Reproducible data treatment with R – Exam

01/12/2020

Global setup (1 point bonus)

1. Unzip the `Exam_2020.zip` file somewhere on the computer, and rename the Rmd file as *name_firstname.Rmd*. You will work in this Rmd file and send it to me by email at the end, at **colin.bousige@univ-lyon1.fr**. Change the name of the author in the YAML header to your own name, but don't touch anything else. *You will get a 1 point bonus if the file you sent me can be knitted without error*, even if you didn't finish the whole exercise: so, comment out any non-working code.
2. All plots should be performed using `ggplot2`.
3. Load the libraries `tidyverse` and `broom` and set the global `ggplot2` theme to `theme_bw()`.

```r
# this is the ONLY package you need:
library(tidyverse)
theme_set(theme_bw())
```

Exercise

Part 1: Data wrangling (∼ 1 hour - 12 points)

Here we will recursively treat some x-ray diffraction (XRD) data obtained during a high-pressure experiment at the ESRF (the synchrotron in Grenoble). In this experiment, we measured the XRD patterns as a function of the pressure for two samples of the same rock, nicknamed *PY02*. PY02 is a pyrobitumen, *i.e.* some coal-like amorphous and porous carbon structure. The goal was to measure the *bulk modulus* κ of this rock, defined as:

$$\frac{1}{\kappa} = -\frac{1}{V}\frac{\partial V}{\partial P}.$$

The bulk modulus characterizes the volume reduction of a sample as a function of the applied pressure, it's unit being a pressure unit (usually in the GPa range).

The bulk modulus is usually obtained during a high pressure experiment thanks to the Murnaghan equation of state:

$$\frac{V}{V_0} = \left(1 + P\frac{\kappa'}{\kappa}\right)^{-1/\kappa'}$$

where V is the volume of the unit cell, V_0 the volume at ambient pressure, and κ' is the pressure derivative of κ. In first approximation, we will consider that κ' is constant and that $\kappa' \simeq 4$.

1. Define the `Murnaghan(P, K, Kp)` function that, given the pressure P, the bulk modulus K and its derivative Kp (defaulting to $Kp = 4$), returns the relative volume $\frac{V}{V_0}$.

```r
Murnaghan <- function(P, K, Kp=4){
    (1 + P*Kp/K)^(-1/Kp)
}
```

2. Find in the `Data` folder all the files containing the experimental diffraction patterns. They are under the form `PY02_Pxx.dat` or `PY02bis_Pxx.dat`. Store this list of files in a vector called `flist`.

```r
flist <- list.files(path="Data", pattern = "_P")
```

3. Read and store the two files containing the pressures of each run, `PY02_pressures.dat` and `PY02bis_pressures.dat`. Combine these two tables into a single tidy one called `Pressures`, containing three columns `run`, `P.kbar` and `sample` (`sample` being either `PY02` or `PY02bis`).

```r
PY02_P            <- read_table2("Data/PY02_pressures.dat")
PY02bis_P         <- read_table2("Data/PY02bis_pressures.dat")
PY02bis_P$sample <- "PY02bis"
PY02_P$sample     <- "PY02"
Pressures         <- bind_rows(PY02_P, PY02bis_P)
```

4. Define the `norm01(x)` function that given a vector, returns this vector normalized to [0,1]

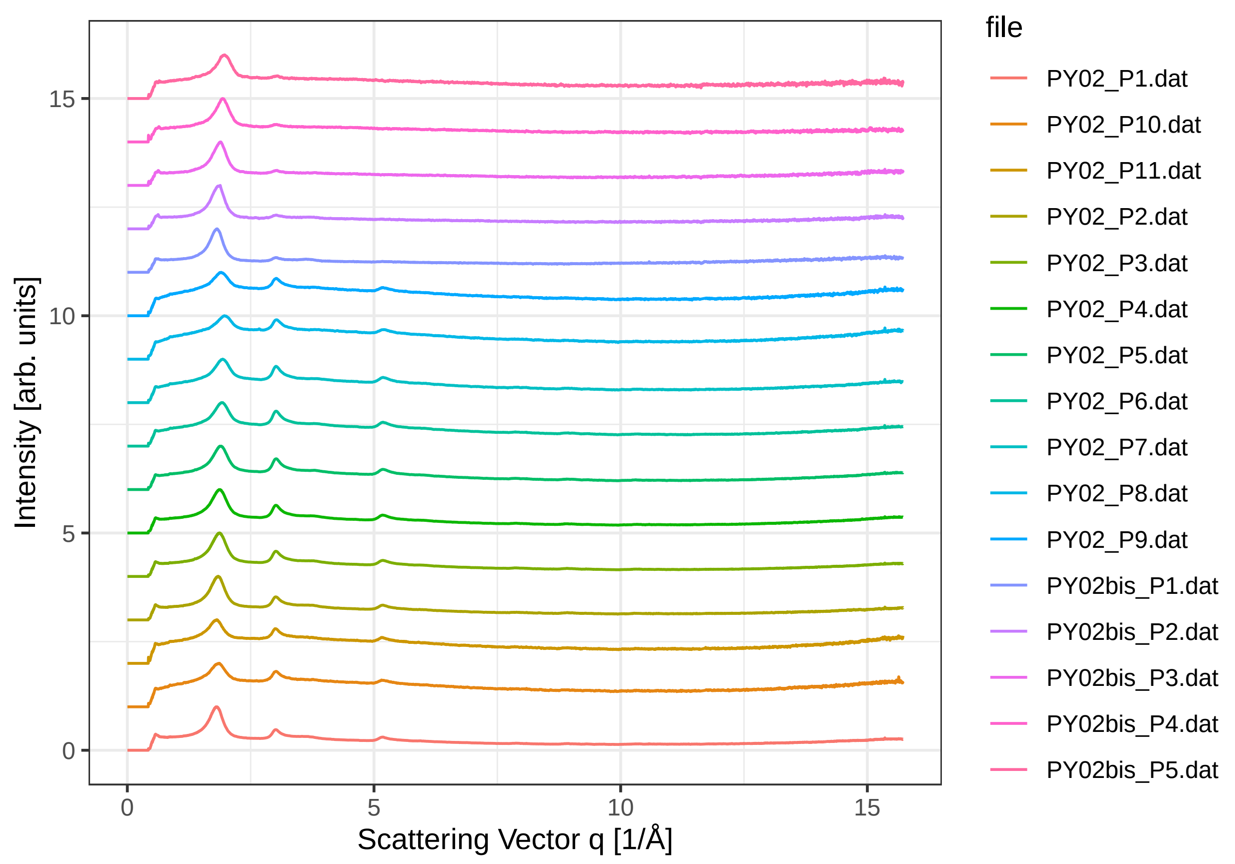
```r
norm01 <- function(x) {
    (x-min(x))/(max(x)-min(x))
}
```

5. Now let's read all the data files in `flist` and store the result in a tidy `tibble` called `data` with three columns, `file`, `q` and `int` (the data files containing the diffracted intensity as a function of the scattering vector *q*). Either use a `for loop` or the `tidyverse`-friendly version using `purrr::map()` (+1 bonus for this version). Add the `int_n` column containing the normalized intensity **for each file** (*i.e.* each file's data should be normalized to [0,1], do not normalize the whole `int` column directly).

```r
# "for loop" solution
data <- tibble()
for(file in flist){
    d <- read_table2(file.path("Data", file), col_names=c("q","int")) %>%
            mutate(int_n = norm01(int),
                   file  = file)
    data <- bind_rows(data, d)
}
# tidyverse solution
data <- tibble(file=flist) %>%
    mutate(data=map(file, ~ read_table2(file.path("Data", .), col_names=c("q","int")))) %>%
    unnest(data) %>%
    group_by(file) %>%
    mutate(int_n=norm01(int))
```

6. Plot all the normalized diffractograms on top of each other, with lines of a different color for each file, and define nice axis labels, such as *Scattering Vector q [1/Å]* and *Intensity [arb. units]*.

```r
data %>%
    ggplot(aes(x = q,
               y = int_n + as.numeric(factor(file))-1,
               color = file))+
        geom_line()+
        labs(x = "Scattering Vector q [1/Å]",
             y = "Intensity [arb. units]")
```



7. We are interested in the position of the first peak around $q = 1.8$ Å^{-1}. Starting from `data`, create the `pospeak` tibble that stores the position `Q` of the maximum of each diffractogram. You will do so by recursively (use the pipe `%>%`):
   - Make sure to filter data for scattering vectors below 2.5 Å^{-1}.
   - For each file, summarize the data by creating the columns
     - `Q`, storing the position of the maximum in intensity
     - `dQ`, storing the error on the value of `Q`. We will estimate it as 0.5% of the `Q` values.
   - Using the function `separate(column_name, c("sample","run","extension"))`, add the columns `run` and `sample` that transforms the column `file` to the run number and sample name. Make sure the column `run` contains a number (so, remove the string "P"), and get rid of the `extension` column.
   - Finally, order the tibble by ascending run for each sample

```
pospeak <- data %>%
    filter(q<2.5) %>%
    group_by(file) %>%
    summarise(Q  = q[which.max(int)],
              dQ = Q*0.005) %>%
    separate(file, c("sample","run","ext")) %>%
    select(-ext) %>%
    mutate(run = as.numeric(gsub("P","",run))) %>%
    arrange(sample, run)
```

8. Now join this table together with the `Pressures` one in order to get the sample names and pressures in this table. Finally, add the new column `P` containing the pressure in GPa (1 GPa = 10 kbar).

```r
pospeak <- pospeak %>%
    inner_join(Pressures) %>%
    mutate(P = P.kbar/10)
```

In case you didn't manage to get there, I provide in the exam .Rmd file the expected `pospeak` table so that you can continue with the exercise.

Part 2: Fitting ($\sim$ 1 hour - 6 points)

9. We now need to estimate the volume variation and the error on its measurement. Normally, we would need to get the crystal lattice volume, but as mentioned earlier, the PY02 sample is an amorphous rock, and thus it has no crystalline order. In first approximation, we can consider that the volume reduction is linked to the inter-atomic distance reduction through:

$$\frac{V}{V_0} = \left(\frac{r}{r_0}\right)^3,$$

where r is the most representative inter-atomic distance (*i.e.* the one whose Fourier transform will have the highest intensity), the scattering vector q being linked to the distance r by $q = 2\pi/r$. In our case, $r \simeq 2\pi/1.8 \simeq 3.5$ Å is the distance between two graphitic-like planes. Using this, add to the `pospeak` tibble the columns `VV0` and `dVV0` containing, respectively, the volume reduction and an estimation of its measurement error.
    - Make sure that the `V0` part refers to the value at ambient pressure (*i.e.* $P = 0$) *of the corresponding sample*.
    - Make sure you propagate the errors in a proper way.

```r
pospeak <- pospeak %>%
    group_by(sample) %>%
    mutate(VV0 = (Q[P==0]/Q)^3,
          dVV0 = 3*VV0*sqrt((dQ[P==0]/Q[P==0])^2 + (dQ/Q)^2))
```

10. Fit the Murnaghan equation of state to the experimental data. Make sure to add an amplitude parameter A (that should be close to 1) to allow for more leeway on the fit. We fix the derivative value at $Kp = 4$. The bulk modulus K should be of the order of a few GPa, so work with pressures in GPa.

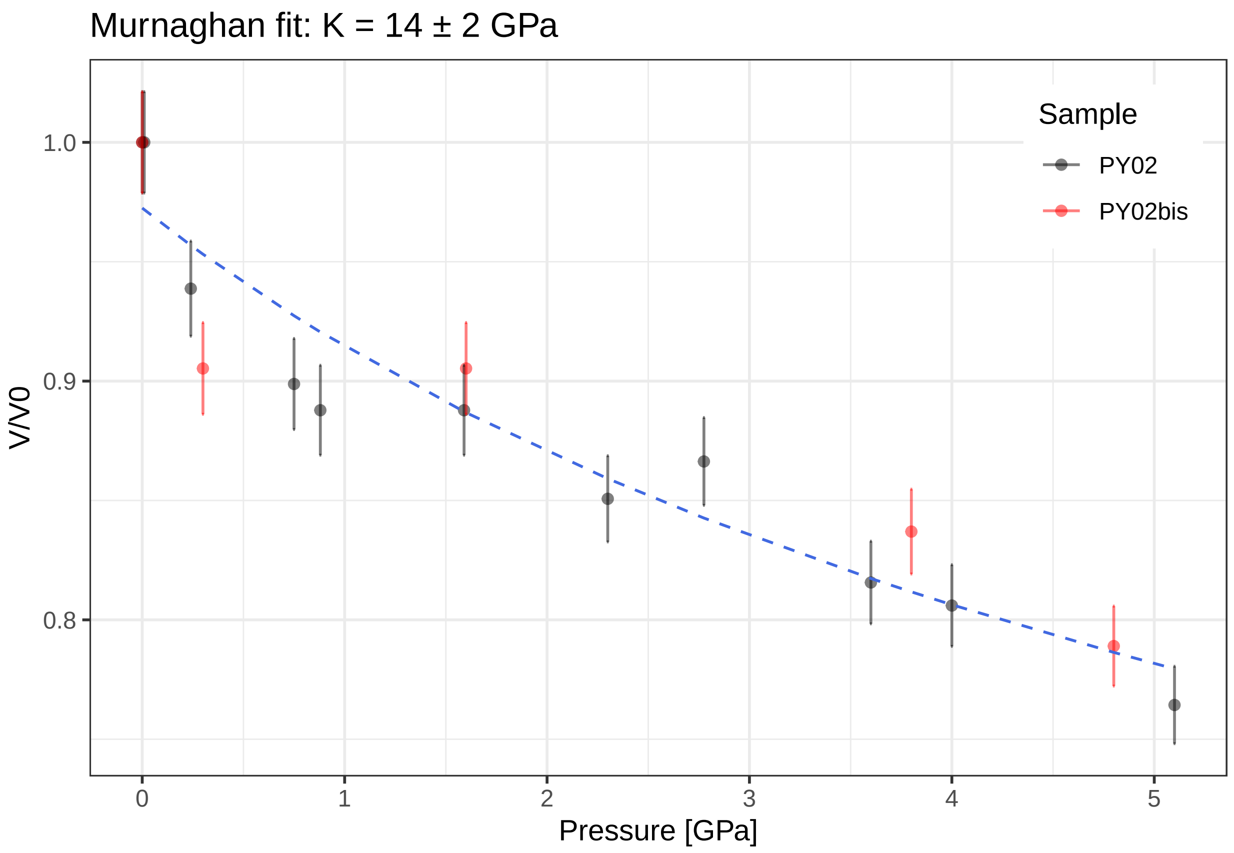
```r
fit <- nls(data = pospeak,
           VV0 ~ A*Murnaghan(P, K),
           start = list(A=1, K=1))
# Include errors in V/V0 for the fit with the "weights" parameter:
# fit <- nls(data = pospeak,
#             VV0 ~ A*Murnaghan(P, K),
#             start = list(A=1, K=1),
#             weights = 1/dVV0^2)
```

11. Store the resulting values of K and it's fitting standard error dK in two variables. Round the values to the appropriate floating number. You may use the `broom::tidy()` function.

```r
K  <- round(broom::tidy(fit) %>% filter(term=="K") %>% .$estimate, 0)
dK <- round(broom::tidy(fit) %>% filter(term=="K") %>% .$std.error, 0)
```

12. Finally, make the plot of the experimental points (a color per sample) and the fit. Make it look like so:

```r
pospeak %>%
    ggplot(aes(x=P, y=VV0))+
        geom_point(alpha=0.5, aes(color=sample))+
        geom_errorbar(alpha=0.5, aes(color=sample, ymin=VV0-dVV0, ymax=VV0+dVV0))+
        geom_line(data=augment(fit), aes(x=P, y=.fitted), color="royalblue", lty=2)+
        labs(x="Pressure [GPa]",
             y="V/V0",
             color="Sample")+
        scale_color_manual(values = c("black","red"))+
        ggtitle(paste("Murnaghan fit: K =", K, "±", dK, "GPa"))+
        theme(legend.position = c(.9,.85))
```


Murnaghan fit: K = 14 ± 2 GPa
Sample
PY02
PY02bis
V/V0
Pressure [GPa]