Exercise sheet: Day 2

Vangelis Theodorakis, Fatemeh Behjati, Julien Gagneur, Marcel Schulz

14 October, 2020

Contents

1 Setup . . . 2

2 Introduction to ggplot . . . 2

3 Histograms . . . 4

4 Boxplots . . . 5

5 Scatterplot . . . 7

6 Understanding a messy dataset . . . 9

7 Fixing a messy dataset . . . 10

1 Setup

```
library(ggplot2)
library(data.table)
library(magrittr)   # Needed for %>% operator
library(tidyr)
library(readxl)
library(dplyr)
```

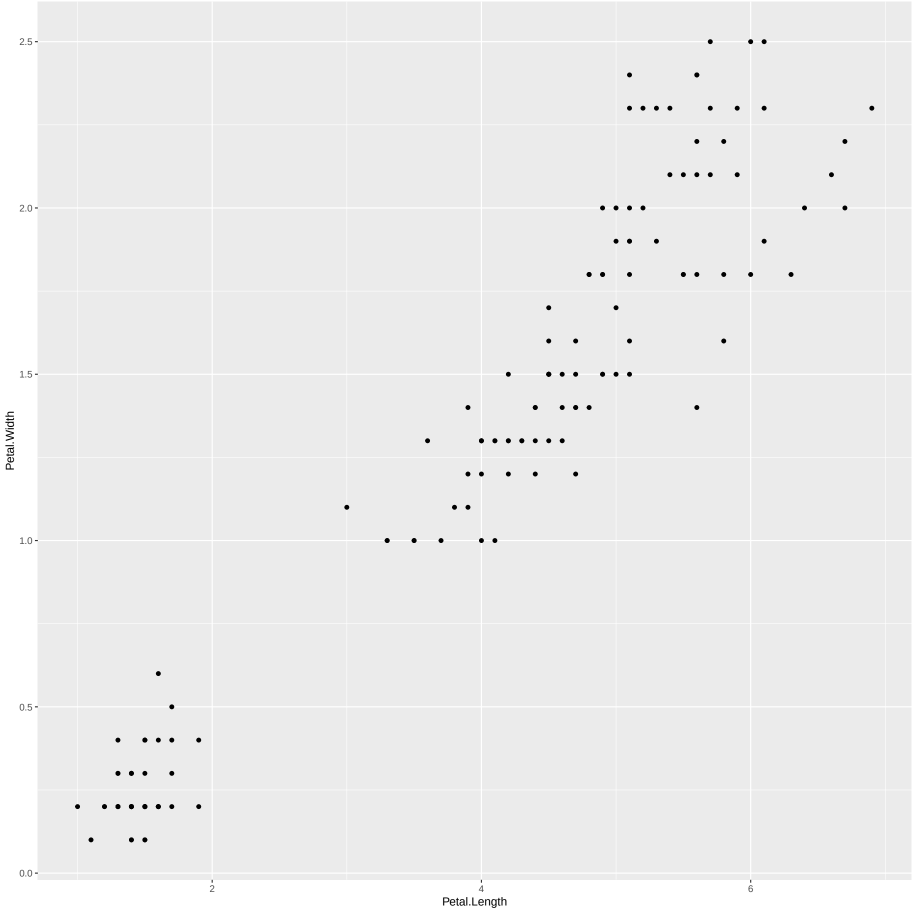
2 Introduction to ggplot

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

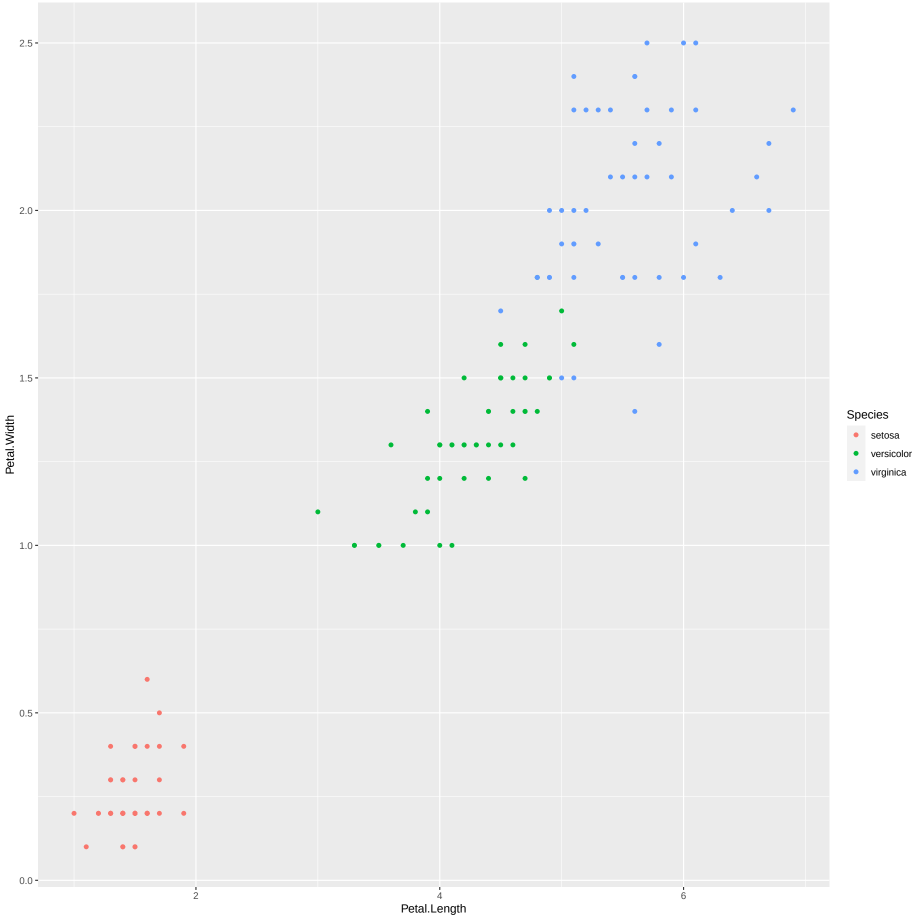
1) Are there any relationships/correlations between petal length and width? How would you show it?

2) Do petal lengths and widths correlate in every species?

```
## Answer: 1)
data(iris)
ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width)) +
  geom_point()
```



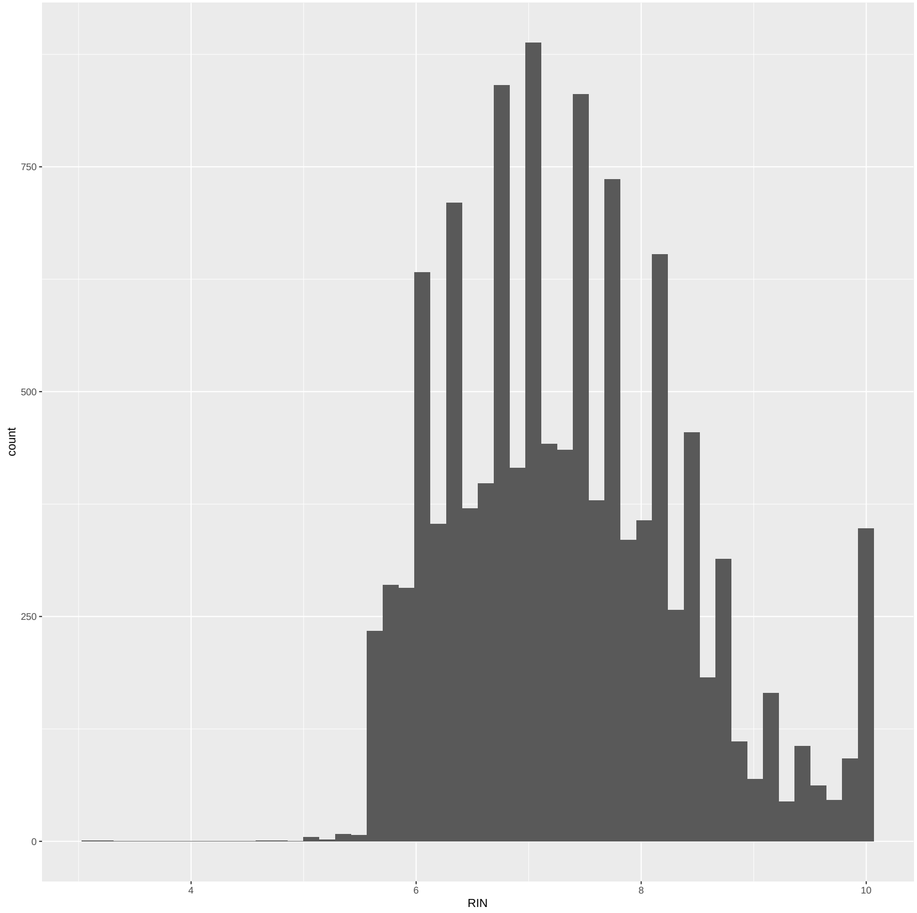
```
## Answer: 2)
ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) +
  geom_point()
```



3 Histograms

Get the *gtex-annotation.csv* data and do a histogram of the RIN (RNA integrity number) column using 10, 20, 50, 100 bins to see how this affects the visualisation.

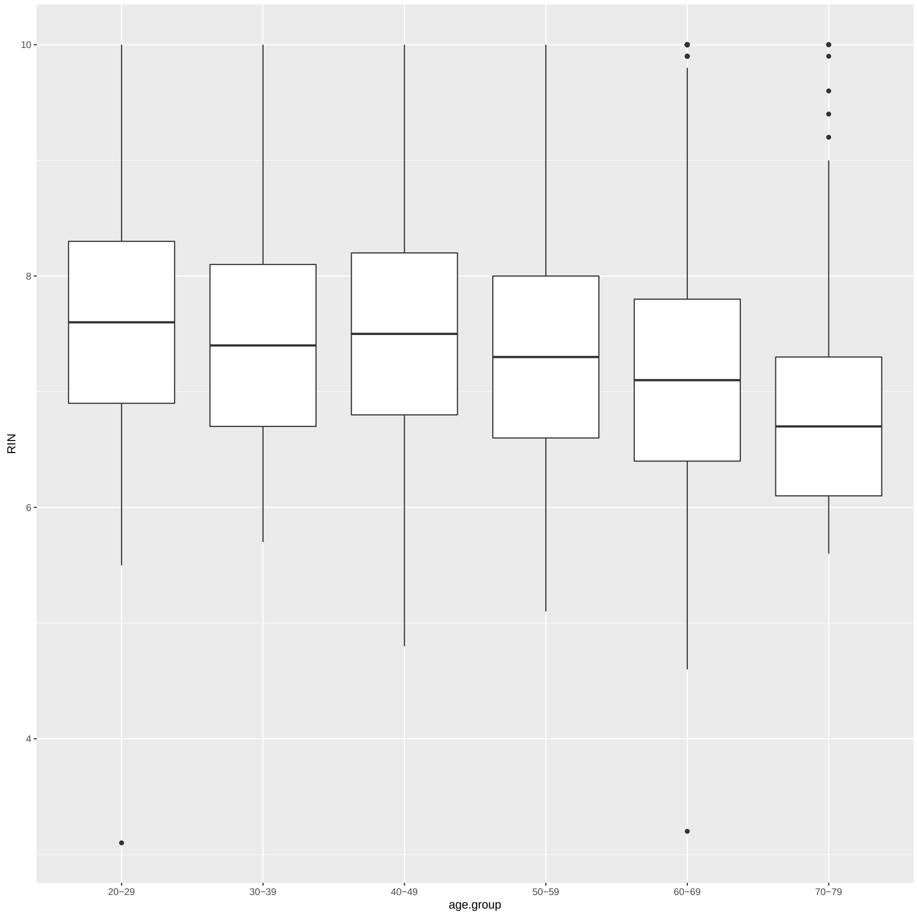
```
gtex.annotation <- fread("~/Projects/ncRNA-workshop/extdata/gtex-dummy-dataset.csv")
ggplot(gtex.annotation, aes(RIN)) + geom_histogram(bins = 50)
```



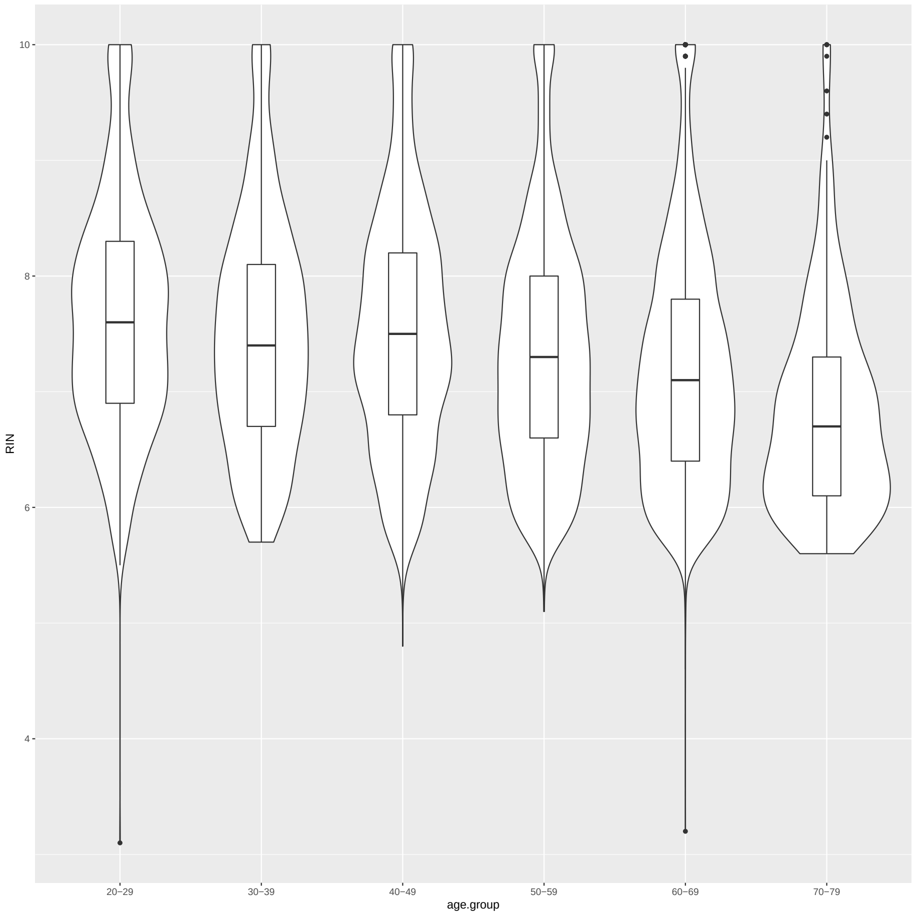
4 Boxplots

Get the *gtex-annotation.csv* data and do some boxplots of the RIN (RNA integrity number) column against the age groups. Do you see something interesting? Do the same using violin plots. Now try to combine the violin and the boxplot into one plot (use `width = 0.2`).

```
ggplot(gtex.annotation, aes(age.group, RIN)) + geom_boxplot()
```



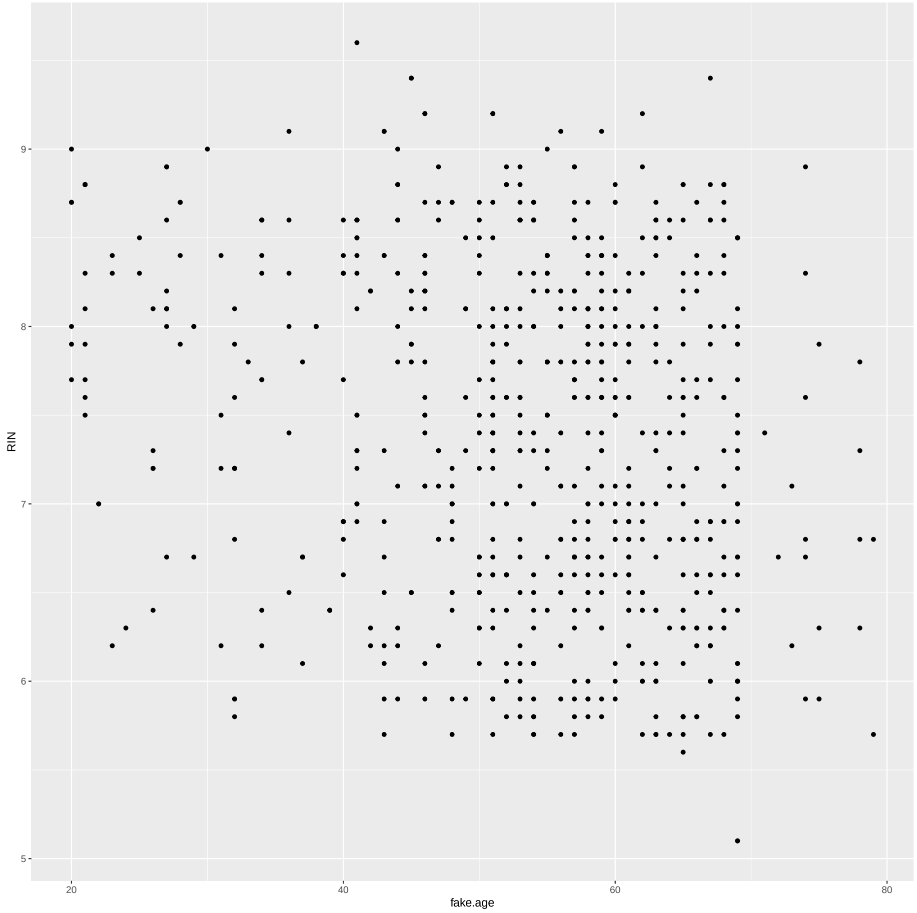
```
ggplot(gtex.annotation, aes(age.group, RIN)) + geom_violin() + geom_boxplot(width=0.2)
```



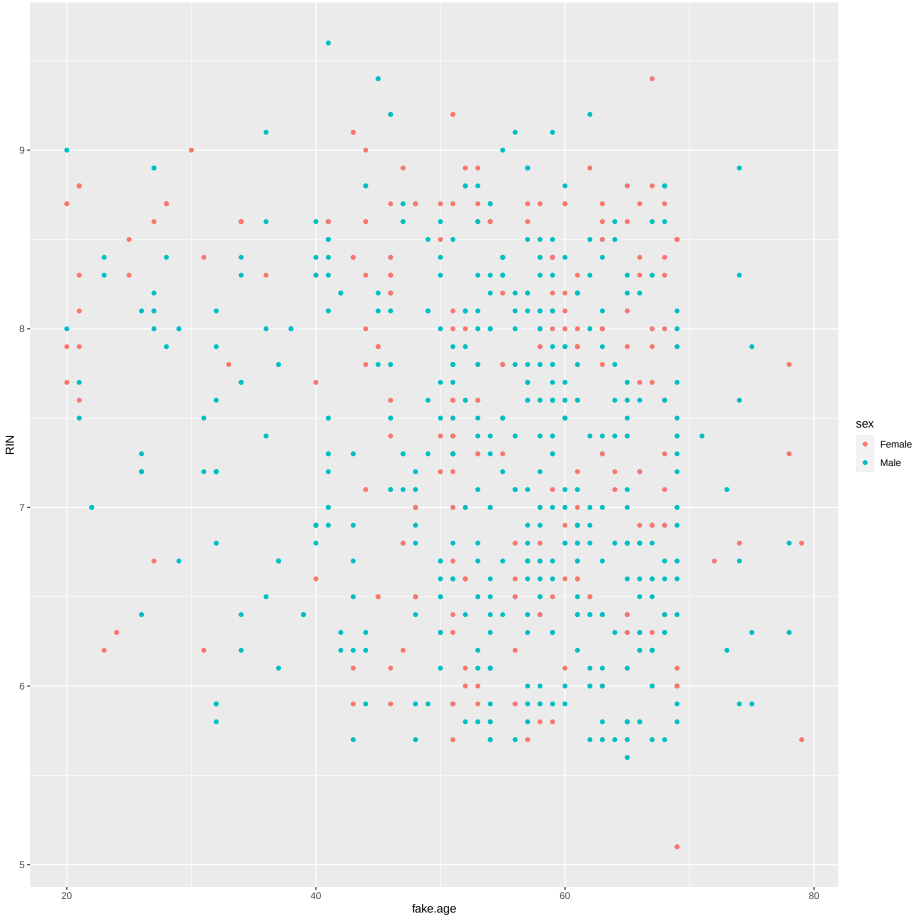
5 Scatterplot

Make a scatterplot between the fake.age and the RIN for heart. Do you see any associasion between fake.age and RIN? Now color the points by `sex` so that you have the labels `Male` and `Female` on the legend. Do you see any associasion between fake.age and RIN when it is controled for sex?

```
sex <- gtex.annotation$sex

sex[sex == 2 & !is.na(sex)] <- "Female"

sex[sex == 1 & !is.na(sex)] <- "Male"

gtex.annotation$sex <- sex

ggplot(gtex.annotation[tissue == "Heart"], aes(fake.age, RIN)) + geom_point()
```



```
ggplot(gtex.annotation[tissue == "Heart"], aes(fake.age, RIN, color=sex)) + geom_point()
```



6 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?

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

7 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?

## Answer:
## 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
```

```
## Fix a messy data

### First melt the table: wide -> long
dt <- melt(data = messy_dt,
           id.vars = c("id", "year", "month", "element"),
           variable.name = "day")

# You can ignore the warning message
# measure.vars is missing. When missing, measure.vars will become all columns outside id.vars.
# value.name: name for the molten data values column(s). The default name is 'value'.


### Then make the column day into integer
dt[, day := as.integer(gsub(pattern = "d", replacement = "", x = day))]


### Join all date related columns into one. Use unite or paste
```

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


### Dcast the table: long -> wide
dt <- dcast(data = dt, formula = ... ~ element, value.var = "value")


### Remove entries with both NA values,
tidy_dt <- dt[!(is.na(TMAX) & is.na(TMIN))]


## na.omit(dt) would also do the job
# tidy_dt <- na.omit(dt)

head(tidy_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(tidy_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)]
```