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

- you can also fit vectors with the correct dimensions

```
nums <- 1:12
matrix(data = nums, nrow = 3)
matrix(data = nums, nrow = 3, byrow = TRUE)
```

matrices

- naming a matrix: `NULL or a list of length 2 giving the row and column names respectively.` (?) look at the examples!

matrices

```
dim1_names <- c("row1", "row2", "row3")
dim2_names <- c("col1", "col2", "col3", "col4")
names_matrix <- list(dim_1 = dim1_names, dim_2 = dim2_names)
str(names_matrix)
m <- matrix(data = nums, nrow = 3,
        dimnames = names_matrix)
dim(m)
dimnames(m)
```

- you can convert easily between data types `data.frame("m")`, `as.data.frame("m")`, `as.vector(m)`. (the same goes for changes between "numeric", "logical")

starting with data frames

the survey dataset

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

Column	Description
record_id	Unique id for the observation
month	month of observation
day	day of observation
year	year of observation
plot_id	ID of a particular plot
species_id	2-letter code
sex	sex of animal (“M”, “F”)
hindfoot_length	length of the hindfoot in mm
weight	weight of the animal in grams
genus	genus of animal
species	species of animal
taxon	e.g. Rodent, Reptile, Bird, Rabbit
plot_type	type of plot

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)

inspecting `data.frame` objects

```
str(surveys)
dim(surveys)
nrow(surveys)
ncol(surveys)
head(surveys) # 6 rows by default
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: `dim()`, `ncol()`, `nrow()` selection between brackets `[]` **BUT with the two dimensions separated by a comma**: `[rows, columns]`
- we'll try to refer to these operations as **selecting columns** and **filtering rows**

selecting columns

- with numeric indexes and vectors

```
surveys[, 6]
surveys[1, ]
surveys[ , 13]
surveys[4, 13]
surveys[1:4, 1:3]
```

indexing and subsetting data frames

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

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

selecting columns by name

```
names(surveys)

  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

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 (as a vector)

# first column of the data frame (as a dataframe)

# first three elements in the 7th column (as a vector)

# the 3rd row of the data frame

# equivalent to head_surveys <- head(surveys)
```

indexing and subsetting data frames

```
sub <- surveys[1:10,]
# 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 (as a vector)
sub[, 1]
# first column of the data frame (as a dataframe)
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, ]
```

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.

dealing with missing data

```
sub <- surveys[1:10,]
#str(sub)
sub$hindfoot_length
sub$hindfoot_length == NA #it cannot compare! because it's NA
#we use is.na:
#is.na(sub$hindfoot_length) # yes! returns a logical vector
sub$hindfoot[!is.na(sub$hindfoot_length)]
```

dealing with missing data

- in some functions: `na.rm`

```
mean(sub$hindfoot_length)
```

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

```
mean(sub$hindfoot_length, na.rm = T)
```

```
## [1] 31.5
```

dealing with missing data

- Dealing with missing data in dataframes: **filtering rows that have NAs**

```
non_NA_w <- surveys[!is.na(surveys$weight),]
dim(non_NA_w)
```

```
## [1] 32283    13
```

```
non_NA <- surveys[!is.na(surveys$weight) &
                  !is.na(surveys$hindfoot_length),]
dim(non_NA)
```

```
## [1] 30738    13
```

dealing with NAs

```
#complete.cases(surveys)
surveys1 <- surveys[complete.cases(surveys) , ]
surveys2 <- na.omit(surveys)
dim(surveys1)
```

```
## [1] 30738    13
```

```
dim(surveys2)
```

```
## [1] 30738    13
```

write csv objects to disk

```
write.csv(surveys1, "data/processed/surveys_mod.csv")
```

remember you never overwrite your original, raw data!

read the modified csv

```
surveys <- read.csv("data/processed/surveys_mod.csv")
str(surveys)
```

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 = FALSE)
str(surveys)

surveys <- read.csv("data/raw/portal_data_joined.csv",
                    stringsAsFactors = TRUE)
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)
```

(actually this is a way to create new columns)

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)
```

let's make a plot of a factor variable

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

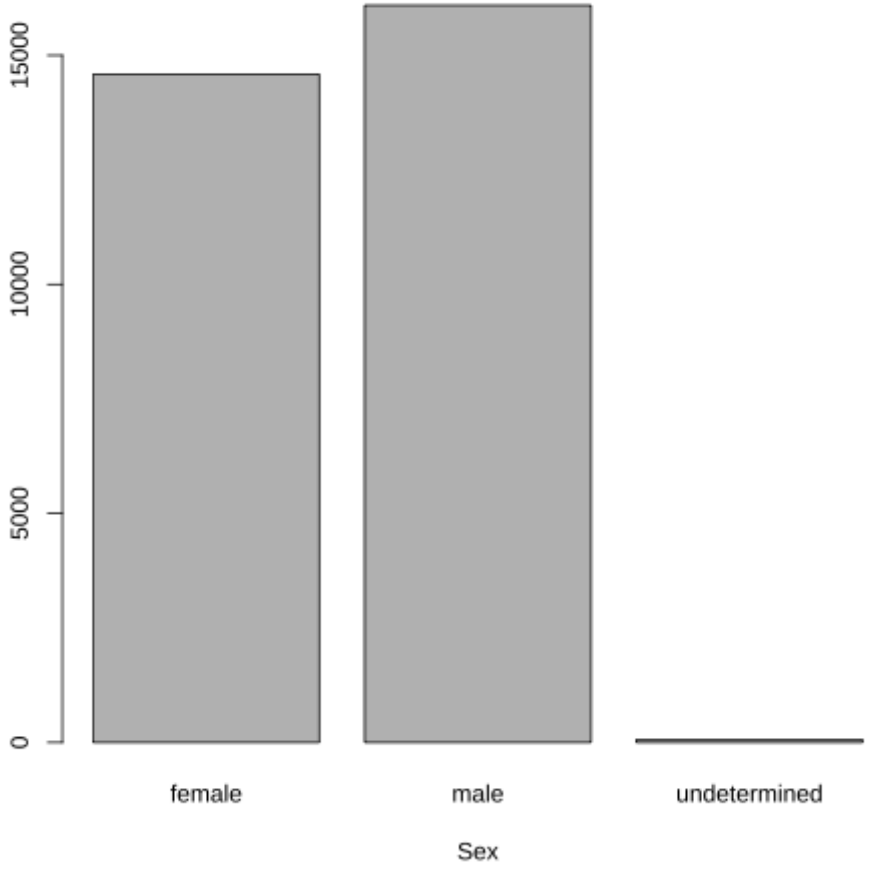
let's rename this label

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”)?

some basic plotting

```
plot(surveys$hindfoot_length)
plot(surveys$weight)
plot(sort(surveys$hindfoot_length))
plot(sort(surveys$weight))
```

scatterplots

- two continuous variables

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

boxplots

```
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"
```

boxplots

plotting basics

all parameters for plotting are in function `par()`

```
plot(x, y)
```

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
```

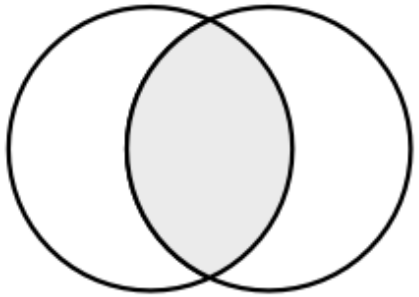
```
dim(plots)
```

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

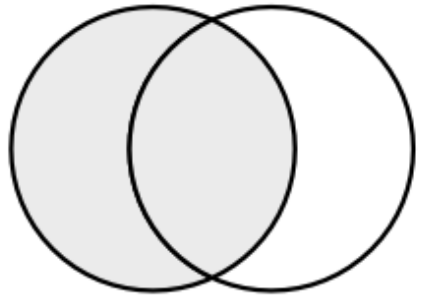
```
dim(surveys_plots)
```

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

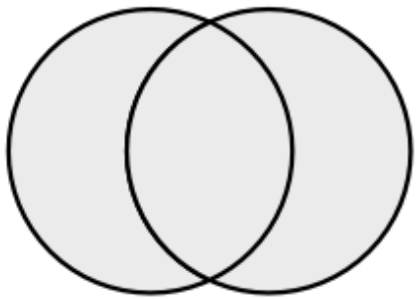
dplyr joins



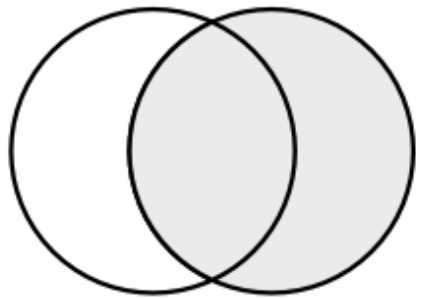
`inner_join(x, y)`



`left_join(x, y)`



`full_join(x, y)`



`right_join(x, y)`