Project Plan

Don Ngo

2024-06-18

#Packages (to be adjusted)

```
library(data.table)
library(readtext)
library(stringr)
```

General Objectives

- Create a private GitHub repository.
- Extract information from crystallography files.
- Identify necessary information for cluster analyses.
- Build functions.
- Build a package.

Specific Objectives and Tasks

Task 1: Create a Private GitHub Repository

Action: Set up a private repository on GitHub.
Requirement: Each code segment added should be accompanied by an explanation.

Task 2: Bind CIF Files into One Table

Steps:

1. **Download CIF files from the provided Google Drive link.**
2. **Set the working directory:**

```
setwd("~/repos/ml-crystals/")
```

3. **List CIF files**

```
cif_list <- Sys.glob(paths = "data/*.cif")
cif_tab <- lapply(cif_list, readLines , warn =  FALSE)
```

##Task 3 and 4: Extracting Data

```
# Extracting the chemical formula
extract_chemical_formula <- function(cif_content) {
  formula_lines <- grep("_chemical_formula_sum", cif_content, value = TRUE)
  if (length(formula_lines) > 0) {
    chemical_formula <- gsub("_chemical_formula_sum", "", formula_lines)
    chemical_formula <- gsub("'", "", chemical_formula)
    chemical_formula <- trimws(chemical_formula)
```

```
    return(chemical_formula)
  } else {
    return(NA)
  }
}
# Extracting the structure type
extract_structure_type <- function(cif_content) {
  structure_lines <- grep("_chemical_name_structure_type", cif_content, value = TRUE)
  if (length(structure_lines) > 0) {
    structure_type <- gsub("_chemical_name_structure_type", "", structure_lines)
    structure_type <- gsub("'", "", structure_type)
    structure_type <- trimws(structure_type)
    return(structure_type)
  } else {
    return(NA)
  }
}
# Extracting the space group
extract_space_group_name <- function(cif_content) {
  space_group_lines <- grep("_space_group_name_H-M_alt", cif_content, value = TRUE)
  if (length(space_group_lines) > 0) {
    space_group <- gsub("_space_group_name_H-M_alt", "", space_group_lines)
    space_group <- gsub("'", "", space_group)
    space_group <- trimws(space_group)
    return(space_group)
  } else {
    return(NA)
  }
}

extract_space_group_number <- function(cif_content) {
  space_group_number_lines <- grep("_space_group_IT_number", cif_content, value = TRUE)
  if (length(space_group_number_lines) > 0) {
    space_group_number <- gsub("_space_group_IT_number", "", space_group_number_lines)
    space_group_number <- trimws(space_group_number)
    return(space_group_number)
  } else {
    return(NA)
  }
}

# Function to extract unit cell length and associated errors
extract_unit_cell_metrics <- function(cif_content) {
  extract_value_with_error <- function(line) {
    if (grepl("\\(", line)) {
      value <- sub("\\(.*\\)", "", line)
      error <- sub(".*\\((.*)\\).*", "\\1", line)
      return(list(value = as.numeric(value), error = as.numeric(error)))
    } else {
      return(list(value = as.numeric(line), error = NA))
    }
  }
```

```
  a_line <- grep("_cell_length_a", cif_content, value = TRUE)
  b_line <- grep("_cell_length_b", cif_content, value = TRUE)
  c_line <- grep("_cell_length_c", cif_content, value = TRUE)

  a <- extract_value_with_error(gsub("_cell_length_a", "", a_line))
  b <- extract_value_with_error(gsub("_cell_length_b", "", b_line))
  c <- extract_value_with_error(gsub("_cell_length_c", "", c_line))

  unit_cell_metrics <- data.frame(
    parameter = c("a", "b", "c"),
    value = c(a$value, b$value, c$value),
    error = c(a$error, b$error, c$error)
  )

  return(unit_cell_metrics)
}
```

Task 5: extracting atomic coordinates and symmetry operations and applying symmetry operations to the atomic coordinates

```
# Extracting atomic coordinates
extract_atomic_coordinates <- function(cif_content) {
  # Initialize lists to store atom properties
  labels <- vector()
  fractional_x <- vector()
  fractional_y <- vector()
  fractional_z <- vector()
  x_errors <- vector()
  y_errors <- vector()
  z_errors <- vector()

  # Find the start and end of the atom section
  atom_start <- grep("_atom_site_occupancy", cif_content)
  atom_end <- grep("^loop_|^#End", cif_content[atom_start:length(cif_content)]) + atom_start - 1

  # If atom_end has multiple matches, take the first match after atom_start
  if(length(atom_end) > 1) {
    atom_end <- atom_end[1]
  }

  # Iterate over each line containing atomic coordinates
  for (line in cif_content[(atom_start + 1):(atom_end - 1)]) {
    # Extract atom properties from the line
    properties <- strsplit(line, "\\s+")[[1]]

    # Extract and store relevant information
    labels <- c(labels, properties[1])

    # Extract fractional coordinates and errors separately
    fractional_x_val <- gsub("\\(.*\\)", "", properties[5])
    fractional_y_val <- gsub("\\(.*\\)", "", properties[6])
    fractional_z_val <- gsub("\\(.*\\)", "", properties[7])
```

```
    x_error <- ifelse(grepl("\\(", properties[5]), gsub(".*\\((.*)\\).*", "\\1", properties[5]), NA)
    y_error <- ifelse(grepl("\\(", properties[6]), gsub(".*\\((.*)\\).*", "\\1", properties[6]), NA)
    z_error <- ifelse(grepl("\\(", properties[7]), gsub(".*\\((.*)\\).*", "\\1", properties[7]), NA)

    fractional_x <- c(fractional_x, fractional_x_val)
    fractional_y <- c(fractional_y, fractional_y_val)
    fractional_z <- c(fractional_z, fractional_z_val)
    x_errors <- c(x_errors, x_error)
    y_errors <- c(y_errors, y_error)
    z_errors <- c(z_errors, z_error)
  }

  # Create a data frame to store the atomic coordinates
  atomic_coordinates <- data.frame(
    Label = labels,
    x_a = fractional_x,
    y_b = fractional_y,
    z_c = fractional_z,
    x_error = x_errors,
    y_error = y_errors,
    z_error = z_errors
  )
  return(atomic_coordinates)
}

# Extracting symmetry operations
extract_symmetry_operations <- function(cif_content) {
  operation_x <- vector()
  operation_y <- vector()
  operation_z <- vector()

  symmetry_start <- grep("_space_group_symop_operation_xyz", cif_content)
  symmetry_end <- grep("^loop_", cif_content[symmetry_start:length(cif_content)]) + symmetry_start - 1

  if (length(symmetry_end) > 1) {
    symmetry_end <- symmetry_end[1]
  }

  for (line in cif_content[(symmetry_start + 1):(symmetry_end - 1)]) {
    operations <- strsplit(gsub("'", "", line), ",\\s*")[[1]]
    operation_x <- c(operation_x, operations[1])
    operation_y <- c(operation_y, operations[2])
    operation_z <- c(operation_z, operations[3])
  }

  symmetry_operations <- data.frame(
    x = operation_x,
    y = operation_y,
    z = operation_z
  )
  return(symmetry_operations)
}
```

Task 6: Calculating distances between atoms

```
# Insert logic here
```

Extracting from all the .cifs:

```
# Initialize an empty list to store the results
results <- list()

for (cif_file in cif_list) {
  # Read the CIF file
  cif_content <- readLines(cif_file, warn = FALSE)

  # Extract the compound name from the file name
  compound_name <- tools::file_path_sans_ext(basename(cif_file))

  # Extract the relevant data
  chemical_formula <- extract_chemical_formula(cif_content)
  structure_type <- extract_structure_type(cif_content)
  space_group_name <- extract_space_group_name(cif_content)
  space_group_number <- extract_space_group_number(cif_content)
  unit_cell_metrics <- extract_unit_cell_metrics(cif_content)
  atomic_coordinates <- extract_atomic_coordinates(cif_content)
  symmetry_operations <- extract_symmetry_operations(cif_content)

  # Combine the data into a single list
  result <- list(
    compound_name = compound_name,
    chemical_formula = chemical_formula,
    structure_type = structure_type,
    space_group_name = space_group_name,
    space_group_number = space_group_number,
    unit_cell_metrics = unit_cell_metrics,
    atomic_coordinates = list(atomic_coordinates),
    symmetry_operations = list(symmetry_operations)
  )

  # Append the result to the results list
  results <- append(results, list(result))
}

# Convert the results list to a data frame
results_df <- do.call(rbind, lapply(results, function(x) {
  data.frame(
    compound_name = x$compound_name,
    chemical_formula = x$chemical_formula,
    structure_type = x$structure_type,
    space_group_name = x$space_group_name,
    space_group_number = x$space_group_number,
    unit_cell_metrics = I(list(x$unit_cell_metrics)),
    atomic_coordinates = I(x$atomic_coordinates),
    symmetry_operations = I(x$symmetry_operations)
  )
```

```
}))

# Print the results data frame
print(results_df)
```

```
##    compound_name chemical_formula   structure_type space_group_name
## 1     ICSD138351         Na4 Si24         EuGa2Ge4          C m c m
## 2     ICSD161134          Ni2 Si1    TiNiSi#MgSrSi          P b n m
## 3     ICSD182692          Mg2 Si1 Laves(cub)#MgCu2       F d -3 m S
## 4     ICSD237248          Na1 Si6         EuGa2Ge4          C m c m
## 5     ICSD240779          Si6 Sr1         EuGa2Ge4          C m c m
## 6     ICSD245295          Ba1 Si6         EuGa2Ge4          C m c m
## 7     ICSD246806         La1 Si10           LaSi10       P 63/m m c
## 8      ICSD25595          Ca2 Si1    TiNiSi#MgSrSi          P b n m
## 9     ICSD412248          Eu2 Si1    TiNiSi#MgSrSi          P n m a
## 10    ICSD416576          Eu1 Si6         EuGa2Ge4          C m c m
## 11       ICSD422          Si1 Sr2    TiNiSi#MgSrSi          P n m a
## 12     ICSD52281          Co2 Si1    TiNiSi#MgSrSi          P b n m
## 13     ICSD52698          Ba2 Si1    TiNiSi#MgSrSi          P n m a
##    space_group_number unit_cell_metrics atomic_coordinates symmetry_operations
## 1                  63      c("a", "....       c("Si1",....        c("1 -x"....
## 2                  62      c("a", "....       c("Ni1",....        c("1 -x+....
## 3                 227      c("a", "....       c("Mg1",....        c("1 z+1....
## 4                  63      c("a", "....       c("Na1",....        c("1 -x"....
## 5                  63      c("a", "....       c("Sr1",....        c("1 -x"....
## 6                  63      c("a", "....       c("Ba1",....        c("1 -x"....
## 7                 194      c("a", "....       c("La1",....        c("1 x",....
## 8                  62      c("a", "....       c("Ca1",....        c("1 -x+....
## 9                  62      c("a", "....       c("Eu1",....        c("1 x+1....
## 10                 63      c("a", "....       c("Eu1",....        c("1 -x"....
## 11                 62      c("a", "....       c("Sr1",....        c("1 x+1....
## 12                 62      c("a", "....       c("Co1",....        c("1 -x+....
## 13                 62      c("a", "....       c("Ba1",....        c("1 x+1....
```