# Introduction to R

Practice exercises - Answer key

Kim Dill-McFarland, kadm@uw.edu

version October 28, 2021

## Contents

## Setup

Open the Intro R Rproject and start a new working script. Install any new packages, load packages, set a seed, and load data as we did in the workshop.

```
#RNAseq expression and metadata
load("data/RSTR_data_clean_subset.RData")
```

## Exercises

### Day 1: Base R

#### Project setup

1. What are the benefits of storing data in `RData` versus tables (`csv`, `tsv`, etc)?

- Several tables can be stored in a single object and can be loaded together in R with a single `load()` command
- Data are automatically compressed so they take up less hard-drive space
- Data formats are preserved in R such as factors, numbers you want to treat as characters, etc

2. Imagine a hypothetical project with the following data and results. How would you choose to setup your Rproject directory and sub-directories? This is something that may evolve over time, but it is helpful to start with a defined structure to make it easier for you and others to find things.
    - `.RData` file containing all cleaned data for the project
    - 2 `.csv` of raw RNAseq counts and sample metadata (what was cleaned to make the `.RData`)
    - 4 `.csv` with linear model results
    - 25 `.png` plots of gene expression, individual genes
    - 1 `.png` plot of gene expression, faceted with many genes
    - 2 `.R` scripts, 1 for linear modeling and 1 for making plots
    - 1 `.Rmd` report summarizing and interpreting the results

There are many options for this! I would do the following. Note that I personally like to use a lot of sub-directories.

```
project_name/
    data_clean/
        .RData
    data_raw/
        2 data .csv
    figs/
        genes/
            25 individual gene plot .png
        1 facetd gene plot .png
    results/
        4 linear model .csv
    scripts/
        2 .R scripts
    .Rmd
```

Another option.

```
project_name/
    data/
        .RData
        2 data .csv
    results/
        models/
            4 linear model .csv
        figs/
            25 individual gene plot .png
            1 facetd gene plot .png
    2 .R scripts
    .Rmd
```

### Data types

1. What is the difference between a `character` and `factor`?

- A character is any combination of alphanumeric (A-Z, 0-9) and other symbols (_ . - etc) that R treats like a single word. These are analogous to categorical variables in statistics. So for example, we have a variable with data on "MEDIA" vs "TB" in the workshop data set
- A factor is a character variable with some additional formatting. Factors have defined levels in a defined order. If you try to add data not of one of the defined levels, it is seen as an `NA` or missing.

For example, we could format our MEDIA/TB variable to a factor

```r
factor(dat$targets$condition)
```

```
## Loading required package: limma
```

```
##  [1] MEDIA TB    MEDIA TB    MEDIA TB    MEDIA TB    TB    MEDIA MEDIA TB
## [13] MEDIA TB    MEDIA TB    MEDIA TB    MEDIA TB
## Levels: MEDIA TB
```

and force TB to be the first level even though it is the second alphabetically

```r
factor(dat$targets$condition, levels=c("TB","MEDIA"))
```

```
##  [1] MEDIA TB    MEDIA TB    MEDIA TB    MEDIA TB    TB    MEDIA MEDIA TB
## [13] MEDIA TB    MEDIA TB    MEDIA TB    MEDIA TB
## Levels: TB MEDIA
```

And if we only allow TB as a level, we can see the how R replaces everything else with `NA`

```
factor(dat$targets$condition, levels=c("TB"))
```

```
##  [1] <NA> TB   <NA> TB   <NA> TB   <NA> TB   TB   <NA> <NA> TB   <NA> TB   <NA>
## [16] TB   <NA> TB   <NA> TB
## Levels: TB
```

2. What data type does R classify the date `2021.06`? What about `2021/06`? If it is not classified as a "date", how could this impact downstream analyses? Try to predict the outcomes before checking in R.

- Both are treated as numeric. The first as a number with 2 decimal digits and the second as the result of 2021 divided by 6

```
2021.06
```

```
## [1] 2021.06
```

```
class(2021.06)
```

```
## [1] "numeric"
```

```
2021/06
```

```
## [1] 336.8333
```

```
class(2021/06)
```

```
## [1] "numeric"
```

- This could dramatically impact results if these data were actually dates because 1) they are not being treated the same even though they are the same date, 2) scales will be wrong (as in 2021.06 is one month apart from 2021.07 but will be treated at 0.01 apart), 3) some functions that require a date won't run on a numeric, and other issues

2. Challenge: Checkout the package `lubridate` for functions to effectively work with dates in R.

- You can force date formatting like so and it is a lot more intuitive that base R's `as.Date()`. Here, you simply list which date components you have for year (y), month (m), and day (d) in the order they are in. That function does the rest!

```
library(lubridate)

ym("2021.06")
```

```
## [1] "2021-06-01"
```

```
ym("2021/06")
```

```
## [1] "2021-06-01"
```

```
#And if we had a day
ymd("2021.06.2")
```

```
## [1] "2021-06-02"
```

```
#A different order
mdy("06.2.2021")
```

```
## [1] "2021-06-02"
```

- https://lubridate.tidyverse.org/ has more on the `lubridate` package

3. You have an S3 list object named myDataand it contains 2 data frames named A and B. Within B there is a column named variable1. How do you access this variable?

```
myData$B$variable1
```

### Subsetting and filtering

Using dat:

1. What is the mean library size lib.size?

```
mean(dat$targets$lib.size)
```

```
## [1] 9287558
```

2. Try running summary(dat$targets). What kinds of data does it provide? Why are the results different for different variables?

```
summary(dat$targets)
```

```
##     libID              lib.size         norm.factors      FULLIDNO
##  Length:20          Min.   : 2776897   Min.   :0.7735   Length:20
##  Class :character   1st Qu.: 4625309   1st Qu.:0.8871   Class :character
##  Mode  :character   Median :10186231   Median :1.0224   Mode  :character
##                     Mean   : 9287558   Mean   :1.0082
##                     3rd Qu.:12526319   3rd Qu.:1.0903
##                     Max.   :17242699   Max.   :1.3324
##      RSID             condition
##  Length:20          Length:20
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

- You will get min, max, mean, and quartiles for numeric data
- You get the class and length of character vectors
- R automatically detects the data type and provides as much info as it can. Since character variables are simply words, this class has the least summary info.

3. How many libraries have a library size lib.size greater than 5 million and a normalization factor norm.factors less than 1?

```
size.logical <- dat$targets$lib.size > 5E6
norm.logical <- dat$targets$norm.factors < 1

dat$targets[size.logical & norm.logical, ]
```

```
##          libID lib.size norm.factors   FULLIDNO     RSID condition
## 4  RS102306_TB 10101705    0.9096415 84437-1-02 RS102306        TB
## 8  RS102244_TB  8695434    0.8544133 84457-1-02 RS102244        TB
## 9  RS102521_TB  8859096    0.7735422 91360-1-04 RS102521        TB
## 12 RS102340_TB  7368365    0.8892822 84317-1-03 RS102340        TB
## 18 RS102469_TB 10536450    0.9812365 89448-1-04 RS102469        TB
## 20 RS102484_TB 10270757    0.8390154 84427-1-02 RS102484        TB
```

```
#Or combine it all together
dat$targets[dat$targets$lib.size > 5E6 & dat$targets$norm.factors < 1, ]
```

```
##          libID lib.size norm.factors   FULLIDNO     RSID condition
## 4  RS102306_TB 10101705    0.9096415 84437-1-02 RS102306        TB
## 8  RS102244_TB  8695434    0.8544133 84457-1-02 RS102244        TB
## 9  RS102521_TB  8859096    0.7735422 91360-1-04 RS102521        TB
## 12 RS102340_TB  7368365    0.8892822 84317-1-03 RS102340        TB
## 18 RS102469_TB 10536450    0.9812365 89448-1-04 RS102469        TB
## 20 RS102484_TB 10270757    0.8390154 84427-1-02 RS102484        TB
```

```
#And bonus, you can make R count the rows for you
nrow(dat$targets[size.logical & norm.logical, ])
```

```
## [1] 6
```

4. Challenge: Using the function `grepl`, how many libraries are from a donor with an `RSID` that starts with "RS1025"?

```
dat$targets[grepl("^RS1025", dat$targets$RSID), ]
```

```
##             libID lib.size norm.factors   FULLIDNO     RSID condition
## 2     RS102531_TB  2776897    0.9161955 92527-1-08 RS102531        TB
## 3  RS102531_MEDIA  4258343    1.0732788 92527-1-08 RS102531     MEDIA
## 9     RS102521_TB  8859096    0.7735422 91360-1-04 RS102521        TB
## 11 RS102521_MEDIA 14509963    1.0349902 91360-1-04 RS102521     MEDIA
## 14    RS102548_TB 12858053    1.2734417 89902-1-07 RS102548        TB
## 15 RS102548_MEDIA 17242699    1.3323830 89902-1-07 RS102548     MEDIA
```

- Note that ^ means the start of and $ means the end of in a regular expression (regex) used in `grepl`

## R session

```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] lubridate_1.8.0 limma_3.48.3
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.1.1  magrittr_2.0.1  fastmap_1.1.0   generics_0.1.0
##  [5] tools_4.1.1     htmltools_0.5.2 yaml_2.2.1      stringi_1.7.5
##  [9] rmarkdown_2.11  knitr_1.36      stringr_1.4.0   xfun_0.27
## [13] digest_0.6.28   rlang_0.4.12    evaluate_0.14
```