CeDoSIA SS2020 - Exercise Sheet 2: Data Analysis and Visualization

Vangelis Theodorakis, Xueqi Cao, Daniela Andrade Salazar, Julien Gagneur

22 June, 2020

Package

BiocStyle 2.14.4

Contents

1 Setup

```r
library(data.table)
library(magrittr)   # Needed for %>% operator
library(tidyr)
library(readxl)
library(dplyr)
```

2 Introduction to ggplot

The `iris` data is included in the ggplot2 package. First load `ggplot2` package, then check `iris` data with `head(iris)`.

1) Are there any relationships/correlations between petal length and width? How would you show it?
2) Do petal lengths and widths correlate in every species?
3) Fit a regression model and visualize the regression line `geom_smooth()`. Add this as an extra layer on the plot of 1).

```r
# Qestion 1
data(iris)
ggplot(iris, aes(Petal.Length, Petal.Width)) +
  geom_point()
```

```r
# Qestion 2
ggplot(iris, aes(Petal.Length, Petal.Width, color=Species)) +
  geom_point()
```

```r
# Qestion 3
ggplot(iris, aes(Petal.Length, Petal.Width)) +
  geom_point() +
  geom_smooth(method = 'lm')
```

```r
# Qestion 3
ggplot(iris, aes(Petal.Length, Petal.Width)) +
  geom_point(aes(color=Species)) +
  geom_smooth(method = 'lm')

ggplot(iris, aes(Petal.Length, Petal.Width)) +
  geom_point() +
  facet_wrap(~Species) +
  geom_smooth(method = 'lm')
```

3 data.table operations

Load `iris` data, which comes with `ggplot2`. Compute step by step the standard deviation $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$ of the petal length by species.

- Copy the iris data.table into a new one, in order not to mess with it. Use `copy()`.
- Then, add columns with
    - petal length mean per species: $\overline{x}$
    - petal length - petal length mean, squared: $(x_i - \overline{x})^2$
    - sum of this squared difference by species
    - number of occurrences N per species
    - s computed as in the formula. Use `sqrt()`.
- Add another column using the `sd()` by species and compare your results with it using `identical()`.

```
library(data.table)
iris_dt <- as.data.table(iris)
iris2 <- data.table(copy(iris_dt))
iris2[, mean_PL := mean(Petal.Length), by = Species]
iris2[, dif_squared := (Petal.Length - mean_PL)^2]
iris2[, sum_squares := sum(dif_squared), by = Species]
iris2[, N := .N, by = Species]
iris2[, sd_mine := sqrt(1/(N-1)*sum_squares)]
iris2[, sd := sd(Petal.Length), by = Species]
iris2[, identical(sd_mine, sd)]  # Or identical(iris2$sd_mine, iris2$sd)
## [1] TRUE
```

4 Reading and cleaning up data

Load pokemon data with `readRDS`. Open the data.tables to check the information inside them.

```
cat(getwd())
## /data/nasif12/home_if12/theodora/Projects/CEDOSIA-dataviz/lectures-SS20/exercises/exercise-2
poke_dt <- readRDS('extdata/tidy_pokemon_poke_dt.RDS')
evolution_dt <- readRDS('extdata/tidy_pokemon_evolution_dt.RDS')
```

1. Add a column to the poke_dt with the evolutions of each pokemon and the level it requires to evolve. *Hint*: merge() or join()

```
# Using merge
poke_merge <- merge(poke_dt, evolution_dt[,.(Name, Evolution, Level)], by="Name")  # Only pokemon with evolut
poke_merge <- merge(poke_dt, evolution_dt[,.(Name, Evolution, Level)], by="Name", all.x = T)
```

2. Sort the table with Attack scores. Which pokemon has the highest Attack?

```
# Just sort the table
poke_merge[order(-Attack)] %>% head
##           Name Number     Type Total  HP Attack Defense Special_Attack
## 1: Dragonite    149   DRAGON   600  91    134      95            100
## 2: Dragonite    149   FLYING   600  91    134      95            100
## 3:   Flareon    136     FIRE   525  65    130      60             95
## 4:   Kingler     99    WATER   475  55    130     115             50
## 5:   Machamp     68 FIGHTING   505  90    130      80             65
## 6:    Rhydon    112   GROUND   485 105    130     120             45
##    Special_Defense Speed Evolution Level
## 1:             100    80      <NA>    NA
## 2:             100    80      <NA>    NA
## 3:             110    65      <NA>    NA
## 4:              50    75      <NA>    NA
## 5:              85    55      <NA>    NA
## 6:              45    40 Rhyperior    NA
setorder(poke_merge, -Attack) %>% head
##           Name Number     Type Total  HP Attack Defense Special_Attack
## 1: Dragonite    149   DRAGON   600  91    134      95            100
## 2: Dragonite    149   FLYING   600  91    134      95            100
## 3:   Flareon    136     FIRE   525  65    130      60             95
## 4:   Kingler     99    WATER   475  55    130     115             50
## 5:   Machamp     68 FIGHTING   505  90    130      80             65
## 6:    Rhydon    112   GROUND   485 105    130     120             45
##    Special_Defense Speed Evolution Level
## 1:             100    80      <NA>    NA
## 2:             100    80      <NA>    NA
## 3:             110    65      <NA>    NA
## 4:              50    75      <NA>    NA
## 5:              85    55      <NA>    NA
## 6:              45    40 Rhyperior    NA
```

5 Understanding a messy dataset

The following file describes the number of times a person bought a product "a" and "b"

```
messy_file <- file.path('extdata', 'example_product_data.csv')
messy_dt <- fread(messy_file)
messy_dt
##            name producta productb
## 1:     John Doe       NA       12
## 2:    Marry Doe        3        1
## 3: John Johnson        5        1
```

Why is this data-set messy? Which columns should a tidy version of this table have?

```
## Vales are stored as column names.
## Tidy data columns: name, product, n
```

6 Fixing a messy dataset

Read the weather dataset `weather.txt`. It contains the minimal and maximal temperature on a certain city (id) over different dates (year, month, d1-d31). Why is this dataset messy? How would a tidy version of it look like? Create its tidy version.

```
messy_dt <- fread("extdata/weather.txt")
messy_dt %>% head
##             id year month element d1  d2  d3 d4  d5 d6 d7 d8 d9 d10 d11 d12 d13
## 1: MX000017004 2010     1    TMAX NA  NA  NA NA  NA NA NA NA NA  NA  NA  NA  NA
## 2: MX000017004 2010     1    TMIN NA  NA  NA NA  NA NA NA NA NA  NA  NA  NA  NA
## 3: MX000017004 2010     2    TMAX NA 273 241 NA  NA NA NA NA NA  NA 297  NA  NA
## 4: MX000017004 2010     2    TMIN NA 144 144 NA  NA NA NA NA NA  NA 134  NA  NA
## 5: MX000017004 2010     3    TMAX NA  NA  NA NA 321 NA NA NA NA 345  NA  NA  NA
## 6: MX000017004 2010     3    TMIN NA  NA  NA NA 142 NA NA NA NA 168  NA  NA  NA
##    d14 d15 d16 d17 d18 d19 d20 d21 d22 d23 d24 d25 d26 d27 d28 d29 d30 d31
## 1:  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA 278  NA
## 2:  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA 145  NA
## 3:  NA  NA  NA  NA  NA  NA  NA  NA  NA 299  NA  NA  NA  NA  NA  NA  NA  NA
## 4:  NA  NA  NA  NA  NA  NA  NA  NA  NA 107  NA  NA  NA  NA  NA  NA  NA  NA
## 5:  NA  NA 311  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
## 6:  NA  NA 176  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
dim(messy_dt)
## [1] 22 35
```

```
## Why is it messy?
## 1. Variables are stored as columns (days)
## 2. A single entity is scattered across many cells (date)
## 3. Element column is not a variable.
##
## Tidy version: id, date, tmin, tmax
```

```
tidy_dt <- messy_dt %>%
  melt(id.vars=c('id', 'year', "month", "element"), na.rm=TRUE) %>%
  .[, variable := gsub('d', '', variable)] %>%
  unite(col=date, year, month, variable, sep='-') %>%
  dcast(... ~ element) %>%
  .[, date := as.Date(date)]


## wide -> long
dt <- melt(messy_dt, id.vars = c("id", "year", "month", "element"), variable.name = "day")
# you can ignore the warning message
dt[, day := as.integer(gsub("d", "", day))]

# Join all date related columns into one. Use unite or paste
# 1. Using unite():
dt <- unite(dt, "date", c("year", "month", "day"), sep = "-", remove = TRUE)

# 2. Using paste():
# dt[, date := paste(year, month, day, sep = "-")] # convert to date
```

```r
# dt[, c("year", "month", "day") := NULL] # remove reduntant columns


dt <- dcast(dt, ... ~ element, value.var = "value") # long -> wide

dt <- dt[!(is.na(TMAX) & is.na(TMIN))] # remove entries with both NA values,
                                       # na.omit(dt) would also do the job

head(dt)
##             id       date TMAX TMIN
## 1: MX000017004  2010-1-30  278  145
## 2: MX000017004 2010-10-14  295  130
## 3: MX000017004 2010-10-15  287  105
## 4: MX000017004 2010-10-28  312  150
## 5: MX000017004  2010-10-5  270  140
## 6: MX000017004  2010-10-7  281  129

dim(dt)
## [1] 33  4
```

```r
# An alternative tidy code version
tidy_dt <- messy_dt %>%
  melt(id.vars=c('id', 'year', 'month', 'element'), na.rm=TRUE) %>%
  .[, variable := gsub('d', '', variable)] %>%
  unite(date, year, month, variable, sep='-') %>%
  dcast(... ~ element) %>%
  .[, date := as.Date(date)]
```