Programming with R — A Beginners' Guide for Geoscientists

2 - Data

Tobias Stephan

09/02/2022

Contents

Types of Data

Scalars

```
a <- 1 # numeric
b <- "Word" # character
c <- TRUE # logical
```

Vectors

All elements of a vector must have the same mode (numeric, character, etc.).

```
a <- c(1, 2, 5.3, 6, -2, 4) # numeric vector
b <- c("one", "two", "three") # character vector
c <- c(TRUE, TRUE, TRUE, FALSE, TRUE, FALSE) # logical vector
```

Refer to elements of a vector using subscripts.

```r
b[2] # second element in vector b
```

```
## [1] "two"
```

```r
cbind(a, a + 1)
```

Multi-column vector

```
##          a
## [1,]  1.0  2.0
## [2,]  2.0  3.0
## [3,]  5.3  6.3
## [4,]  6.0  7.0
## [5,] -2.0 -1.0
## [6,]  4.0  5.0
```

Matrices

All columns in a matrix ($m \times n$) must have the same mode (numeric, character, etc.) and the same length. The general format is

```r
x <- c(1, 0, 0)
y <- c(0, 1, 0)
z <- c(0, 0, 1)

m <- as.matrix(
  cbind(a, b, c)
)
m
```

```
##      a     b       c
## [1,] "1"   "one"   "TRUE"
## [2,] "2"   "two"   "TRUE"
## [3,] "5.3" "three" "TRUE"
## [4,] "6"   "one"   "FALSE"
## [5,] "-2"  "two"   "TRUE"
## [6,] "4"   "three" "FALSE"
```

Identify rows, columns or elements using subscripts.

```r
m[, 3] # 3rd column of matrix
```

```
## [1] "TRUE"  "TRUE"  "TRUE"  "FALSE" "TRUE"  "FALSE"
```

```r
m[2, ] # 2nd row of matrix
```

```
##      a      b      c
##    "2"  "two" "TRUE"
```

```r
m[2, 3] # 2nd row, 3rd element
```

```
##      c
## "TRUE"
```

Data frames

A data frame is more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.).

```r
mydataframe <- data.frame(a, b, c)
names(mydataframe) <- c("column1", "column2", "column3") # header of the data frame
mydataframe
```

```
##   column1 column2 column3
## 1     1.0     one    TRUE
## 2     2.0     two    TRUE
## 3     5.3   three    TRUE
## 4     6.0     one   FALSE
## 5    -2.0     two    TRUE
## 6     4.0   three   FALSE
```

There are a variety of ways to identify the elements of a data frame:

```r
mydataframe[2:3] # columns 2 to 3 of data frame
```

```
##   column2 column3
## 1     one    TRUE
## 2     two    TRUE
## 3   three    TRUE
## 4     one   FALSE
## 5     two    TRUE
## 6   three   FALSE
```

```r
mydataframe[c("column1", "column3")] # columns ID and Age from data frame
```

```
##   column1 column3
## 1     1.0    TRUE
## 2     2.0    TRUE
## 3     5.3    TRUE
## 4     6.0   FALSE
## 5    -2.0    TRUE
## 6     4.0   FALSE
```

```r
mydataframe$column2 # variable column2 in the data frame
```

```
## [1] "one"   "two"   "three" "one"   "two"   "three"
```

```r
mydataframe$column2[2] # 2nd element of column2
```

```
## [1] "two"
```

Lists

An ordered collection of objects (components). A list allows you to gather a variety of (possibly unrelated) objects under one name.

```r
mylist <- list(name = "Jean", numbers = x, table = mydataframe)
```

Identify elements of a list using the [[]] convention.

```r
mylist[[3]] # 2nd component of the list
```

```
##   column1 column2 column3
## 1     1.0     one    TRUE
## 2     2.0     two    TRUE
## 3     5.3   three    TRUE
## 4     6.0     one   FALSE
## 5    -2.0     two    TRUE
```

```
## 6     4.0    three    FALSE
```

```r
mylist[["numbers"]] # component named mynumbers in list
```

```
## [1] 1 0 0
```

Import data

It is possible to load **every** file type into R' workspace. To import data sets, you can use the RStudio interface for Import: File > Import Dataset > From…

I recommend to import any data via the R console because if you have to repeat the import, it will save time already after 1 repeat.

Text files

The most basic import function is `read.table()` which allows to read the most 'character-separated values' (e.g. white space, tab, comma, semi-colon, … separated tables). The file extension (e.g. .txt, .csv, .dat, …) does not matter.

```r
read.table("path/to/file/table.txt", header = TRUE, sep = ";", dec = ".")
```

The following functions are identical to `read.table()` except for the defaults.

```r
read.csv("path/to/file/table.csv", header = TRUE) #  read 'comma separated value' files
read.csv2("path/to/file/table.csv", header = TRUE) #  same as read.csv() instead uses a comma as decima
```

Some data files are organized by columns that are separated by a defined width (e.g. 3 blank spaces, TAB, …). In this case, you can use `read.delim()`

```r
read.delim("path/to/file/table.dat", header = TRUE)
```

Excel

Excel files can be imported by the function `read_excel()` from the *readxl* package. If you want to import the entire table of a excel sheet, you only give the file path, the excel sheet number (or name):

```r
readxl::read_excel("path/to/file/table.xlsx", sheet = NULL)
```

more

There are some more import functions for special datasets:

```r
readRDS("path/to/file/table.Rdata") # reading R objects
readClipboard() # read from the MS clipboard (MS only)
```

Export data

To write data or tables into a file, we can use similar functions as in the import. You now only have to tell which object you want to save:

```r
write.table(object, file = "path/to/file/table.txt", sep = " ", row.names = FALSE)
write.csv(object, file = "path/to/file/table.txt", row.names = FALSE)
writeClipboard(object) # write to the MS clipboard (MS only)
saveRDS(object, file = "path/to/file/table.Rdata") # wrtie to a R object file
```

Explore and manipulate datasets

For the workshop, I downloaded some U-Pb detrital zircon data from the Rocky Mountains from the Geochron database (http://geochron.org/detritalsearch.php).

The downloaded excel file is Geochron_sample_download.xls

```r
source("R/read_geochron.R")
data <- read_geochron("Data/Geochron_sample_download_UPb.xls")

meta <- data$meta
isotopes <- data$isotopes
```

```r
head(meta)
```

```
## # A tibble: 6 x 40
##   Sample_ID       Unique_ID Sample_Description Sample_Comment Longitude Latitude
##   <chr>           <chr>     <chr>              <chr>              <dbl>    <dbl>
## 1 Whitehorse For~ GEG0000EB Sandstone          <NA>               -115.     50.9
## 2 Horsethief Cre~ GEG0000VB sandstone          <NA>               -117.     50.6
## 3 Hamill Group    GEG0000VC sandstone          <NA>               -117.     50.5
## 4 Mount Wilson F~ GEG0000VE sandstone          <NA>               -117.     52.2
## 5 Spray Lakes Gr~ GEG0000VH sandstone          <NA>               -115.     50.8
## 6 RVF             GEG0000J4 Pure quartz areni~ <NA>               -114.     49.3
## # i 34 more variables: Min_Age_Ma <dbl>, Max_Age_Ma <dbl>,
## #   Detrital_Method <chr>, Detrital_Type <chr>, Detrital_Mineral <chr>,
## #   Stratigraphic_Formation_Name <chr>, Oldest_Frac._Date_Ma <dbl>,
## #   Youngest_Frac._Date_Ma <dbl>, Metadata <chr>, Concordia_Diagram <chr>,
## #   Probability_Density <chr>, CSV_Table <chr>, GeoObject_Type <chr>,
## #   GeoObject_Class <chr>, Collection_Method <chr>, Analyst_Name <chr>,
## #   Laboratory_Name <chr>, Collector <chr>, Rock_Type <chr>, ...
```

```r
head(isotopes)
```

```
## # A tibble: 6 x 26
##   Sample_ID           Unique_ID Fraction_ID t.Pb206U238 st.Pb206U238 t.Pb207U235
##   <chr>               <chr>     <chr>             <dbl>        <dbl>       <dbl>
## 1 Whitehorse Formati~ GEG0000EB Whitehorse~       2071.        31.7        2104.
## 2 Whitehorse Formati~ GEG0000EB Whitehorse~       1780.        15.6        1852.
## 3 Whitehorse Formati~ GEG0000EB Whitehorse~       1336.        27.3        1320.
## 4 Whitehorse Formati~ GEG0000EB Whitehorse~        992.         9.19        990.
## 5 Whitehorse Formati~ GEG0000EB Whitehorse~       1108.        16.9        1113.
## 6 Whitehorse Formati~ GEG0000EB Whitehorse~        936.        17.2         954.
## # i 20 more variables: st.Pb207U235 <dbl>, t.Pb207Pb206 <dbl>,
## #   st.Pb207Pb206 <dbl>, PbPb.cor <dbl>, rho <dbl>, s.rho <dbl>,
## #   Pb206U238 <dbl>, errPb206U238 <dbl>, Pb206Pb204 <dbl>, Pb208Pb206 <dbl>,
## #   U <dbl>, ThU <dbl>, Age_206.238xTh <dbl>, Age_Error_206.238xTh <dbl>,
## #   Age_207.235xPa <dbl>, Age_Error_207.235xPar <dbl>, Age_207.206xTh <dbl>,
## #   Age_Error_207.206xTh <dbl>, Age_207.206xPa_Age <dbl>,
## #   Age_Error_207.206xPa <dbl>
```

Rename columns

rename(data, New_Name = Old_Name)

```r
rename(meta, "Oldest_Fraction_Date_Ma" = "Oldest_Frac._Date_Ma")
```

```
## # A tibble: 24 x 40
```

```
##    Sample_ID      Unique_ID Sample_Description Sample_Comment Longitude Latitude
##    <chr>          <chr>     <chr>              <chr>              <dbl>    <dbl>
##  1 Whitehorse Fo~ GEG0000EB Sandstone          <NA>               -115.     50.9
##  2 Horsethief Cr~ GEG0000VB sandstone          <NA>               -117.     50.6
##  3 Hamill Group   GEG0000VC sandstone          <NA>               -117.     50.5
##  4 Mount Wilson ~ GEG0000VE sandstone          <NA>               -117.     52.2
##  5 Spray Lakes G~ GEG0000VH sandstone          <NA>               -115.     50.8
##  6 RVF            GEG0000J4 Pure quartz areni~ <NA>               -114.     49.3
##  7 BSG-3          GEG0000J1 Coarse granular t~ <NA>               -116.     49.3
##  8 BHF            GEG0000J2 Hematitic coarse,~ <NA>               -116.     49.3
##  9 89-DM-353      GEG0000J3 pelite conglomera~ <NA>               -119.     52.8
## 10 Mount Nelson ~ GEG0000PG Interbedded quart~ <NA>               -116      49.4
## # i 14 more rows
## # i 34 more variables: Min_Age_Ma <dbl>, Max_Age_Ma <dbl>,
## #   Detrital_Method <chr>, Detrital_Type <chr>, Detrital_Mineral <chr>,
## #   Stratigraphic_Formation_Name <chr>, Oldest_Fraction_Date_Ma <dbl>,
## #   Youngest_Frac._Date_Ma <dbl>, Metadata <chr>, Concordia_Diagram <chr>,
## #   Probability_Density <chr>, CSV_Table <chr>, GeoObject_Type <chr>,
## #   GeoObject_Class <chr>, Collection_Method <chr>, Analyst_Name <chr>, ...
```

Select columns

select(data, column1, column2, column3)

```r
# select only the columns "Sample_ID", "Longitude", and  "Latitude":
select(meta, Sample_ID, Longitude, Latitude)
```

```
## # A tibble: 24 x 3
##    Sample_ID              Longitude Latitude
##    <chr>                      <dbl>    <dbl>
##  1 Whitehorse Formation       -115.     50.9
##  2 Horsethief Creek           -117.     50.6
##  3 Hamill Group               -117.     50.5
##  4 Mount Wilson Formation     -117.     52.2
##  5 Spray Lakes Group          -115.     50.8
##  6 RVF                        -114.     49.3
##  7 BSG-3                      -116.     49.3
##  8 BHF                        -116.     49.3
##  9 89-DM-353                  -119.     52.8
## 10 Mount Nelson Formation     -116      49.4
## # i 14 more rows
```

```r
# select all columns but the column "Sample_Description":
select(meta, !Sample_Description)
```

```
## # A tibble: 24 x 39
##    Sample_ID   Unique_ID Sample_Comment Longitude Latitude Min_Age_Ma Max_Age_Ma
##    <chr>       <chr>     <chr>              <dbl>    <dbl>      <dbl>      <dbl>
##  1 Whitehorse~ GEG0000EB <NA>               -115.     50.9        202        235
##  2 Horsethief~ GEG0000VB <NA>               -117.     50.6        542       1000
##  3 Hamill Gro~ GEG0000VC <NA>               -117.     50.5        488        542
##  4 Mount Wils~ GEG0000VE <NA>               -117.     52.2        444        472
##  5 Spray Lake~ GEG0000VH <NA>               -115.     50.8        299        318
##  6 RVF         GEG0000J4 <NA>               -114.     49.3       1000       1600
##  7 BSG-3       GEG0000J1 <NA>               -116.     49.3       1000       1600
##  8 BHF         GEG0000J2 <NA>               -116.     49.3       1000       1600
```

```
##  9 89-DM-353   GEG0000J3 <NA>                -119.     52.8        542       1000
## 10 Mount Nels~ GEG0000PG <NA>                -116      49.4       1000       1600
## # i 14 more rows
## # i 32 more variables: Detrital_Method <chr>, Detrital_Type <chr>,
## #   Detrital_Mineral <chr>, Stratigraphic_Formation_Name <chr>,
## #   Oldest_Frac._Date_Ma <dbl>, Youngest_Frac._Date_Ma <dbl>, Metadata <chr>,
## #   Concordia_Diagram <chr>, Probability_Density <chr>, CSV_Table <chr>,
## #   GeoObject_Type <chr>, GeoObject_Class <chr>, Collection_Method <chr>,
## #   Analyst_Name <chr>, Laboratory_Name <chr>, Collector <chr>, ...
```

Filter tables

`filter(data. column == value)` such filters can include any of the Logical Operators (or "Booleans"), such as `==`, `>`, `>=`, or `!=`:

```
# only samples from British Columbia:
filter(meta, Province == "British Columbia")
```

```
## # A tibble: 11 x 40
##    Sample_ID      Unique_ID Sample_Description Sample_Comment Longitude Latitude
##    <chr>          <chr>     <chr>              <chr>              <dbl>    <dbl>
##  1 Horsethief Cr~ GEG0000VB sandstone          <NA>               -117.     50.6
##  2 Hamill Group   GEG0000VC sandstone          <NA>               -117.     50.5
##  3 Mount Wilson ~ GEG0000VE sandstone          <NA>               -117.     52.2
##  4 89-DM-353      GEG0000J3 pelite conglomera~ <NA>               -119.     52.8
##  5 Mount Nelson ~ GEG0000PG Interbedded quart~ <NA>               -116      49.4
##  6 02TWL225P      GEG0000SK coarse-grained qu~ <NA>               -118.     50.2
##  7 02TWL307       GEG0000SL quartz-feldspar s~ <NA>               -118.     50.2
##  8 02TWL225       GEG0000SM Calcareous quartz~ <NA>               -118.     50.2
##  9 02TWL313       GEG0000SN Calcareous quartz~ <NA>               -118.     50.2
## 10 04TWL025       GEG0000SO Calcareous quartz~ <NA>               -118.     50.6
## 11 04TWL072       GEG0000SP Calcareous quartz~ <NA>               -117.     50.4
## # i 34 more variables: Min_Age_Ma <dbl>, Max_Age_Ma <dbl>,
## #   Detrital_Method <chr>, Detrital_Type <chr>, Detrital_Mineral <chr>,
## #   Stratigraphic_Formation_Name <chr>, Oldest_Frac._Date_Ma <dbl>,
## #   Youngest_Frac._Date_Ma <dbl>, Metadata <chr>, Concordia_Diagram <chr>,
## #   Probability_Density <chr>, CSV_Table <chr>, GeoObject_Type <chr>,
## #   GeoObject_Class <chr>, Collection_Method <chr>, Analyst_Name <chr>,
## #   Laboratory_Name <chr>, Collector <chr>, Rock_Type <chr>, ...
```

Calculate a new column

New columns can be calculated the following: `mutate(data, new_column1 = "Word", new_column2 = old_column1 + 1, new_column_3 = old_column2 / olc_column3)`.

```
# calculate the concordance of the U-Pb ages:
x <- mutate(isotopes, conc = ifelse(
  t.Pb206U238 > 1000,
  t.Pb206U238 / t.Pb207Pb206,
  t.Pb206U238 / t.Pb207U235
))
select(x, Fraction_ID, conc)
```

```
## # A tibble: 1,848 x 2
##    Fraction_ID          conc
##    <chr>               <dbl>
```

```
##  1 Whitehorse (1st)-1  0.970
##  2 Whitehorse (1st)-2  0.921
##  3 Whitehorse (1st)-3  1.03
##  4 Whitehorse (1st)-4  1.00
##  5 Whitehorse (1st)-5  0.987
##  6 Whitehorse (1st)-6  0.981
##  7 Whitehorse (1st)-7  0.983
##  8 Whitehorse (1st)-8  1.02
##  9 Whitehorse (1st)-9  0.988
## 10 Whitehorse (1st)-10 1.03
## # i 1,838 more rows
```

`ifelse()` does a calculation depending on a condition. `ifelse(condition, this, that)` literally means "If condition is TRUE does this. If not, do that".

Sequence of functions

A sequence of functions on the same object can be expressed like the following:

1. one after the other

```r
x <-
  mutate(
    isotopes,
    conc = ifelse(
      t.Pb206U238 > 1000,
      t.Pb206U238 / t.Pb207Pb206,
      t.Pb206U238 / t.Pb207U235
    )
  )
x <- select(x, Fraction_ID, conc)
filter(x, between(conc, 0.85, 1.05))
```

```
## # A tibble: 1,608 x 2
##    Fraction_ID          conc
##    <chr>               <dbl>
##  1 Whitehorse (1st)-1  0.970
##  2 Whitehorse (1st)-2  0.921
##  3 Whitehorse (1st)-3  1.03
##  4 Whitehorse (1st)-4  1.00
##  5 Whitehorse (1st)-5  0.987
##  6 Whitehorse (1st)-6  0.981
##  7 Whitehorse (1st)-7  0.983
##  8 Whitehorse (1st)-8  1.02
##  9 Whitehorse (1st)-9  0.988
## 10 Whitehorse (1st)-10 1.03
## # i 1,598 more rows
```

2. in one step (wrapped version)

```
## # A tibble: 1,608 x 2
##    Fraction_ID          conc
##    <chr>               <dbl>
##  1 Whitehorse (1st)-1  0.970
##  2 Whitehorse (1st)-2  0.921
##  3 Whitehorse (1st)-3  1.03
##  4 Whitehorse (1st)-4  1.00
```

```
##  5 Whitehorse (1st)-5  0.987
##  6 Whitehorse (1st)-6  0.981
##  7 Whitehorse (1st)-7  0.983
##  8 Whitehorse (1st)-8  1.02
##  9 Whitehorse (1st)-9  0.988
## 10 Whitehorse (1st)-10 1.03
## # i 1,598 more rows
```

3. pipe version

   Since R version> 4, there is a convenient and more intuitive way to have a sequence of functions –the **pipe** command `|>` (shortcut in RStudio is [CTRL]+[SHIFT]+[M]):

```
isotopes |>
  mutate(conc = ifelse(
    t.Pb206U238 > 1000,
    t.Pb206U238 / t.Pb207Pb206,
    t.Pb206U238 / t.Pb207U235
  ))  |>
  select(Fraction_ID, conc) |>
  filter(between(conc, 0.85, 1.05))
```

```
## # A tibble: 1,608 x 2
##    Fraction_ID          conc
##    <chr>               <dbl>
##  1 Whitehorse (1st)-1  0.970
##  2 Whitehorse (1st)-2  0.921
##  3 Whitehorse (1st)-3  1.03
##  4 Whitehorse (1st)-4  1.00
##  5 Whitehorse (1st)-5  0.987
##  6 Whitehorse (1st)-6  0.981
##  7 Whitehorse (1st)-7  0.983
##  8 Whitehorse (1st)-8  1.02
##  9 Whitehorse (1st)-9  0.988
## 10 Whitehorse (1st)-10 1.03
## # i 1,598 more rows
```

They all lead to the same result. . . .

Merge tables

```
require(dplyr)
combined <- left_join(isotopes, meta, by = "Sample_ID")
```

Grouped calculations or statistics

```
combined %>%
  group_by(Sample_ID) %>%
  summarise(length(na.omit(t.Pb206U238)))
```

```
## # A tibble: 24 x 2
##    Sample_ID `length(na.omit(t.Pb206U238))`
##    <chr>                                <int>
##  1 02TWL225                                35
##  2 02TWL225P                               49
##  3 02TWL307                                46
```

```
##  4 02TWL313                               31
##  5 04TWL025                               28
##  6 04TWL072                               49
##  7 89-DM-353                               4
##  8 BHF                                    11
##  9 BSG-3                                  13
## 10 BTC                                    92
## # i 14 more rows
```

Content | previous course: Basics | next course: Decriptive statistics