetting data frames

- minus sign to **remove** the indexed column or row

```
# The whole data frame, except the first column
surveys[, -1]
surveys[-(7:34786), ] # Equivalent to head(surveys)
```

selecting columns by name

```
surveys["species_id"]       # Result is a data.frame
surveys[, "species_id"]     # Result is a vector
surveys[["species_id"]]     # Result is a vector
surveys$species_id          # Result is a vector
```

- R has several ways to do some things

challenge

- Create a data.frame (`surveys_200`) containing only the data in row 200 of the `surveys` dataset
- Notice how `nrow()` gave you the number of rows in a data.frame? Use that number to pull out just that last row in the data frame
- Compare that with what you see as the last row using `tail()` to make sure it's meeting expectations
- Pull out that last row using `nrow()` instead of the row number.
- Create a new data frame (`surveys_last`) from that last row.
- Use `nrow()` to extract the row that is in the middle of the data frame. Store the content of this row in an object named `surveys_middle`.
- Combine `nrow()` with the - notation above to reproduce the behavior of `head(surveys)`, keeping just the first through 6th rows of the surveys dataset.

some basic plotting

```
x <- surveys$weight
y <- surveys$hindfoot_length
plot(x, y)
```

plotting a single variable

```
plot(surveys$hindfoot_length)
```

factors

- **factors**: vectors (one-dimensional) representing **categorical variables** and thus having **levels**. ordered (`c("low", "medium", "high")` or unordered (`c("green", "blue", "red")`)

- R < 4.0 had a default behavior `stringsAsFactors = TRUE` so any character column was transformed into a factor

```
`?read.csv()`
?default.stringsAsFactors
```

today if we want factors we have to transform the vectors

factors

```
## Compare the difference between our data read as
#`factor` vs `character`.
surveys <- read.csv("data_raw/portal_data_joined.csv",
                    stringsAsFactors = TRUE)
str(surveys)
surveys <- read.csv("data_raw/portal_data_joined.csv",
                    stringsAsFactors = FALSE)
str(surveys)
```

factors

Convert the column "plot_type" and "sex" into a factor:

```
surveys$plot_type <- factor(surveys$plot_type)
surveys$sex <- factor(surveys$sex)
```

```
head(surveys$plot_type)
```

```
## [1] Control Control Control Control Control Control
## 5 Levels: Control Long-term Krat Exclosure ... Spectab exclosure
```

```
levels(surveys$plot_type)
```

```
## [1] "Control"                 "Long-term Krat Exclosure"
## [3] "Rodent Exclosure"        "Short-term Krat Exclosure"
## [5] "Spectab exclosure"
```

```
plot(surveys$weight ~ surveys$plot_type)
```

working with factors

```
sex <- factor(c("male", "female", "female", "male"))
levels(sex) # in alphabetical order!
nlevels(sex)
sex
sex <- factor(sex, levels = c("male", "female"))
sex # after re-ordering
as.character(sex)

year_fct <- factor(c(1990, 1983, 1977, 1998, 1990))
as.numeric(year_fct)                # Wrong! And there is no warning...
as.numeric(as.character(year_fct)) # Works...
as.numeric(levels(year_fct))
as.numeric(levels(year_fct))[year_fct]    # The recommended way.
```

let's make a plot of a factor variable

```
plot(as.factor(surveys$sex))
```

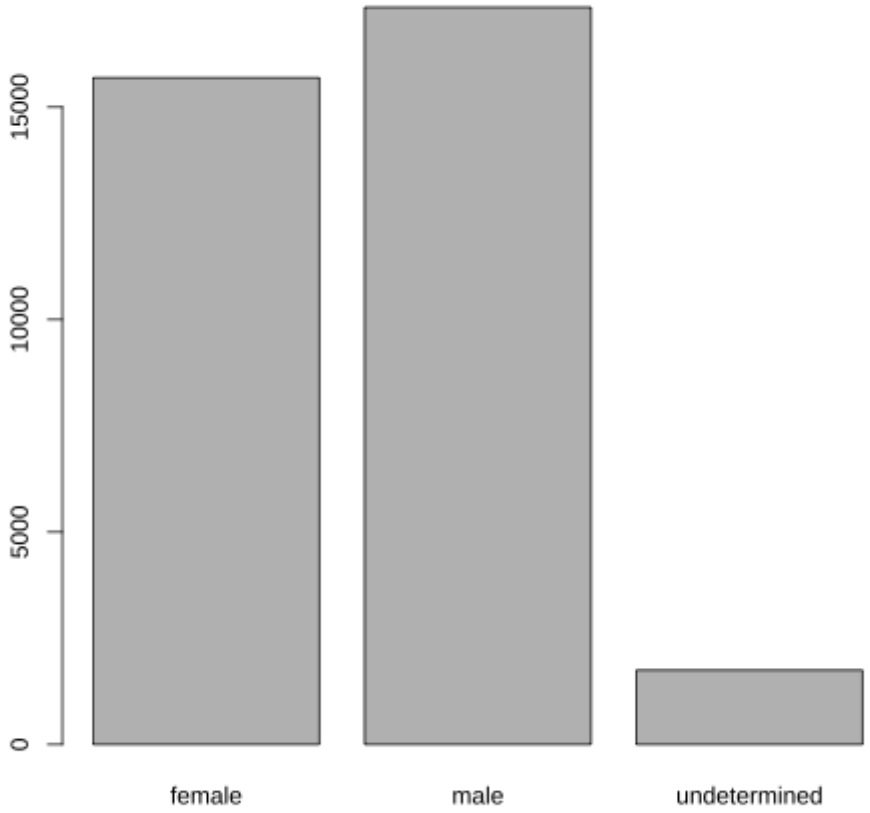
let's rename this label

let's make a plot of a factor variable

```
plot(sex)
```

let's rename this label

challenge



- Rename “F” and “M” to “female” and “male” respectively.
- Now that we have renamed the factor level to “undetermined”, can you recreate the barplot such that “undetermined” is last (after “male”)?

working with several tables

- in real analysis settings you will have many tables that are related
- in ecology for example:
  - sites x species
  - sites x environmental conditions
  - species x characteristics
  - individuals x individual measurement

working with several tables

```
download.file("https://ndownloader.figshare.com/files/3299483",
              "data/raw/species.csv")
download.file("https://ndownloader.figshare.com/files/10717177",
              "data/raw/surveys.csv")
download.file("https://ndownloader.figshare.com/files/3299474",
              "data/raw/plots.csv")
```

```
library(readr)
species <- read.csv("data/raw/species.csv")
surveys <- read.csv("data/raw/surveys.csv")
plots <- read.csv("data/raw/plots.csv")
surveys_plots <- merge(surveys, plots)
dim(surveys)
```

```
## [1] 35549     9
```

```
dim(plots)
```

```
## [1] 24  2
```

```
dim(surveys_plots)
```

```
## [1] 35549    10
```

dplyr joins