The structure and interaction of silica in plant cell walls: Data analysis report

Felipe Beltran

6/16/2021

Introduction

In this document we will save all data manipulation and analysis related to the infrared spectra from the project: **The structure and interaction of silica in plant cell walls**

Loading data into `R`

First, we create an object called `names`, which is a `list`, in which we will save the names of the files with the `.CSV` extension contained in this folder.

```r
names <- list.files(pattern = '.CSV')
```

Then, we use the function `lapply` to apply the function `read.csv` to each element of the list `names` and then save each of the `data.frames` from the files in to R.

```r
spectra.list <- lapply(names, read.csv, header = F)
```

Now, se can save the frequencies of incident light as a vector called `wavenumbers`. The code `spectra.list[[1]][1]` selects the first element of the list `spectra.list` using the index operator for lists `[[1]]`, which is a `data.frame`. Then, we select the first column of that `data.frame` using the index operator `[1]`:

```r
wavenumbers <- unlist(spectra.list[[1]][1])
```

The object `wavenumbers` has 1 row and 1869 columns, each column is one of the frequencies used by the spectrometer in the experiment.

As we did previously, we can use again the function `lapply` to apply the index operator `[` and select ONLY the second column of each `data.frame` that has inside the absorbance at each frequency.

```r
spectra.list2 <- lapply(spectra.list, '[', 2)
```

In order to make data easier to manipulate, we can transform the list of columns called `spectra.list2` into a `data.frame` as follows:

```
spectra.df <- as.data.frame(t(as.data.frame(spectra.list2)))
```

the object `spectra.df` has 318 rows, i.e. 318 spectra from different samples and 1869 columns, or infrared frequencies.

Since the function `t()` transforms `data.frame` class objects into `matrix` class objects, we have to assign again to `spectra.df` (a matrix) the corresponding column and row names:

```
rownames(spectra.df) <- names
colnames(spectra.df) <- wavenumbers
# gsub('.{6}$', '', names)
```

Tidying up the `data.frame`

Since the names of the sample right now have the following structure:

```
head(names)
```

```
## [1] "125-1.CSV" "125-2.CSV" "125-3.CSV" "126-1.CSV" "126-2.CSV" "126-3.CSV"
```

We can get rid of the unnecessary characters in the names of the samples, such as the extension and other symbols such as `"` or `.`. For that we can use the function `gsub()` and `regex` or regular expressions in order to replace these symbols for nothing, or `''`.

The regular expression for this replacement is built as follows:

- `.` matches any element that complies with the requirement, or applies the query to every element
- `{4}` replaces characters by `''` exactly 4 times
- `$` matches the end of each name

That is described before allows us to tell `R` that we want to replace the last 4 characters in each name by nothing, or `''`.

```
names2 <- gsub('.{4}$', '', rownames(spectra.df))
head(names2)
```

```
## [1] "125-1" "125-2" "125-3" "126-1" "126-2" "126-3"
```

There are some samples that have longer names, such as:

```
head(names2[292:318])
```

```
## [1] "P1-1_Thu May 09 07-26-09 2019 (GMT+02-00)"
## [2] "P1-2_Thu May 09 07-26-09 2019 (GMT+02-00)"
## [3] "P1-3_Thu May 09 07-26-09 2019 (GMT+02-00)"
## [4] "P2-1_Thu May 09 07-26-09 2019 (GMT+02-00)"
## [5] "P2-2_Thu May 09 07-26-09 2019 (GMT+02-00)"
## [6] "P2-3_Thu May 09 07-26-09 2019 (GMT+02-00)"
```

We can also save just the information that is useful for displaying in tables and plots:

```
names2[292:318] <- gsub('.{41}$', '', rownames(spectra.df)[292:318])
head(names2[292:318])
```

```
## [1] "P1-1" "P1-2" "P1-3" "P2-1" "P2-2" "P2-3"
```

Finally, we assign the curated names to our data frame, `spectra.df`

```
rownames(spectra.df) <- names2
```

Plotting intial data

Since we have joined several spectra from different samples, we can try to visualize what we are dealing with.

Knowing that we have already the names of the samples, it would be handy if we used a factor that helped us to differentiate replicas of experiments:

```
cols <- factor(gsub('.{2}$', '', names2))
head(cols)
```

```
## [1] 125 125 125 126 126 126
## 106 Levels: 125 126 127 128 129 130 132 133 134 135 136 137 138 139 140 ... P9
```

If we use the factor `cols`, we can plot three replicas of one sample using the same color

Now, we can use a loop to plot each one of the rows of our `data.frame`:

```
for(i in  1:length(rownames(spectra.df))){

  plot(wavenumbers,
    spectra.df[i,],
    axes = F,
    xlab = '',
    ylab = '',
    xlim = c(4000, 400),
    ylim= c(0,0.2),
    type = 'l',
    col =cols[i]

  )
  par(new = T)
}

box()
axis(1)
axis(2)
title(main = '',
      xlab = expression(paste('Wave number (cm'^'-1',')')),
      ylab ='absorbance (a.u.)')
```

As we can see in Figure 1 In the range between 2500 and 2000 cm-1 variability due to experimental noise and background correction is present. In order to use multivariate methods and extract the chemical information from the dataset, we can select a region of interest (ROI) in which we take into account characteristic bands for molecules which concentration can be different between samples, such as silicon oxides, or lignin monomers.

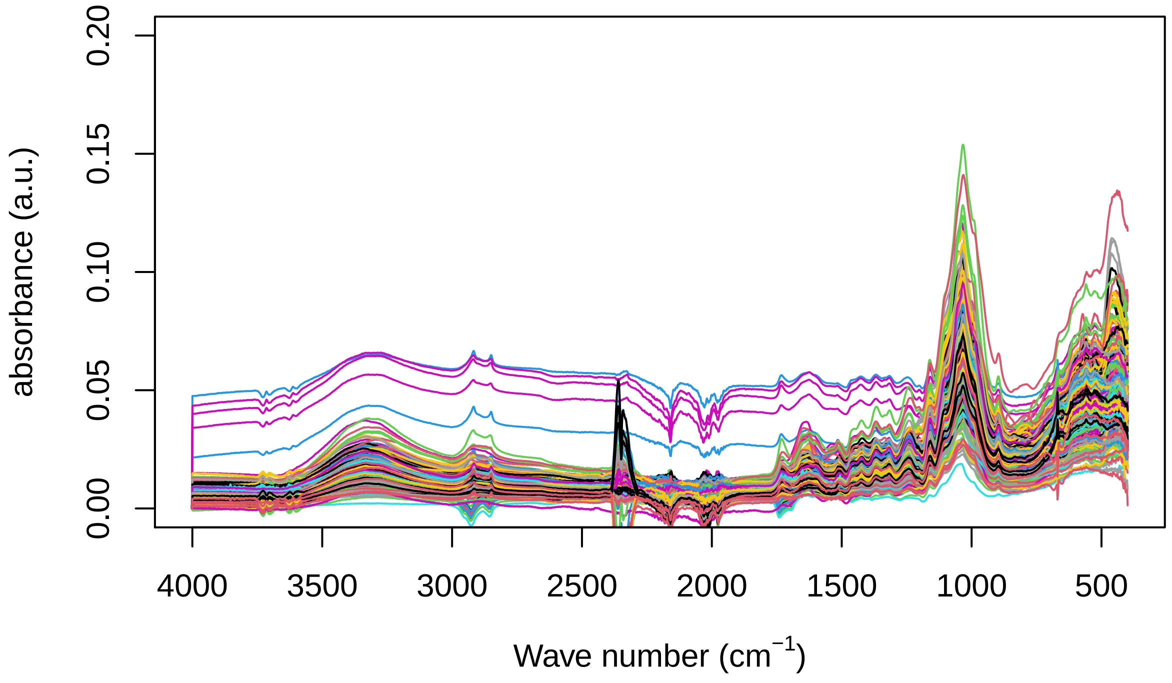


Figure 1: raw spectra of triplicated samples

range selection

Now, we can check for where is that ROI in our data frame. Since the frequencies are arranged along the columns, we can search for the columns that have column names between 1700 and 400 cm^{-1}

```
colnames(spectra.df)[c(1,676)]
```

```
## [1] "399.1989" "1700.935"
```

Once we know where the ROI is in the data frame, we can create a subset called `range1` to store the spectra of all samples within this range, and plot it:

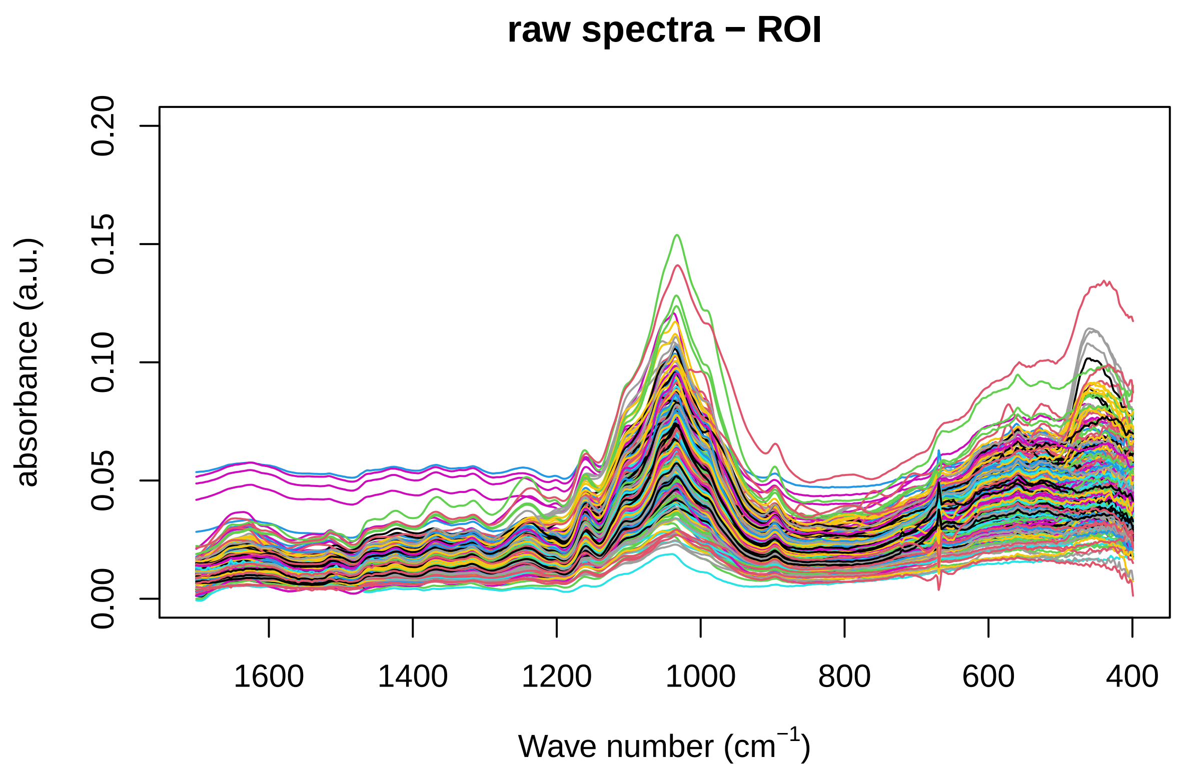
```
range1 <- spectra.df[,c(1:676)]
wavenumbers1 <- wavenumbers[c(1:676)]
```

```
for(i in  1:length(rownames(range1))){

  plot(wavenumbers1,
    range1[i,],
    axes = F,
    xlab = '',
    ylab = '',
    xlim = c(1700, 400),
    ylim= c(0,0.2),
    type = 'l',
    col =cols[i]

  )
  par(new = T)
}

box()
axis(1)
axis(2)
title(main = 'raw spectra - ROI',
      xlab = expression(paste('Wave number (cm'^'-1',')')),
      ylab ='absorbance (a.u.)')
```



mean spectra calculation

Since each sample has three replicated spectra, we can calculate the mean of each three samples as follows:

- First, we create a vector to tell R which samples to look for. Using the function `unique()` we extract the unique values without taking their repetitions:

```
search.vector <- unique(unlist(cols))
head(search.vector)
```

```
## [1] 125 126 127 128 129 130
## 106 Levels: 125 126 127 128 129 130 132 133 134 135 136 137 138 139 140 ... P9
```

Once we have the names for each sample, we can search for $\frac{318}{3} = 106$ triads of samples. Once we know where they are, we can calculate the mean for each three replicates.

First, we create a list in which we are going to save the position of each three spectra:

```
index <- list(106)
```

Now we can again use the help of regular expressions to search for every sample that matches with one single sample:

As an example:

The regular expression that we will use this time is `(?=.*<search>)` where `<search>` is replaced with the sample's name.

This will find any element for the vector `rownames(spectra.df)` that matches with the character in `search vector`. For example, if we search for the first sample, `search.vector[1]` which is sample `125`:

```
as.character(search.vector[1]) # the name of the first sample
```

```
## [1] "125"
```

The search will tell us the positions in `rownames(spectra.df)` where where we can find the character `'125'`

```
 which(
   grepl(
     paste0('(?=.*',as.character(search.vector[1]),')'),
     rownames(spectra.df),
     perl=T
     )
   )
```

```
## [1] 1 2 3
```

Then we can confirm:

```
rownames(spectra.df)[c(1, 2, 3)]
```

```
## [1] "125-1" "125-2" "125-3"
```

This process can be automated using a `for` loop in R:

```
for (i in 1:106){

  index[[i]] <- which(
    grepl(
      paste0('(?=.*',as.character(search.vector[i]),')'),
                       rownames(spectra.df),
                       perl=T
                       )
                 )
}
```

```
search.vector[2]
```

```
## [1] 126
## 106 Levels: 125 126 127 128 129 130 132 133 134 135 136 137 138 139 140 ... P9
```

```
rownames(spectra.df)[c(index[[2]])]
```

```
## [1] "126-1" "126-2" "126-3"
```

Once we know where each triplicate is, we can proceed to calculate the means:

- First, we create a matrix to save the results of the mean

```
mean <- matrix(ncol= ncol(range1),
               nrow = nrow(range1)/3)
```

Then we asign to this empty matrix, `mean`, its corresponding row and column names:

```
# mean <- as.data.frame(mean)

colnames(mean) <- colnames(range1)
rownames(mean) <- search.vector
```

```
for(j in 1:length(colnames(range1))){

for(i in 1:length(rownames(mean))){

mean[i,j] <- mean(c(range1[index[[i]][1],j],
                    range1[index[[i]][2],j],
                    range1[index[[i]][3],j]
                      ) )
}
}
mean <- as.data.frame(mean)
```