Quality assurance/quality control (QA/QC) of the health data

Presenter: Dr. Ming-Chien Mark Tsou

Research Center for Environmental Changes

Academia Sinica

Outline

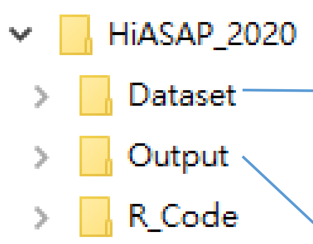
Part 1: Introduction of datasets

Part 2: Introduction of RStudio

Part 3: QA/QC of heart rate variability (HRV) data

Part 1:
Introduction of dataset

Datasets

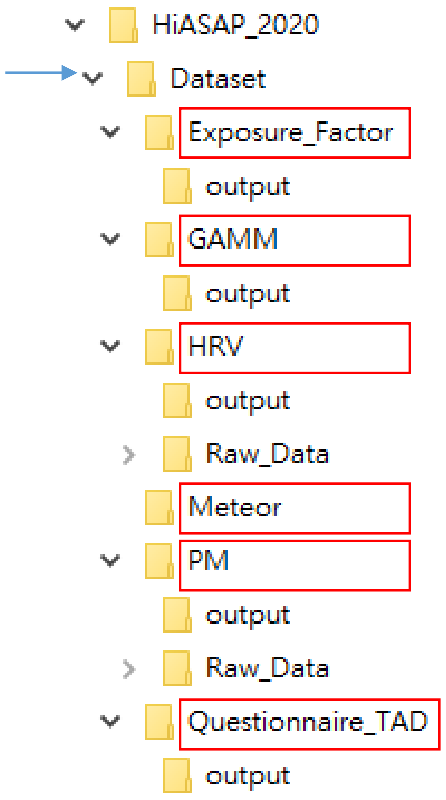


- HiASAP_2020
  - Dataset
  - Output
  - R_Code

1. The “Dataset” folder including PM, health (heart rate variability), questionnaire, time activity diaries and meteorological data

2. The “Output” folder including the all outputs which will generate in this training course

3. The “R_Code” folder



- HiASAP_2020
  - Dataset
    - Exposure_Factor
      - output
    - GAMM
      - output
    - HRV
      - output
      - Raw_Data
    - Meteor
    - PM
      - output
      - Raw_Data
    - Questionnaire_TAD
      - output

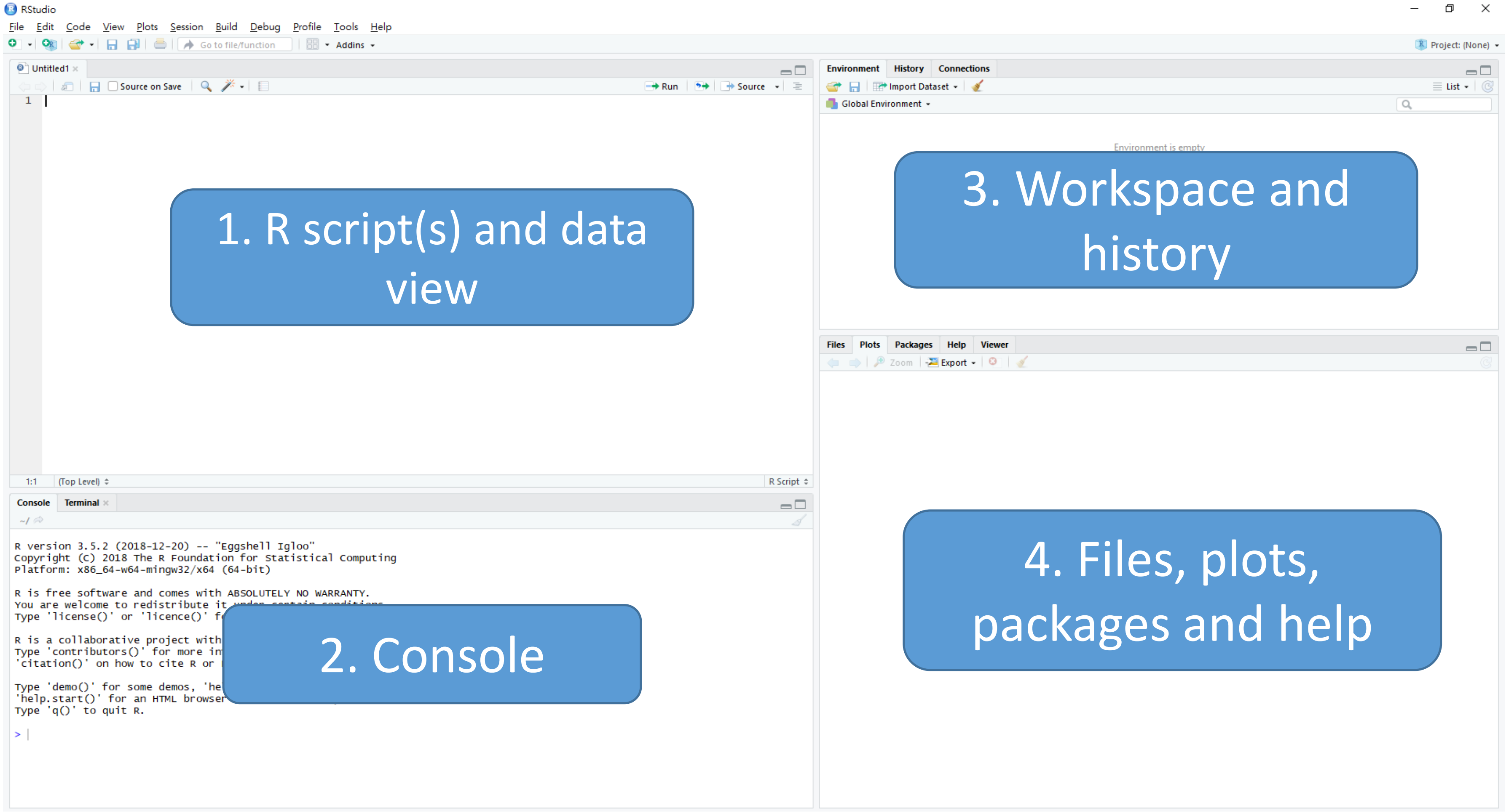
The datasets including 6 main sub-folders:

1. Exposure factors
2. GAMM (Generalized Additive Mixed Model)
3. HRV (heart rate variability) data
4. Meteorological data
5. PM data
6. Questionnaire and time-activity diary data

Part 2:
Introduction of RStudio

RStudio environments

- RStudio allows the user to run R in a more user-friendly environment.



1. R script(s) and data view

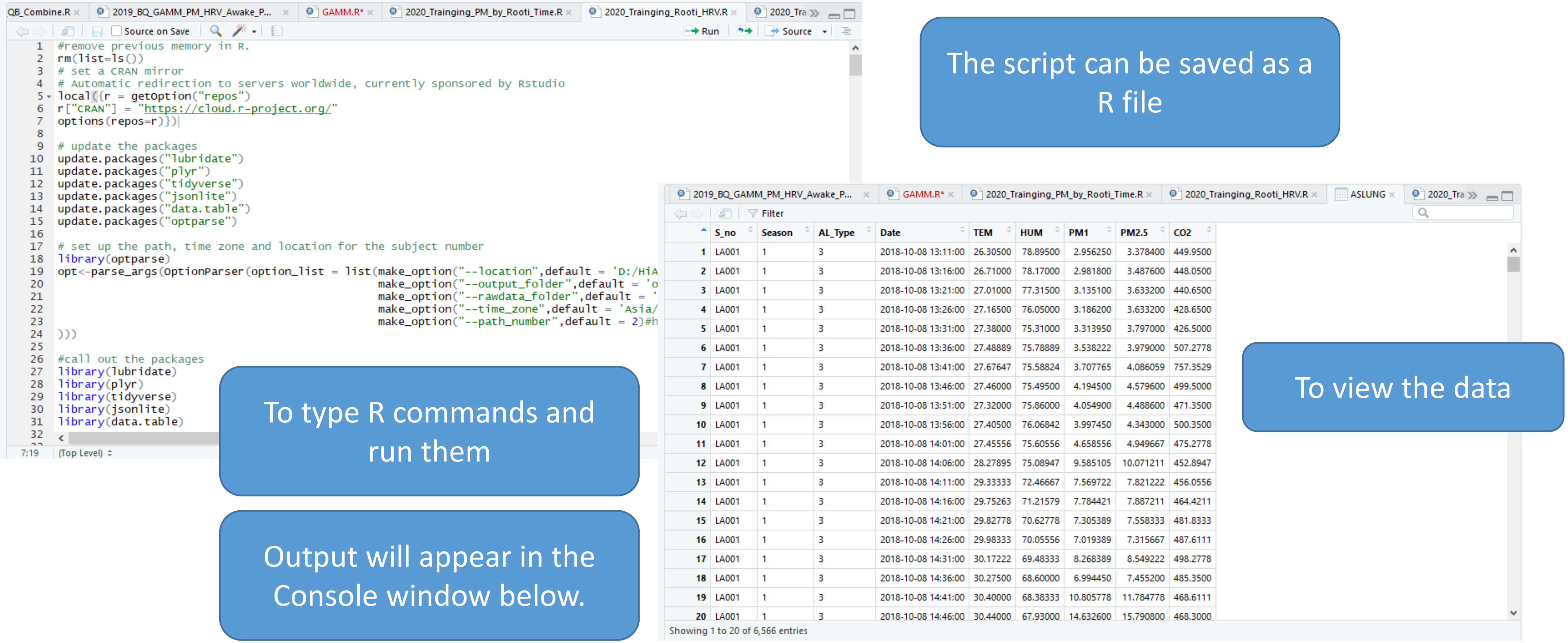
2. Console

3. Workspace and history

4. Files, plots, packages and help

```
R version 3.5.2 (2018-12-20) -- "Eggshell Igloo"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it
Type 'license()' or 'licence()' f

R is a collaborative project with
Type 'contributors()' for more in
'citation()' on how to cite R or

Type 'demo()' for some demos, 'he
'help.start()' for an HTML browser
Type 'q()' to quit R.

> |
```

1. R script(s) and data view (upper left window)


QB_Combine.R
2019_BQ_GAMM_PM_HRV_Awake_P...
GAMM.R*
2020_Trainging_PM_by_Rooti_Time.R
2020_Trainging_Rooti_HRV.R
2020_Tra
Source on Save
Run
Source
7:19
(Top Level)
ASLUNG
Filter

```r
#remove previous memory in R.
rm(list=ls())
# set a CRAN mirror
# Automatic redirection to servers worldwide, currently sponsored by Rstudio
local({r = getOption("repos")
r["CRAN"] = "https://cloud.r-project.org/"
options(repos=r)})

# update the packages
update.packages("lubridate")
update.packages("plyr")
update.packages("tidyverse")
update.packages("jsonlite")
update.packages("data.table")
update.packages("optparse")

# set up the path, time zone and location for the subject number
library(optparse)
opt<-parse_args(OptionParser(option_list = list(make_option("--location",default = 'D:/HiA
                                                make_option("--output_folder",default = 'o
                                                make_option("--rawdata_folder",default = '
                                                make_option("--time_zone",default = 'Asia/
                                                make_option("--path_number",default = 2)#h
)))

#call out the packages
library(lubridate)
library(plyr)
library(tidyverse)
library(jsonlite)
library(data.table)
```

	S_no	Season	AL_Type	Date	TEM	HUM	PM1	PM2.5	CO2
1	LA001	1	3	2018-10-08 13:11:00	26.30500	78.89500	2.956250	3.378400	449.9500
2	LA001	1	3	2018-10-08 13:16:00	26.71000	78.17000	2.981800	3.487600	448.0500
3	LA001	1	3	2018-10-08 13:21:00	27.01000	77.31500	3.135100	3.633200	440.6500
4	LA001	1	3	2018-10-08 13:26:00	27.16500	76.05000	3.186200	3.633200	428.6500
5	LA001	1	3	2018-10-08 13:31:00	27.38000	75.31000	3.313950	3.797000	426.5000
6	LA001	1	3	2018-10-08 13:36:00	27.48889	75.78889	3.538222	3.979000	507.2778
7	LA001	1	3	2018-10-08 13:41:00	27.67647	75.58824	3.707765	4.086059	757.3529
8	LA001	1	3	2018-10-08 13:46:00	27.46000	75.49500	4.194500	4.579600	499.5000
9	LA001	1	3	2018-10-08 13:51:00	27.32000	75.86000	4.054900	4.488600	471.3500
10	LA001	1	3	2018-10-08 13:56:00	27.40500	76.06842	3.997450	4.343000	500.3500
11	LA001	1	3	2018-10-08 14:01:00	27.45556	75.60556	4.658556	4.949667	475.2778
12	LA001	1	3	2018-10-08 14:06:00	28.27895	75.08947	9.585105	10.071211	452.8947
13	LA001	1	3	2018-10-08 14:11:00	29.33333	72.46667	7.569722	7.821222	456.0556
14	LA001	1	3	2018-10-08 14:16:00	29.75263	71.21579	7.784421	7.887211	464.4211
15	LA001	1	3	2018-10-08 14:21:00	29.82778	70.62778	7.305389	7.558333	481.8333
16	LA001	1	3	2018-10-08 14:26:00	29.98333	70.05556	7.019389	7.315667	487.6111
17	LA001	1	3	2018-10-08 14:31:00	30.17222	69.48333	8.268389	8.549222	498.2778
18	LA001	1	3	2018-10-08 14:36:00	30.27500	68.60000	6.994450	7.455200	485.3500
19	LA001	1	3	2018-10-08 14:41:00	30.40000	68.38333	10.805778	11.784778	468.6111
20	LA001	1	3	2018-10-08 14:46:00	30.44000	67.93000	14.632600	15.790800	468.3000

Showing 1 to 20 of 6,566 entries

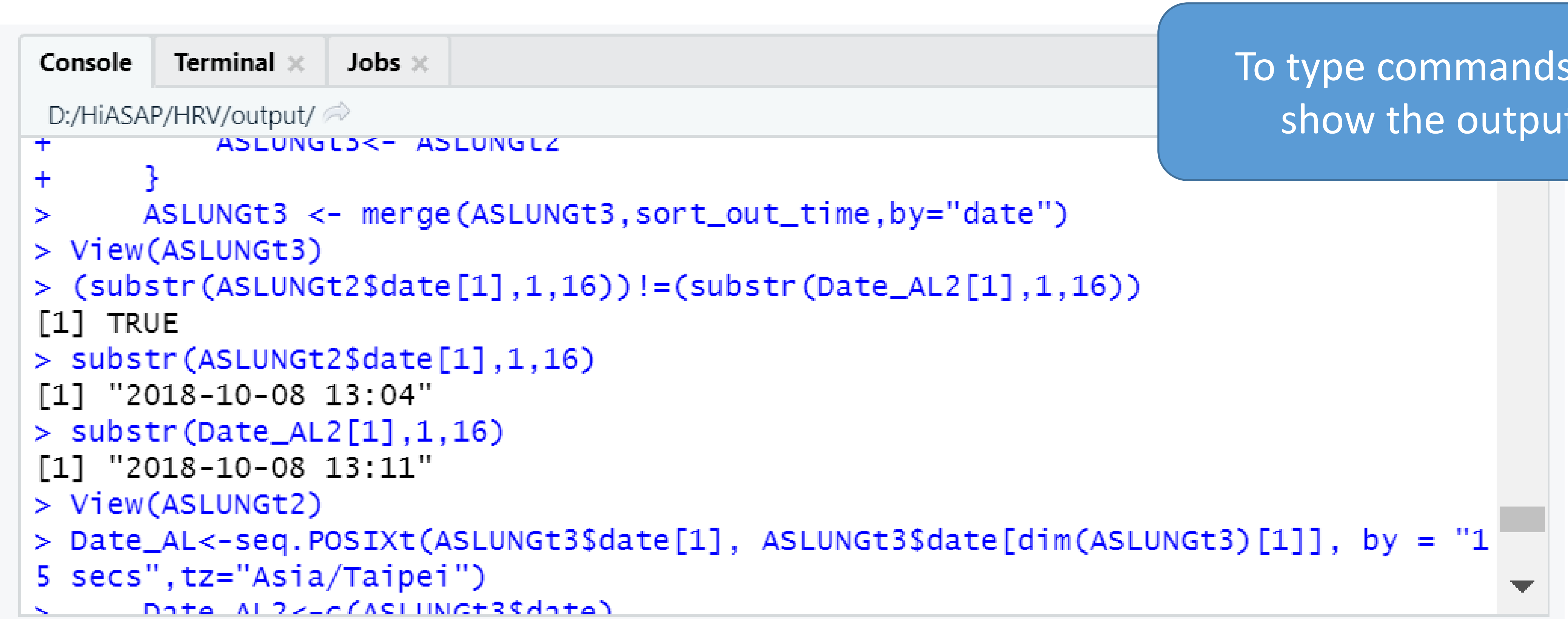
The script can be saved as a R file

To type R commands and run them

Output will appear in the Console window below.

To view the data

2. Console



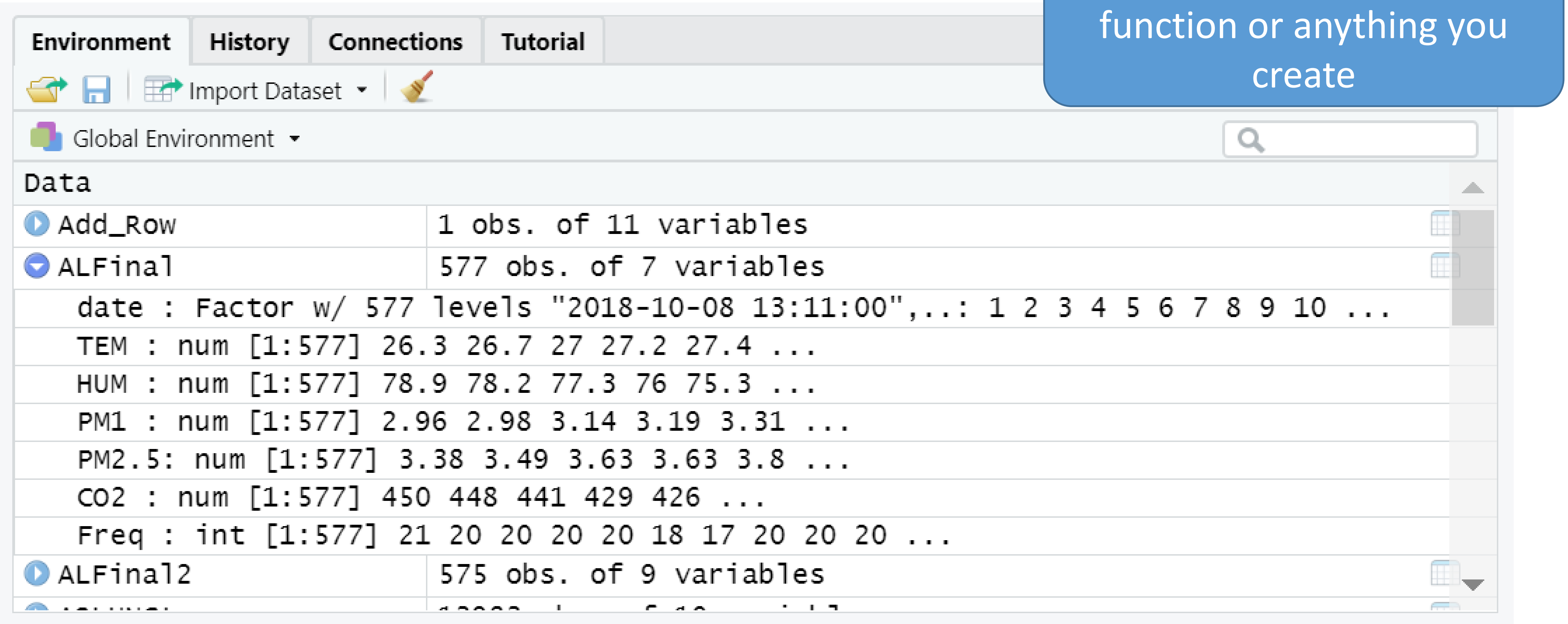
D:/HiASAP/HRV/output/

```
+          ASLUNGt3<- ASLUNGt2
+      }
>      ASLUNGt3 <- merge(ASLUNGt3,sort_out_time,by="date")
> View(ASLUNGt3)
> (substr(ASLUNGt2$date[1],1,16))!=(substr(Date_AL2[1],1,16))
[1] TRUE
> substr(ASLUNGt2$date[1],1,16)
[1] "2018-10-08 13:04"
> substr(Date_AL2[1],1,16)
[1] "2018-10-08 13:11"
> View(ASLUNGt2)
> Date_AL<-seq.POSIXt(ASLUNGt3$date[1], ASLUNGt3$date[dim(ASLUNGt3)[1]], by = "1
5 secs",tz="Asia/Taipei")
>      Date_AL2<-c(ASLUNGt3$date)
```

To type commands and show the outputs

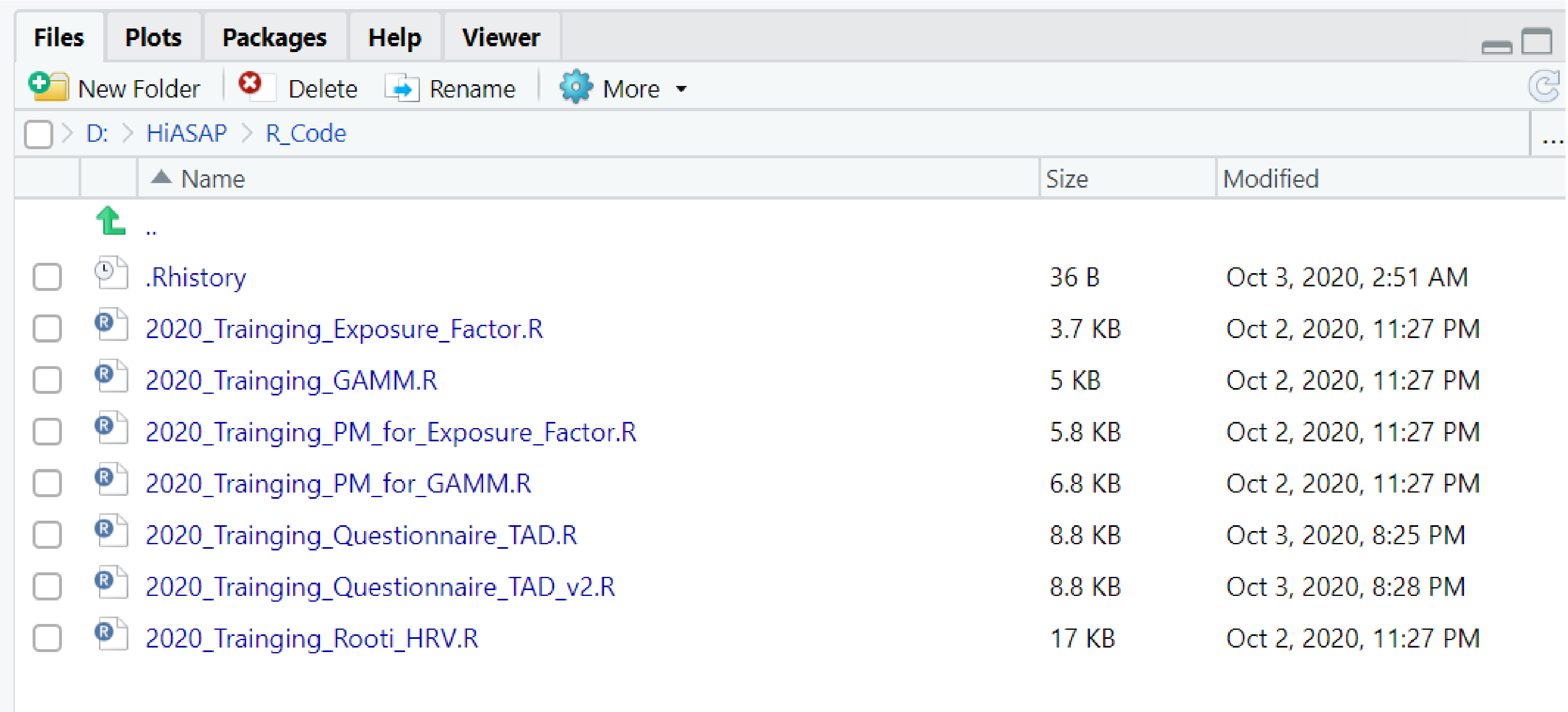
3. Workspace

To store any object, value, function or anything you create



```
Data
Add_Row        1 obs. of 11 variables
ALFinal        577 obs. of 7 variables
   date : Factor w/ 577 levels "2018-10-08 13:11:00",..: 1 2 3 4 5 6 7 8 9 10 ...
   TEM : num [1:577] 26.3 26.7 27 27.2 27.4 ...
   HUM : num [1:577] 78.9 78.2 77.3 76 75.3 ...
   PM1 : num [1:577] 2.96 2.98 3.14 3.19 3.31 ...
   PM2.5: num [1:577] 3.38 3.49 3.63 3.63 3.8 ...
   CO2 : num [1:577] 450 448 441 429 426 ...
   Freq : int [1:577] 21 20 20 20 20 18 17 20 20 20 ...
ALFinal2       575 obs. of 9 variables
```

4. Files, plots, packages and help
- Files tab



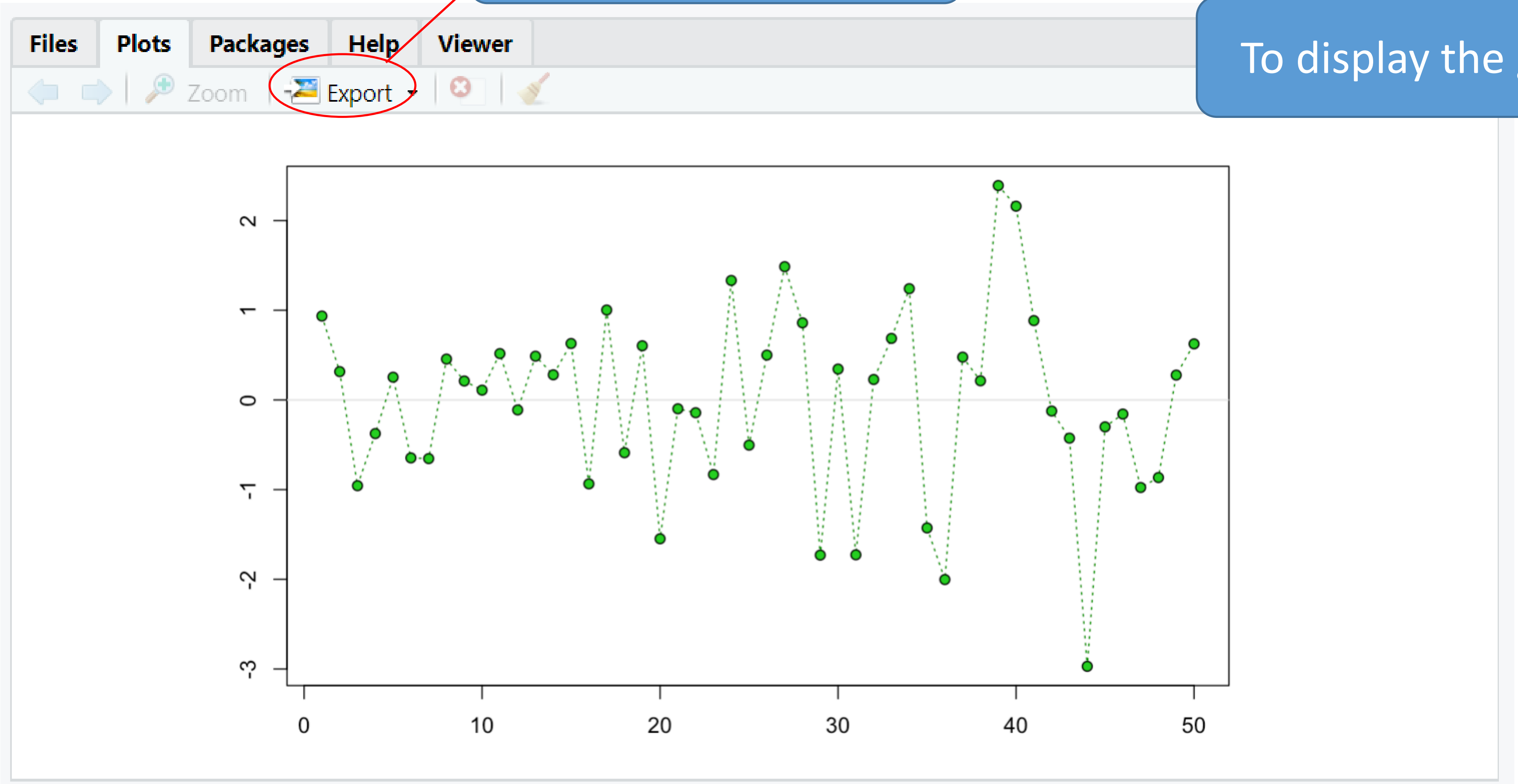
□ > D: > HiASAP > R_Code

	Name	Size	Modified
	..		
□	.Rhistory	36 B	Oct 3, 2020, 2:51 AM
□	2020_Trainging_Exposure_Factor.R	3.7 KB	Oct 2, 2020, 11:27 PM
□	2020_Trainging_GAMM.R	5 KB	Oct 2, 2020, 11:27 PM
□	2020_Trainging_PM_for_Exposure_Factor.R	5.8 KB	Oct 2, 2020, 11:27 PM
□	2020_Trainging_PM_for_GAMM.R	6.8 KB	Oct 2, 2020, 11:27 PM
□	2020_Trainging_Questionnaire_TAD.R	8.8 KB	Oct 3, 2020, 8:25 PM
□	2020_Trainging_Questionnaire_TAD_v2.R	8.8 KB	Oct 3, 2020, 8:28 PM
□	2020_Trainging_Rooti_HRV.R	17 KB	Oct 2, 2020, 11:27 PM

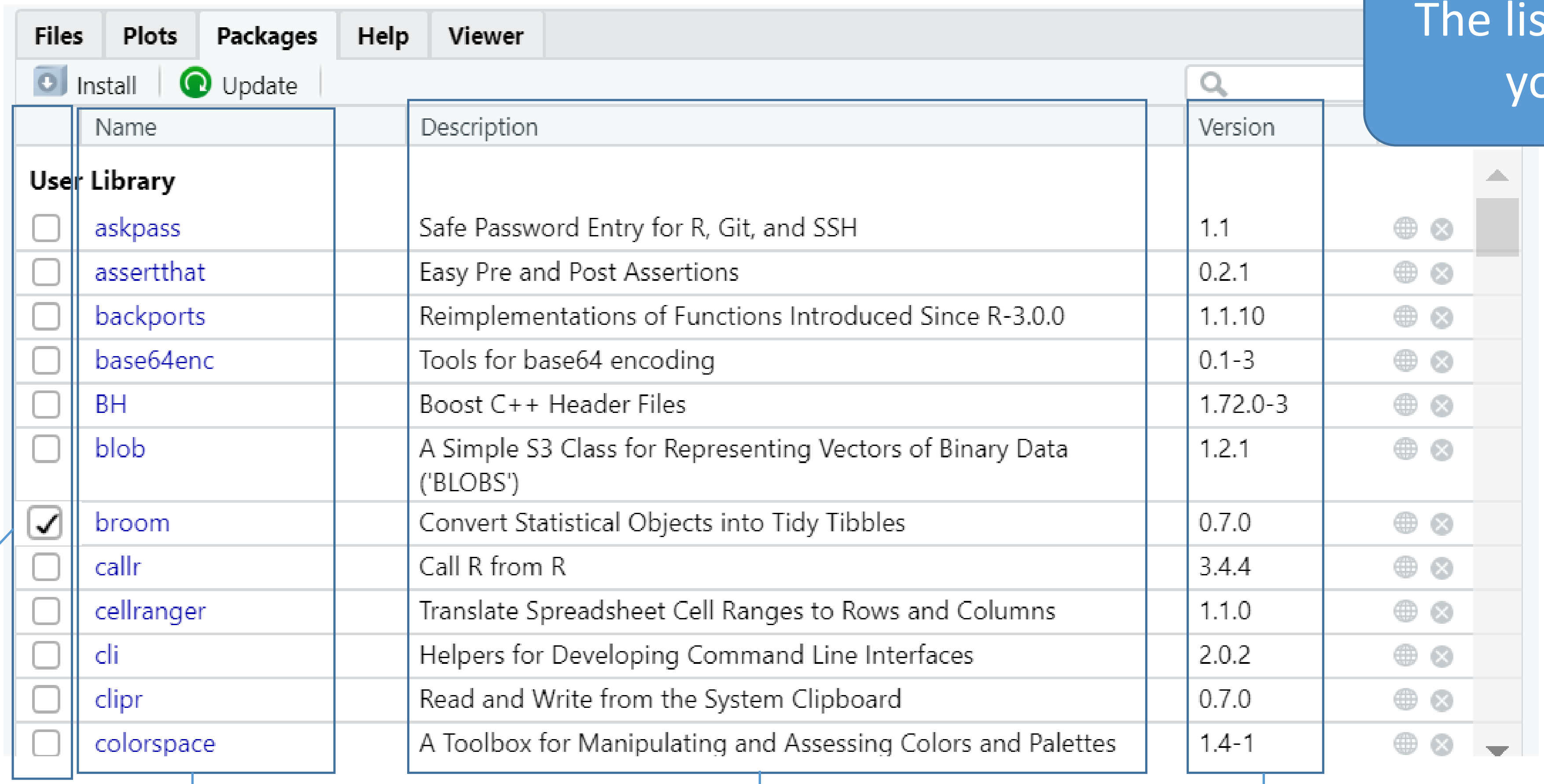
4. Files, plots, packages and help - Plots tab

To export the graphs

To display the graphs



4. Files, plots, packages and help
- Packages tab



	Name	Description	Version
	User Library		
□	askpass	Safe Password Entry for R, Git, and SSH	1.1
□	assertthat	Easy Pre and Post Assertions	0.2.1
□	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.10
□	base64enc	Tools for base64 encoding	0.1-3
□	BH	Boost C++ Header Files	1.72.0-3
□	blob	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.2.1
☑	broom	Convert Statistical Objects into Tidy Tibbles	0.7.0
□	callr	Call R from R	3.4.4
□	cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
□	cli	Helpers for Developing Command Line Interfaces	2.0.2
□	clipr	Read and Write from the System Clipboard	0.7.0
□	colorspace	A Toolbox for Manipulating and Assessing Colors and Palettes	1.4-1

The list of R packages you installed

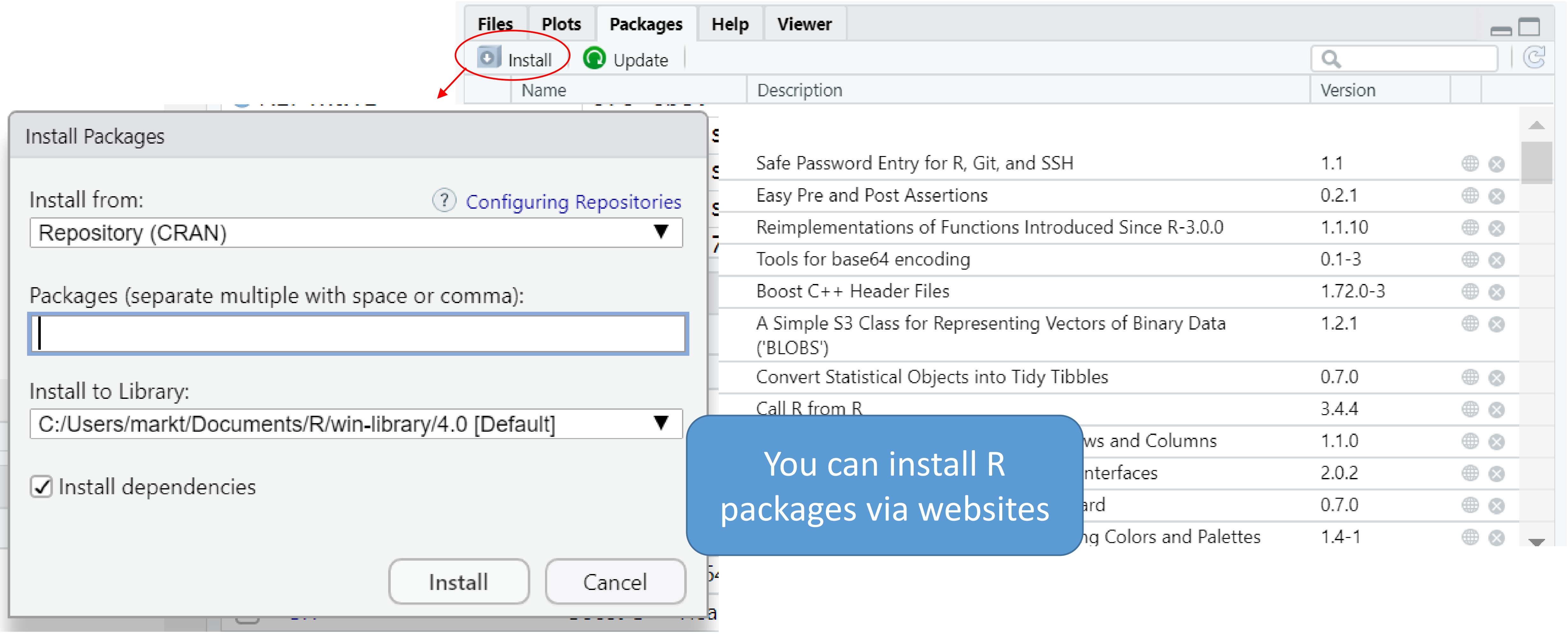
The tick means the R package has been loaded

The name of R packages

The description of R packages

The version of R packages

4. Files, plots, packages and help - Packages tab



Name	Description	Version
	Safe Password Entry for R, Git, and SSH	1.1
	Easy Pre and Post Assertions	0.2.1
	Reimplementations of Functions Introduced Since R-3.0.0	1.1.10
	Tools for base64 encoding	0.1-3
	Boost C++ Header Files	1.72.0-3
	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.2.1
	Convert Statistical Objects into Tidy Tibbles	0.7.0
	Call R from R	3.4.4
	ws and Columns	1.1.0
	nterfaces	2.0.2
	ard	0.7.0
	g Colors and Palettes	1.4-1

Install Packages

Install from: Repository (CRAN)

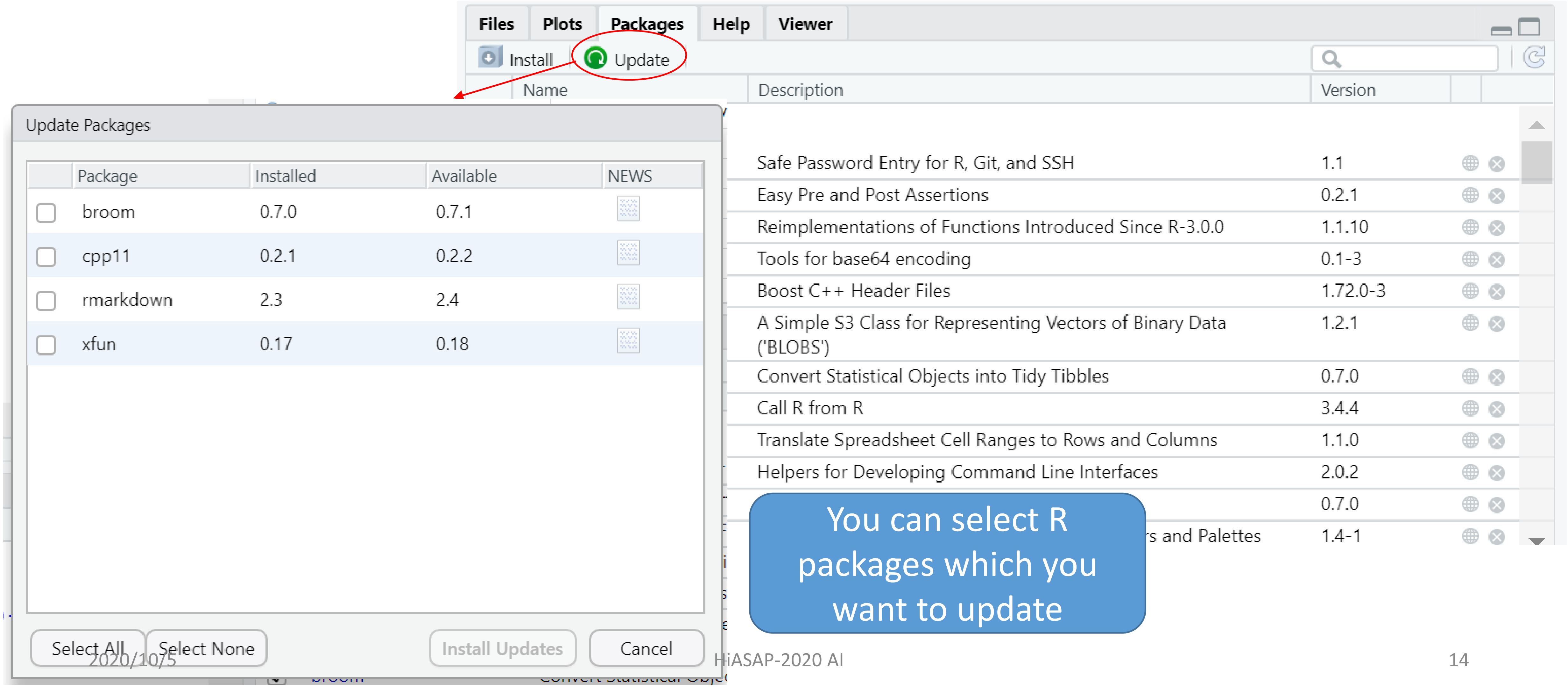
Packages (separate multiple with space or comma):

Install to Library: C:/Users/markt/Documents/R/win-library/4.0 [Default]

☑ Install dependencies

You can install R packages via websites

4. Files, plots, packages and help
- Packages tab



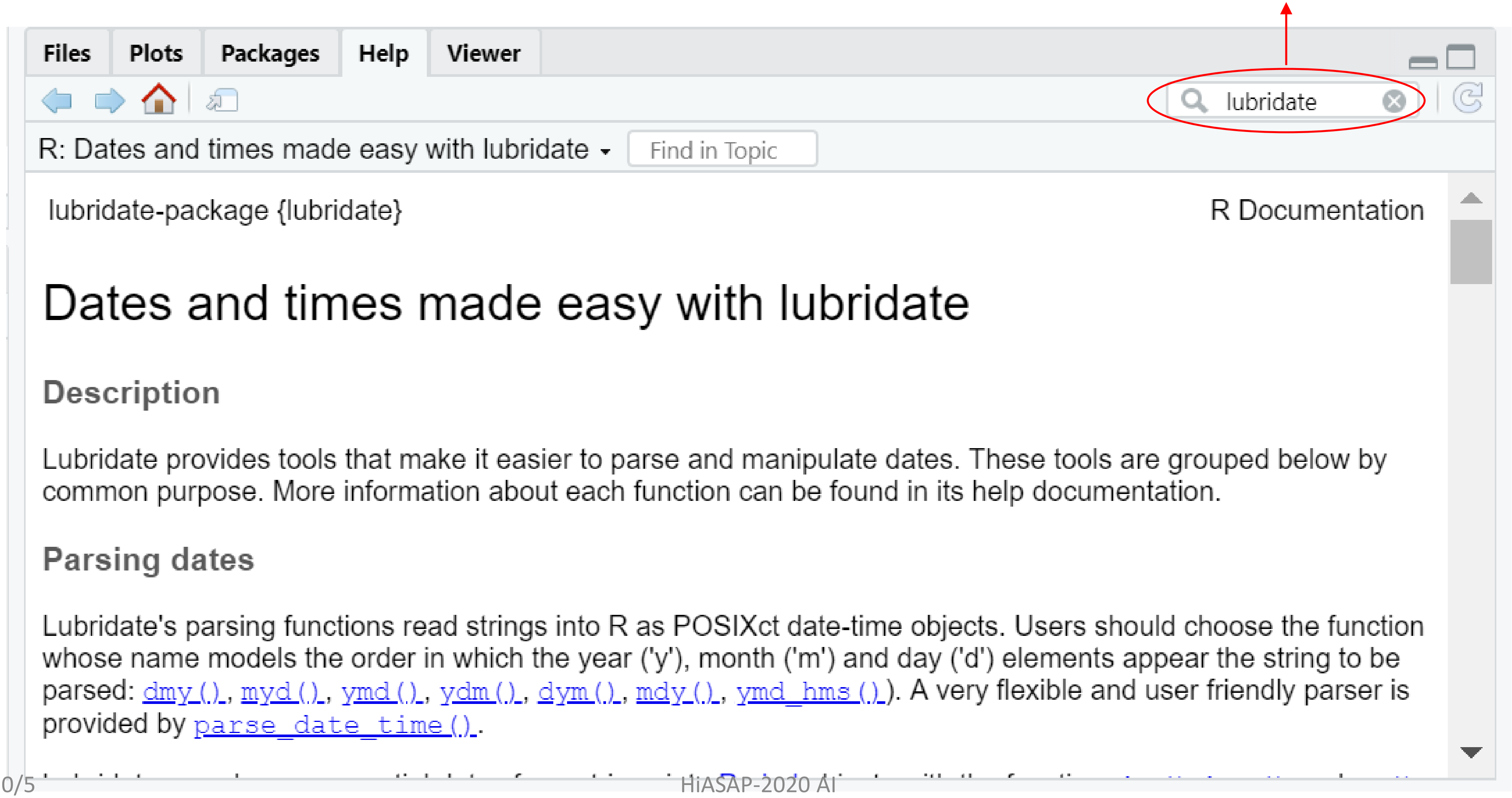
Package	Installed	Available	NEWS
□ broom	0.7.0	0.7.1	
□ cpp11	0.2.1	0.2.2	
□ rmarkdown	2.3	2.4	
□ xfun	0.17	0.18	

Name	Description	Version
	Safe Password Entry for R, Git, and SSH	1.1
	Easy Pre and Post Assertions	0.2.1
	Reimplementations of Functions Introduced Since R-3.0.0	1.1.10
	Tools for base64 encoding	0.1-3
	Boost C++ Header Files	1.72.0-3
	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.2.1
	Convert Statistical Objects into Tidy Tibbles	0.7.0
	Call R from R	3.4.4
	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
	Helpers for Developing Command Line Interfaces	2.0.2
		0.7.0
	s and Palettes	1.4-1

You can select R packages which you want to update

4. Files, plots, packages and help
- Help tab

You can search the documents of R packages



lubridate-package {lubridate} R Documentation

Dates and times made easy with lubridate

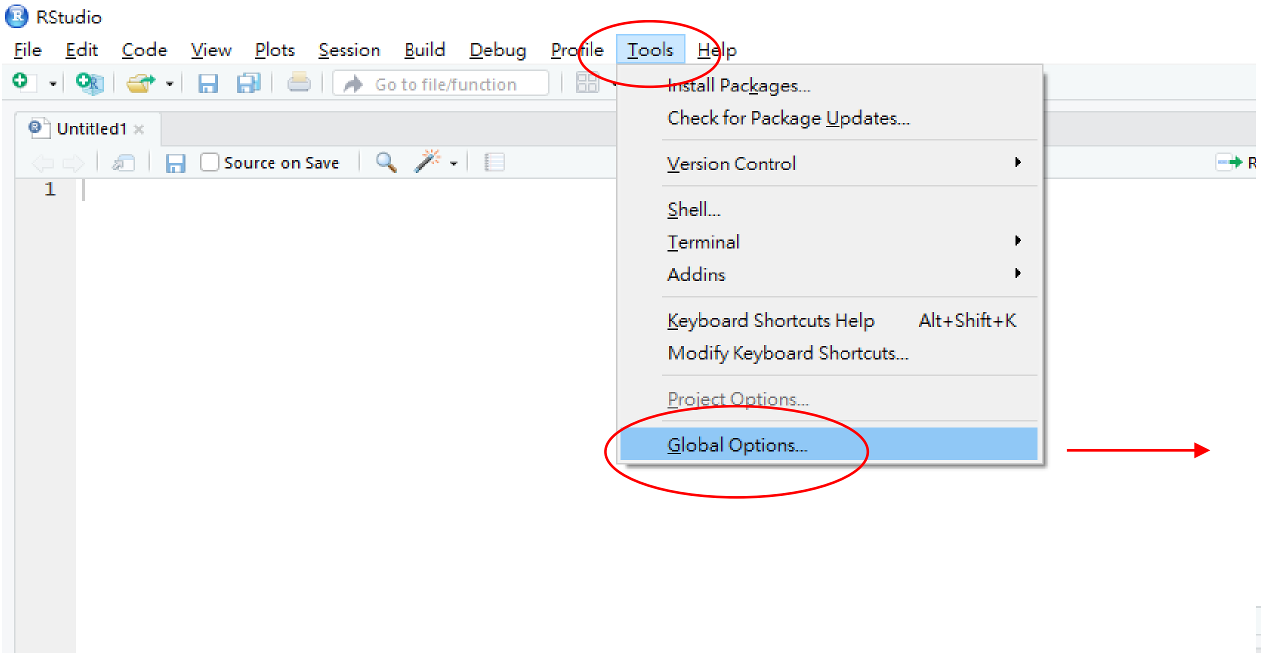
Description

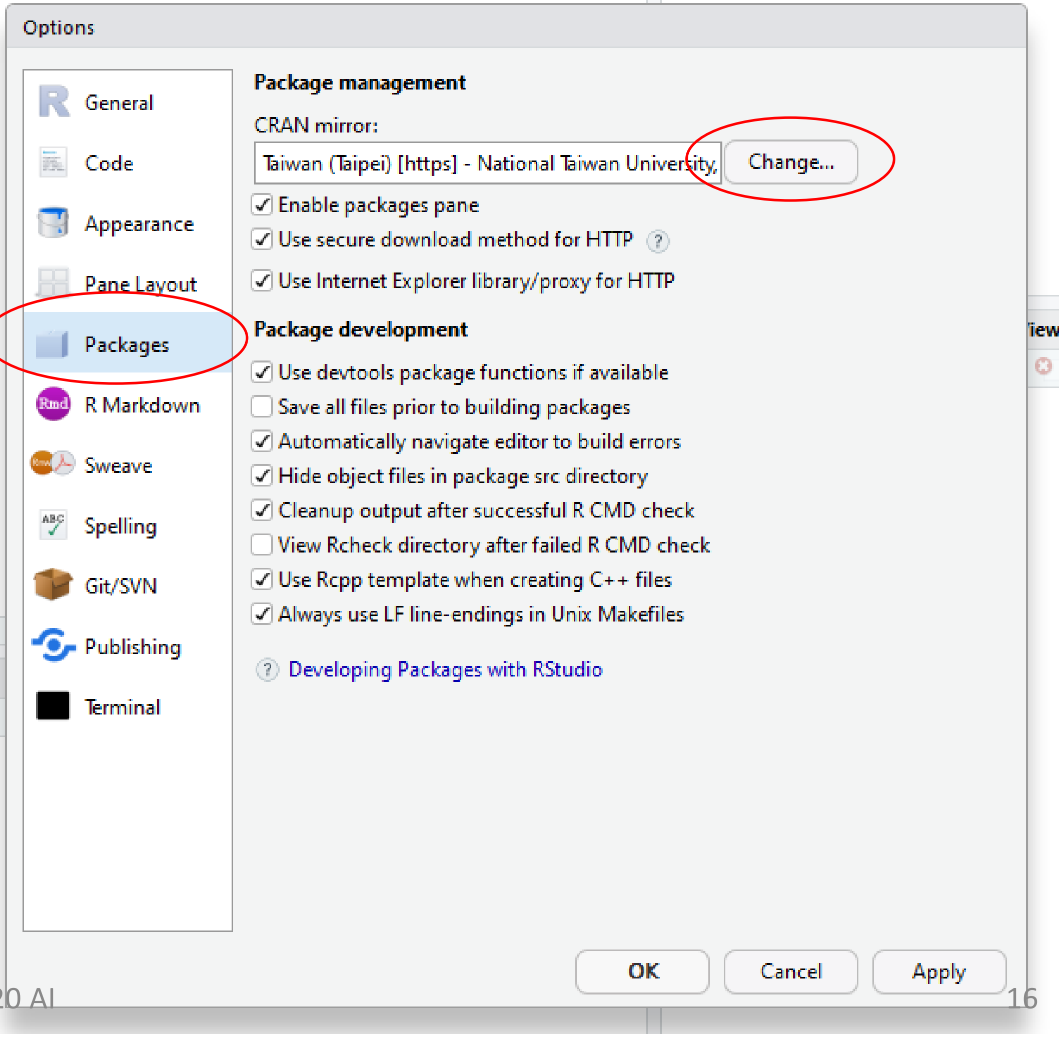
Lubridate provides tools that make it easier to parse and manipulate dates. These tools are grouped below by common purpose. More information about each function can be found in its help documentation.

Parsing dates

Lubridate's parsing functions read strings into R as POSIXct date-time objects. Users should choose the function whose name models the order in which the year ('y'), month ('m') and day ('d') elements appear the string to be parsed: dmy(), myd(), ymd(), ydm(), dym(), mdy(), ymd_hms()). A very flexible and user friendly parser is provided by parse_date_time().

To set the CRAN mirror to download/update the R packages


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Untitled1
Source on Save
1
Install Packages...
Check for Package Updates...
Version Control
Shell...
Terminal
Addins
Keyboard Shortcuts Help Alt+Shift+K
Modify Keyboard Shortcuts...
Project Options...
Global Options...



Package management

CRAN mirror: Taiwan (Taipei) [https] - National Taiwan University,

☑ Enable packages pane

☑ Use secure download method for HTTP

☑ Use Internet Explorer library/proxy for HTTP

Package development

☑ Use devtools package functions if available

□ Save all files prior to building packages

☑ Automatically navigate editor to build errors

☑ Hide object files in package src directory

☑ Cleanup output after successful R CMD check

□ View Rcheck directory after failed R CMD check

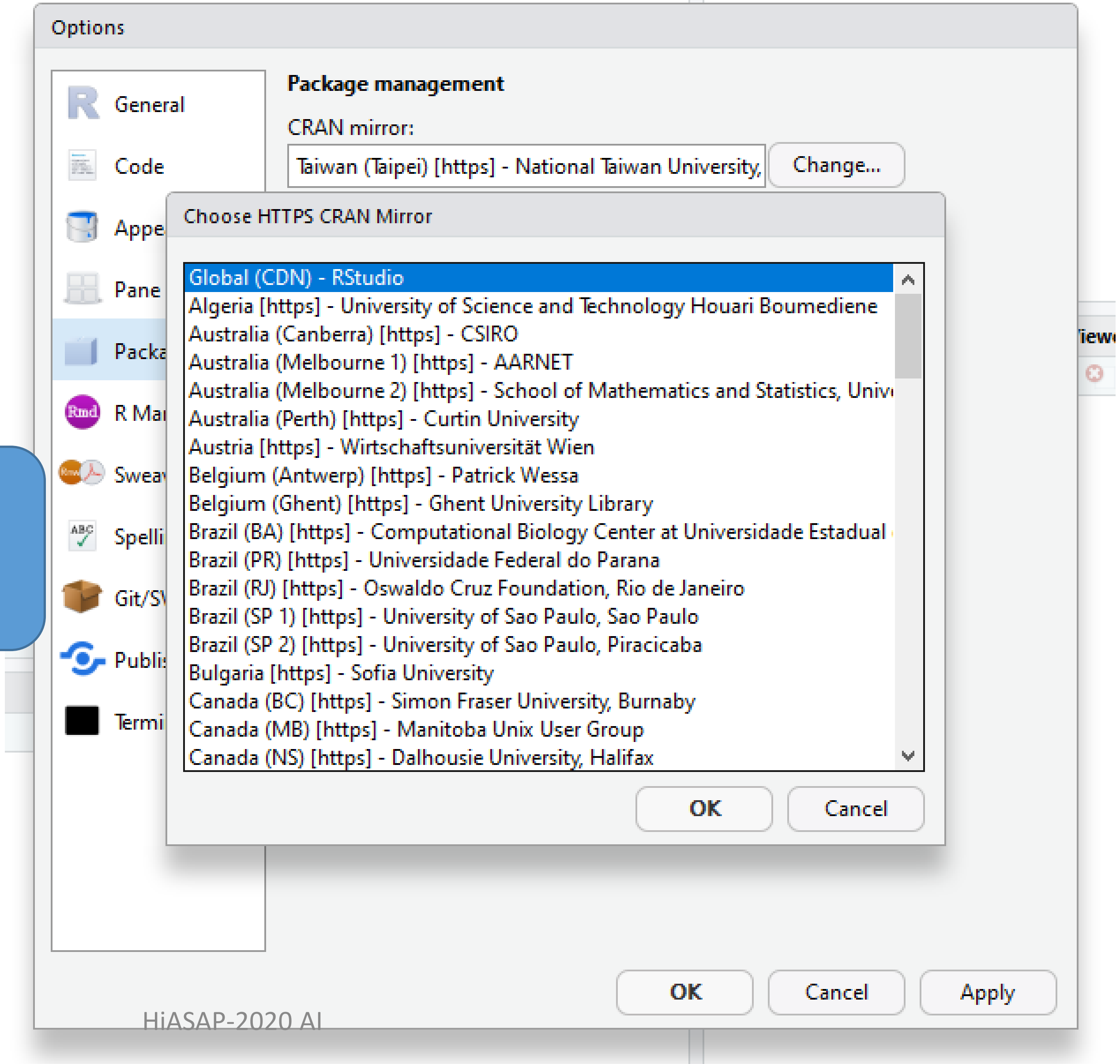
☑ Use Rcpp template when creating C++ files

☑ Always use LF line-endings in Unix Makefiles

Developing Packages with RStudio

Tools -> Global Options ->Packages -> CRAN mirror -> To click “Change” To change the CRAN mirror

To set the CRAN mirror to download/update the R packages



Package management

CRAN mirror: Taiwan (Taipei) [https] - National Taiwan University,

Choose HTTPS CRAN Mirror

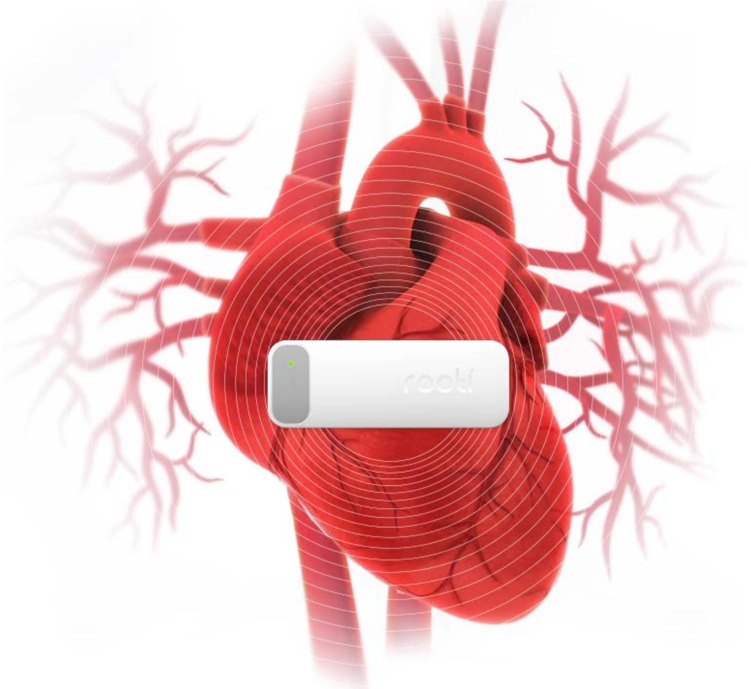
```
Global (CDN) - RStudio
Algeria [https] - University of Science and Technology Houari Boumediene
Australia (Canberra) [https] - CSIRO
Australia (Melbourne 1) [https] - AARNET
Australia (Melbourne 2) [https] - School of Mathematics and Statistics, Univ
Australia (Perth) [https] - Curtin University
Austria [https] - Wirtschaftsuniversität Wien
Belgium (Antwerp) [https] - Patrick Wessa
Belgium (Ghent) [https] - Ghent University Library
Brazil (BA) [https] - Computational Biology Center at Universidade Estadual
Brazil (PR) [https] - Universidade Federal do Parana
Brazil (RJ) [https] - Oswaldo Cruz Foundation, Rio de Janeiro
Brazil (SP 1) [https] - University of Sao Paulo, Sao Paulo
Brazil (SP 2) [https] - University of Sao Paulo, Piracicaba
Bulgaria [https] - Sofia University
Canada (BC) [https] - Simon Fraser University, Burnaby
Canada (MB) [https] - Manitoba Unix User Group
Canada (NS) [https] - Dalhousie University, Halifax
```

TO choose the CRAN mirror nearest to you to minimize network load

Part 3:
QA/QC of HRV data

What kind of data we can get from Rooti

- Standard deviation of all normal to normal intervals (SDNN)
- Square root of the mean of the sum of the squares of differences between adjacent NN intervals (RMSSD)
- LF (low frequency power)
- HF (high frequency power)
- VLF (very low frequency power)
- TP (total power)
- HF/LF ratio
- HR (heart rate)
- Activity data
- Sleep index

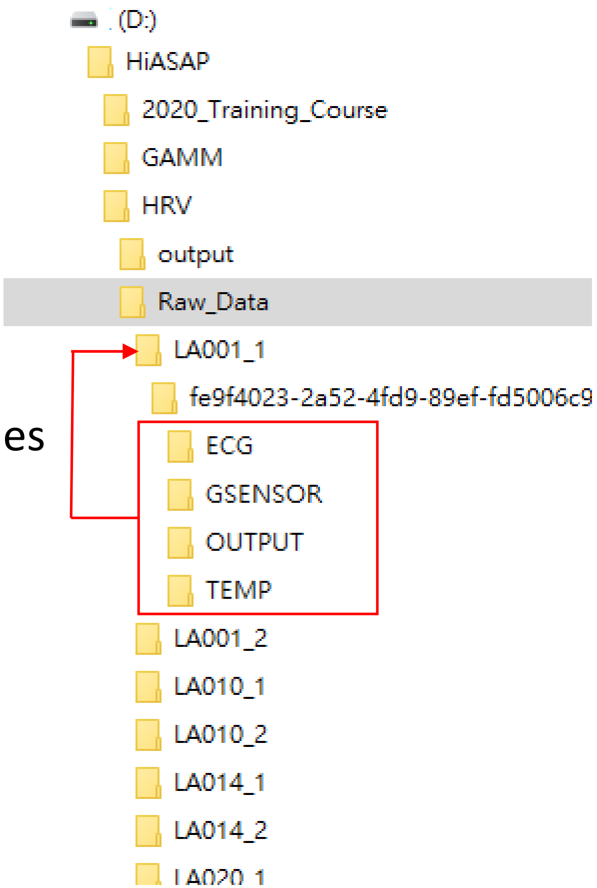


RootiRx sensor

Source: https://www.rootilabs.com/

Moving all data files to its own file for every subject

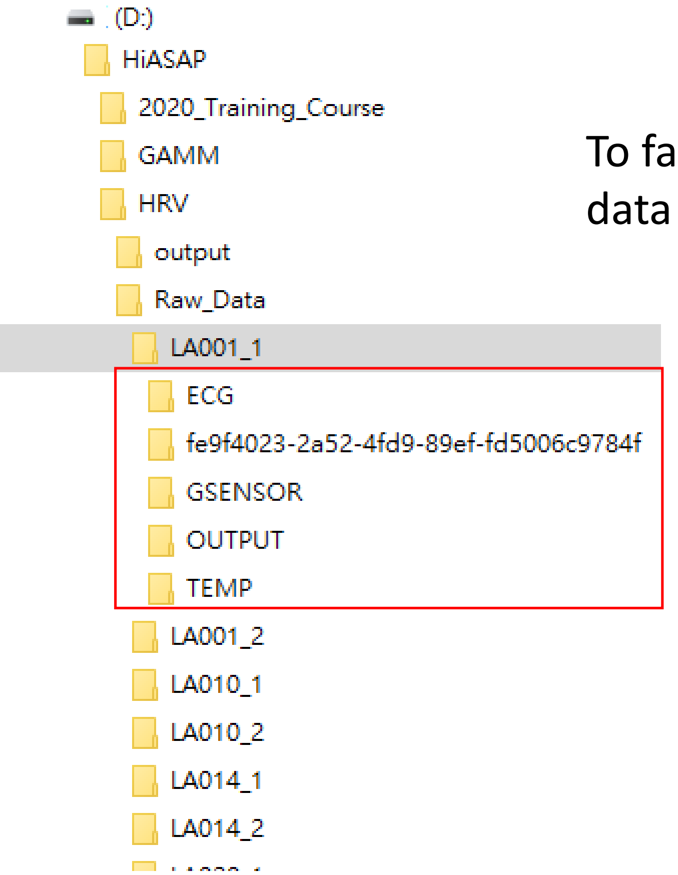
Before



```
(D:)
  HiASAP
    2020_Training_Course
    GAMM
    HRV
      output
      Raw_Data
        LA001_1
          fe9f4023-2a52-4fd9-89ef-fd5006c9
          ECG
          GSENSOR
          OUTPUT
          TEMP
        LA001_2
        LA010_1
        LA010_2
        LA014_1
        LA014_2
        LA020_1
```

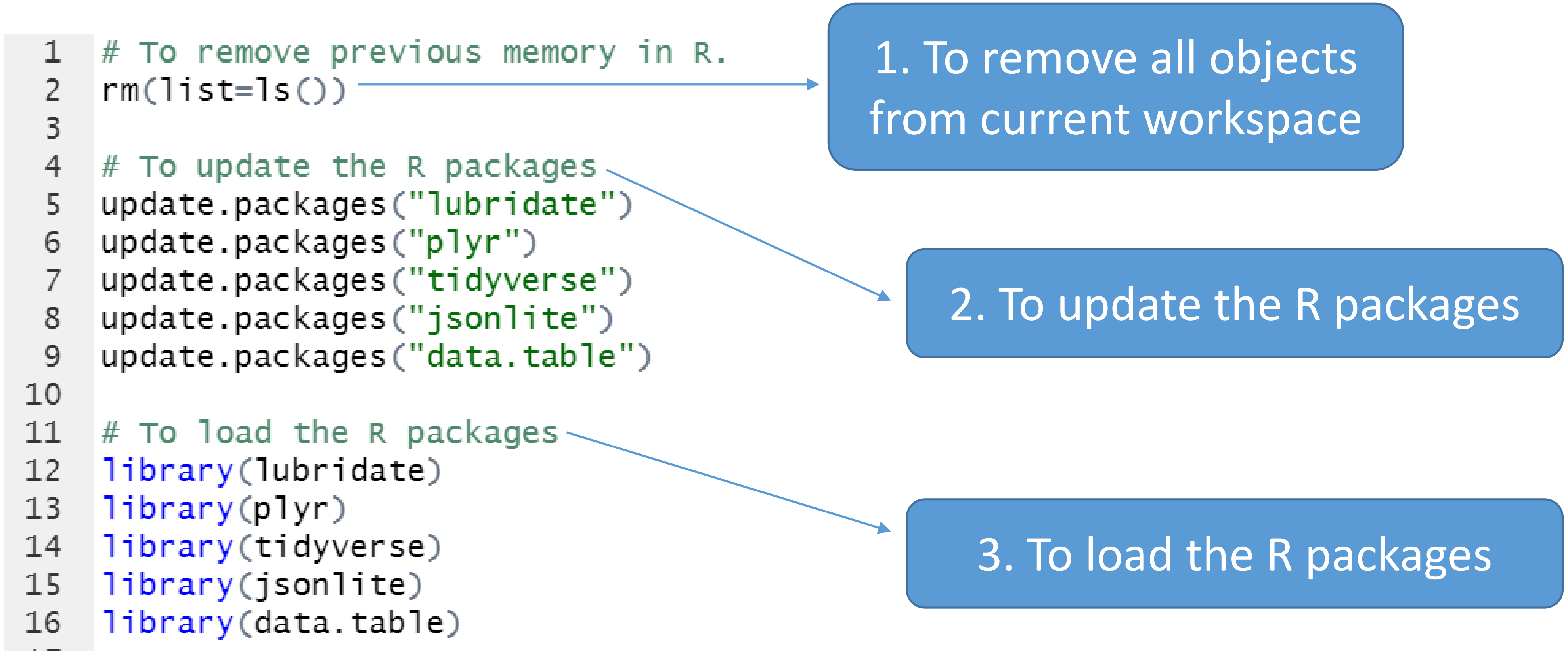
Moving files

After



```
(D:)
  HiASAP
    2020_Training_Course
    GAMM
    HRV
      output
      Raw_Data
        LA001_1
          ECG
          fe9f4023-2a52-4fd9-89ef-fd5006c9784f
          GSENSOR
          OUTPUT
          TEMP
        LA001_2
        LA010_1
        LA010_2
        LA014_1
        LA014_2
```

To facilitate to read data during analysis



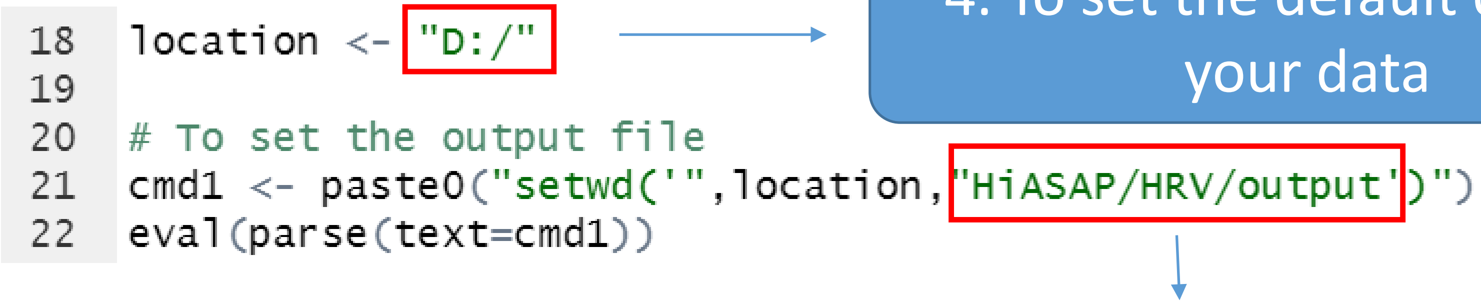
```r
# To remove previous memory in R.
rm(list=ls())

# To update the R packages
update.packages("lubridate")
update.packages("plyr")
update.packages("tidyverse")
update.packages("jsonlite")
update.packages("data.table")

# To load the R packages
library(lubridate)
library(plyr)
library(tidyverse)
library(jsonlite)
library(data.table)
```

1. To remove all objects from current workspace

2. To update the R packages

3. To load the R packages

□ → You can modify by yourself

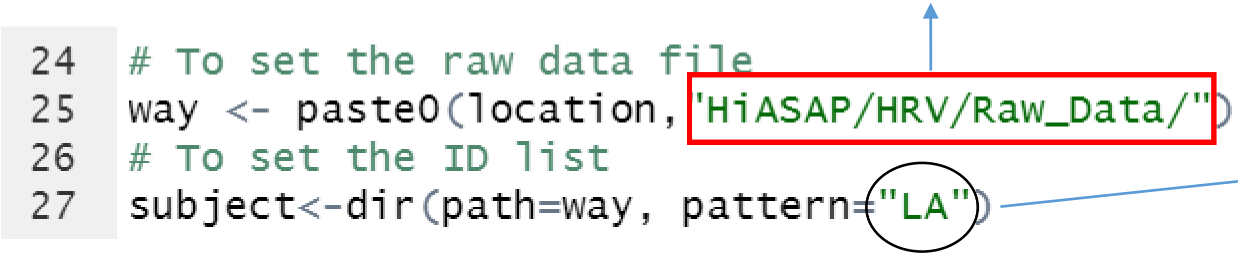


```
location <- "D:/"

# To set the output file
cmd1 <- paste0("setwd('",location,"HiASAP/HRV/output')")
eval(parse(text=cmd1))
```

4. To set the default drive of your data

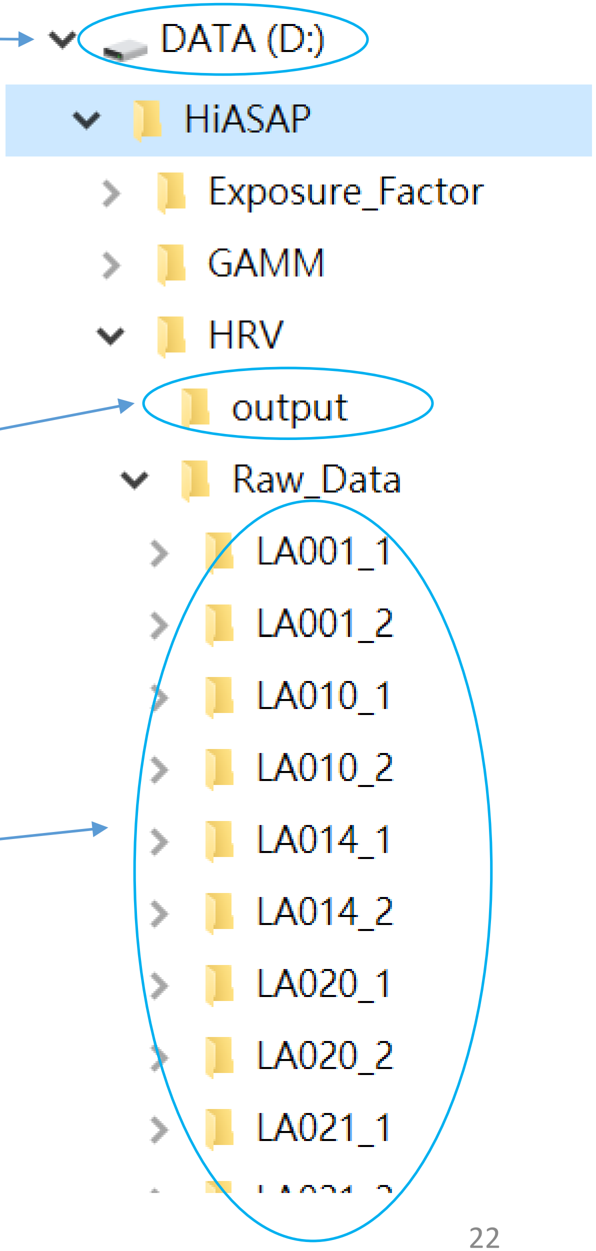
5. To set the path of output file

6. To set the path of HRV data



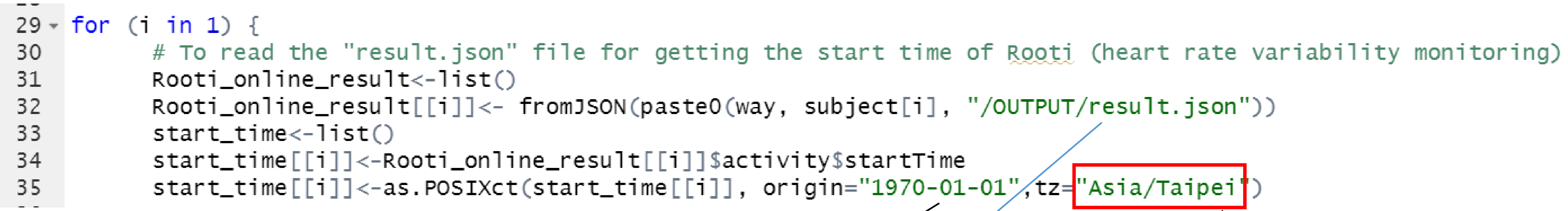
```
# To set the raw data file
way <- paste0(location,"HiASAP/HRV/Raw_Data/")
# To set the ID list
subject<-dir(path=way, pattern="LA")
```

To set the pattern (keyword) to select files



7. To read the results of HRV monitoring for getting the start time

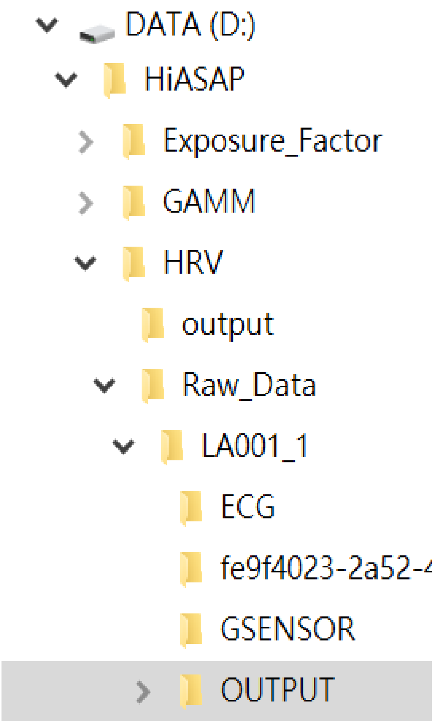
□ → You can modify by yourself

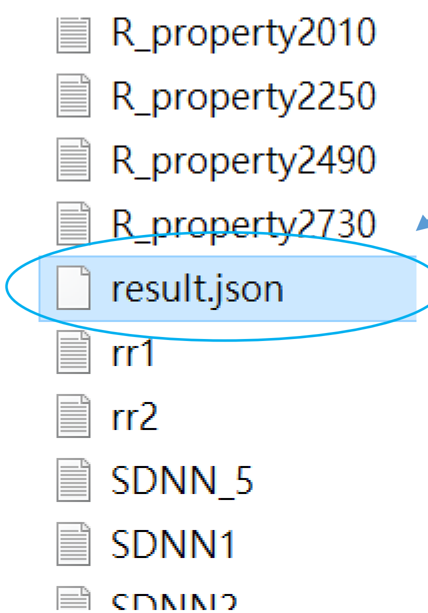


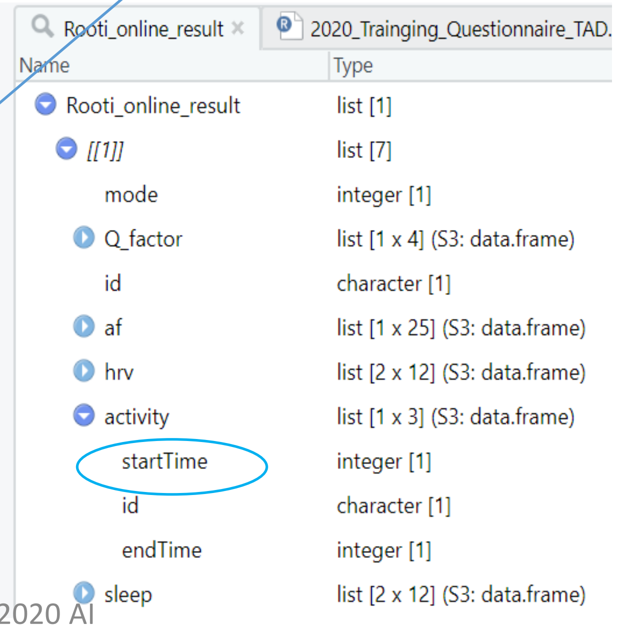
```
for (i in 1) {
     # To read the "result.json" file for getting the start time of Rooti (heart rate variability monitoring)
     Rooti_online_result<-list()
     Rooti_online_result[[i]]<- fromJSON(paste0(way, subject[i], "/OUTPUT/result.json"))
     start_time<-list()
     start_time[[i]]<-Rooti_online_result[[i]]$activity$startTime
     start_time[[i]]<-as.POSIXct(start_time[[i]], origin="1970-01-01",tz="Asia/Taipei")
```

The time is present as how many seconds has passed since Jan 1, 1970

The code of time zone







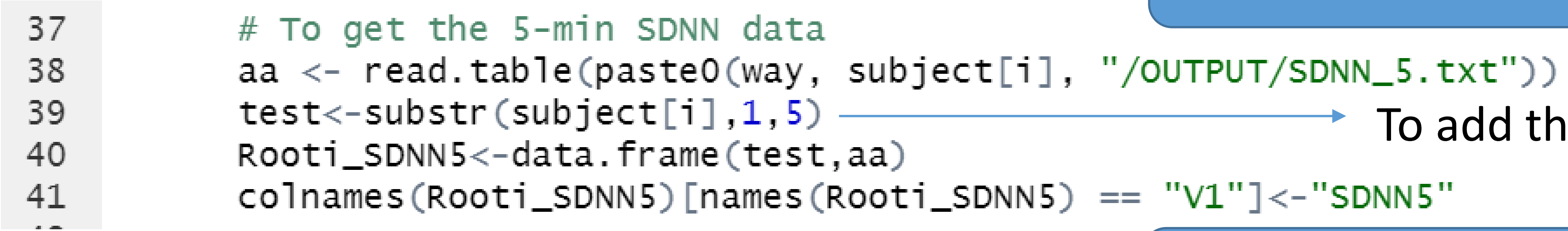
Name	Type
Rooti_online_result	list [1]
[[1]]	list [7]
mode	integer [1]
Q_factor	list [1 x 4] (S3: data.frame)
id	character [1]
af	list [1 x 25] (S3: data.frame)
hrv	list [2 x 12] (S3: data.frame)
activity	list [1 x 3] (S3: data.frame)
startTime	integer [1]
id	character [1]
endTime	integer [1]
sleep	list [2 x 12] (S3: data.frame)

More detailed information about the code of time zone can be found in the following website: https://data.iana.org/time-zones/theory.html

Extracting the HRV data

From line 37 to 211, we extract the results of each HRV indices with the almost same procedure

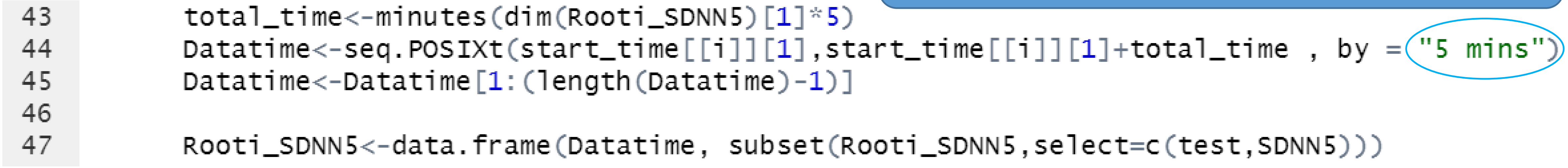
8. To read the results of 5-min SDNN



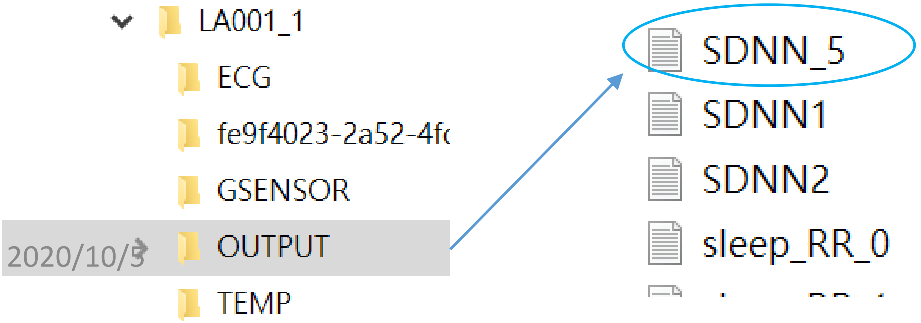
```
37        # To get the 5-min SDNN data
38        aa <- read.table(paste0(way, subject[i], "/OUTPUT/SDNN_5.txt"))
39        test<-substr(subject[i],1,5)
40        Rooti_SDNN5<-data.frame(test,aa)
41        colnames(Rooti_SDNN5)[names(Rooti_SDNN5) == "V1"]<-"SDNN5"
```

To add the variable of subjects' ID

9. To create the variable of data time

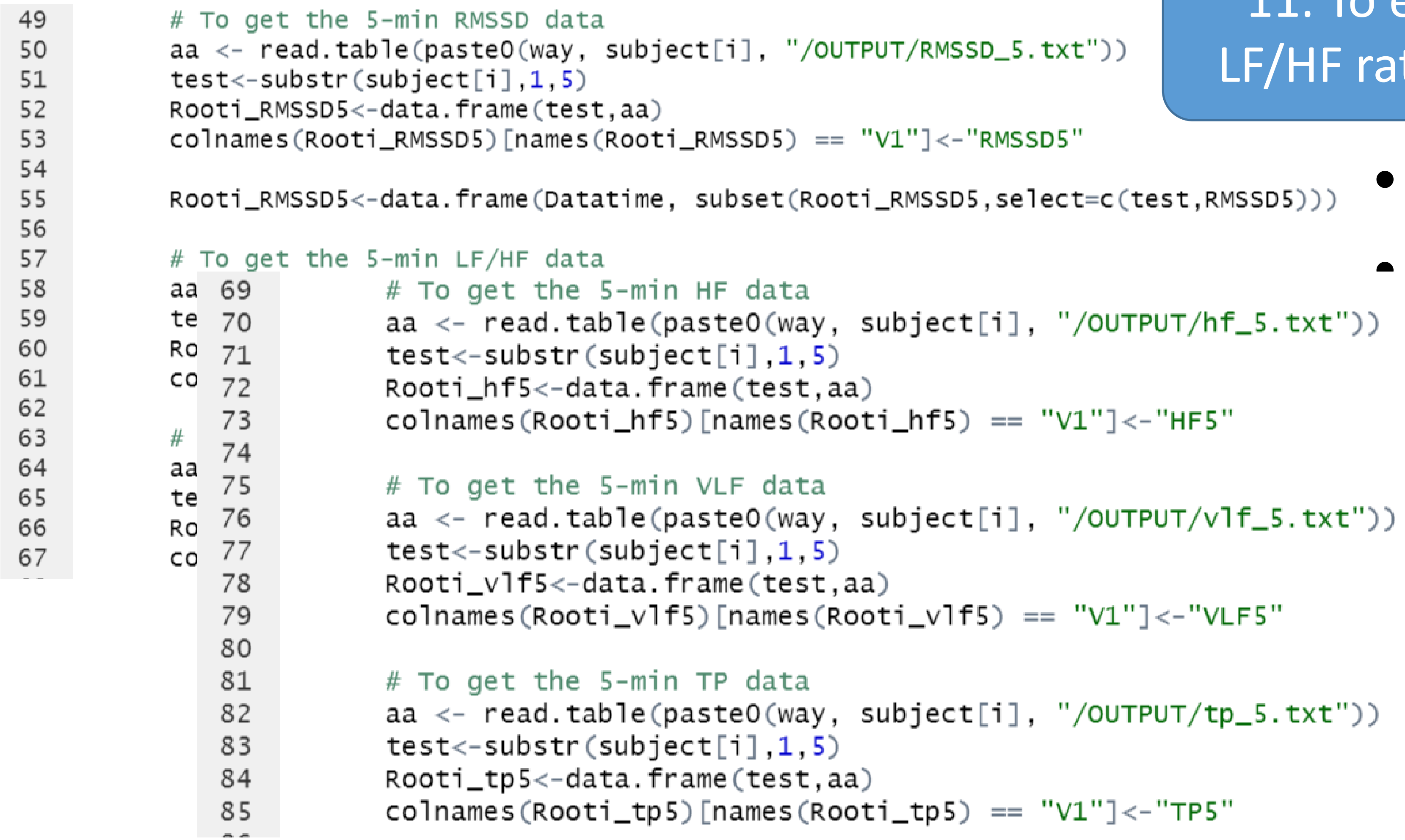


```
43        total_time<-minutes(dim(Rooti_SDNN5)[1]*5)
44        Datatime<-seq.POSIXt(start_time[[i]][1],start_time[[i]][1]+total_time , by ="5 mins")
45        Datatime<-Datatime[1:(length(Datatime)-1)]
46
47        Rooti_SDNN5<-data.frame(Datatime, subset(Rooti_SDNN5,select=c(test,SDNN5)))
```



10. To combine data time with SDNN data

Extracting the HRV data



```
49      # To get the 5-min RMSSD data
50      aa <- read.table(paste0(way, subject[i], "/OUTPUT/RMSSD_5.txt"))
51      test<-substr(subject[i],1,5)
52      Rooti_RMSSD5<-data.frame(test,aa)
53      colnames(Rooti_RMSSD5)[names(Rooti_RMSSD5) == "V1"]<-"RMSSD5"
54
55      Rooti_RMSSD5<-data.frame(Datatime, subset(Rooti_RMSSD5,select=c(test,RMSSD5)))
56
57      # To get the 5-min LF/HF data
58      aa
59      te
60      Ro
61      co
62
63      #
64      aa
65      te
66      Ro
67      co
```

```
69      # To get the 5-min HF data
70      aa <- read.table(paste0(way, subject[i], "/OUTPUT/hf_5.txt"))
71      test<-substr(subject[i],1,5)
72      Rooti_hf5<-data.frame(test,aa)
73      colnames(Rooti_hf5)[names(Rooti_hf5) == "V1"]<-"HF5"
74
75      # To get the 5-min VLF data
76      aa <- read.table(paste0(way, subject[i], "/OUTPUT/vlf_5.txt"))
77      test<-substr(subject[i],1,5)
78      Rooti_vlf5<-data.frame(test,aa)
79      colnames(Rooti_vlf5)[names(Rooti_vlf5) == "V1"]<-"VLF5"
80
81      # To get the 5-min TP data
82      aa <- read.table(paste0(way, subject[i], "/OUTPUT/tp_5.txt"))
83      test<-substr(subject[i],1,5)
84      Rooti_tp5<-data.frame(test,aa)
85      colnames(Rooti_tp5)[names(Rooti_tp5) == "V1"]<-"TP5"
```

11. To extract the 5-min RMSSD, LF/HF ratio, LF, HF, VLF and TP data

- From Line 49 to 85
- The same procedure as SDNN

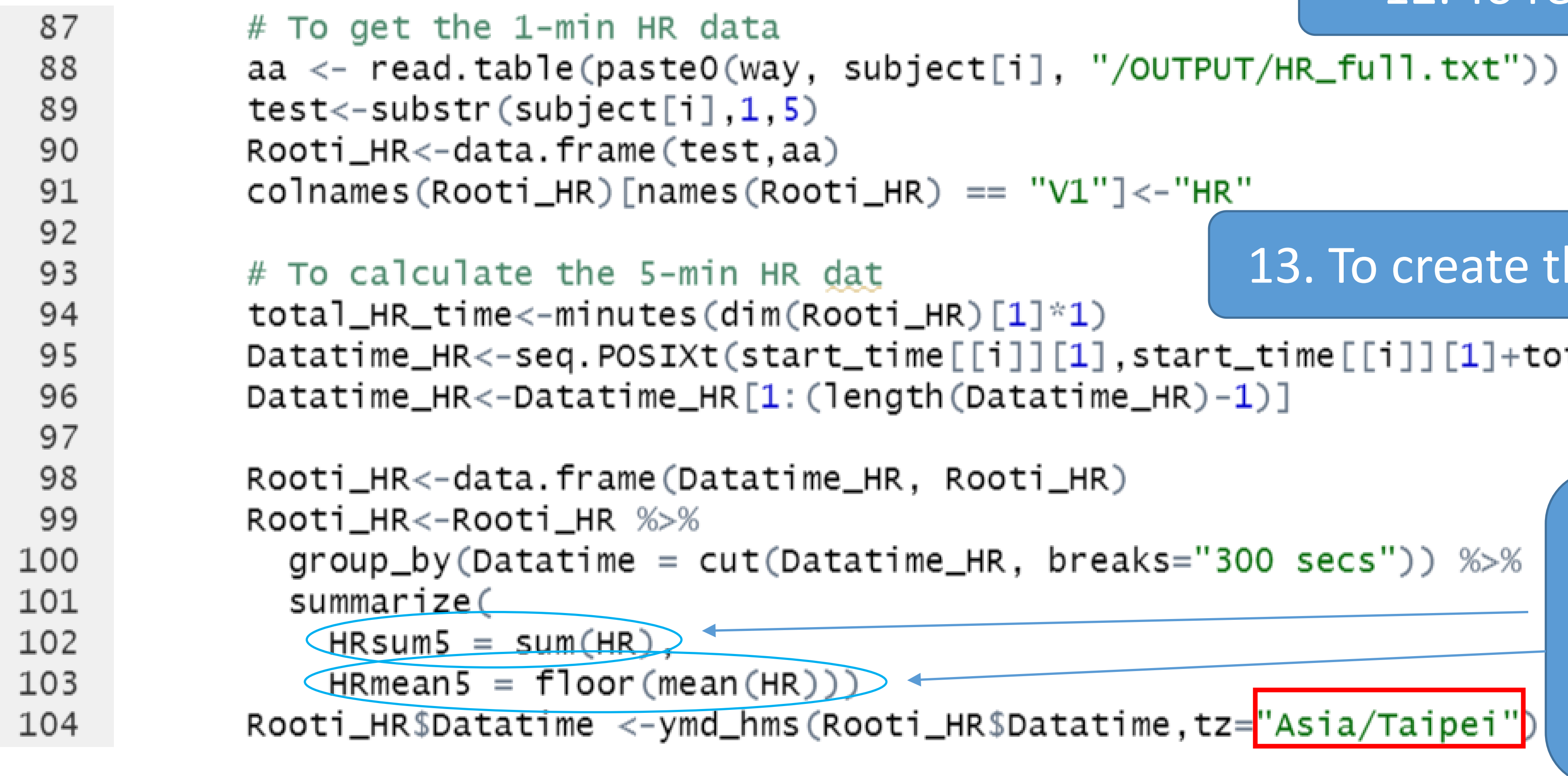
□ → You can modify by yourself

Extracting the heart rate (HR) data

12. To read the 1-min HR data

13. To create the variable of data time

14. To calculate the sum and mean of HR data for 5-min intervals



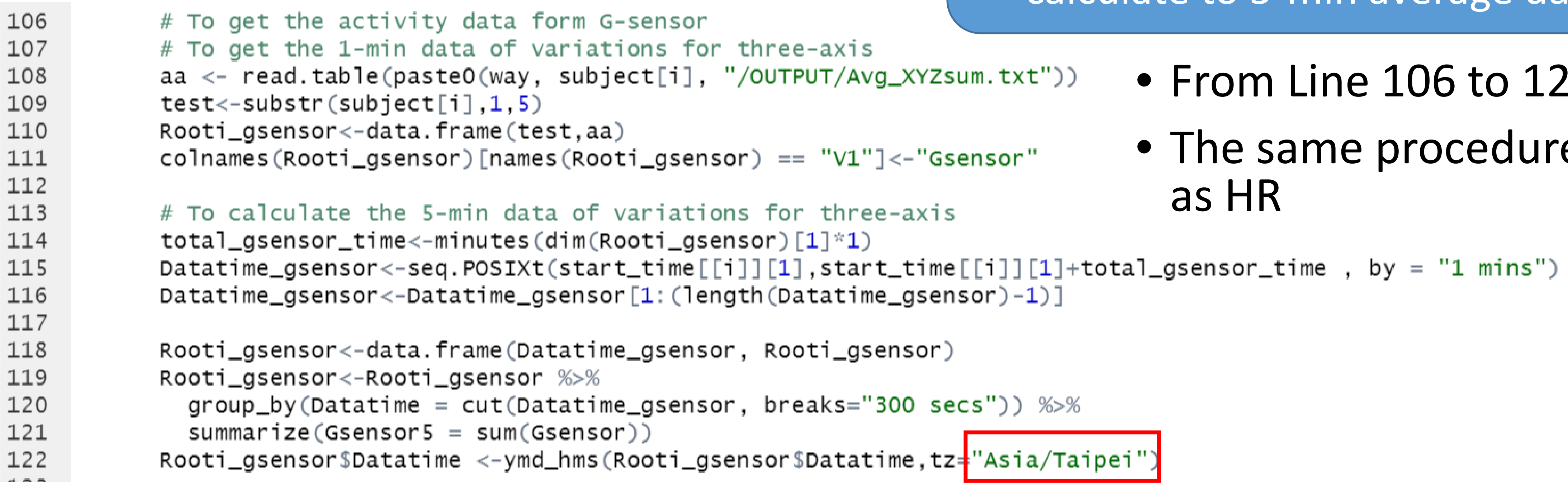
```
    # To get the 1-min HR data
    aa <- read.table(paste0(way, subject[i], "/OUTPUT/HR_full.txt"))
    test<-substr(subject[i],1,5)
    Rooti_HR<-data.frame(test,aa)
    colnames(Rooti_HR)[names(Rooti_HR) == "V1"]<-"HR"

    # To calculate the 5-min HR dat
    total_HR_time<-minutes(dim(Rooti_HR)[1]*1)
    Datatime_HR<-seq.POSIXt(start_time[[i]][1],start_time[[i]][1]+total_HR_time , by = "1 mins")
    Datatime_HR<-Datatime_HR[1:(length(Datatime_HR)-1)]

    Rooti_HR<-data.frame(Datatime_HR, Rooti_HR)
    Rooti_HR<-Rooti_HR %>%
      group_by(Datatime = cut(Datatime_HR, breaks="300 secs")) %>%
      summarize(
        HRsum5 = sum(HR),
        HRmean5 = floor(mean(HR)))
    Rooti_HR$Datatime <-ymd_hms(Rooti_HR$Datatime,tz="Asia/Taipei")
```

→ You can modify by yourself

Extracting activity data (variations for three-axis)

- Activity data
  - Variations for X-, Y- and Z- axis
  - Accelerations for X-, Y- and Z- axis

15. To extract the 1-min activity data of variations for three-axis, and then calculate to 5-min average data

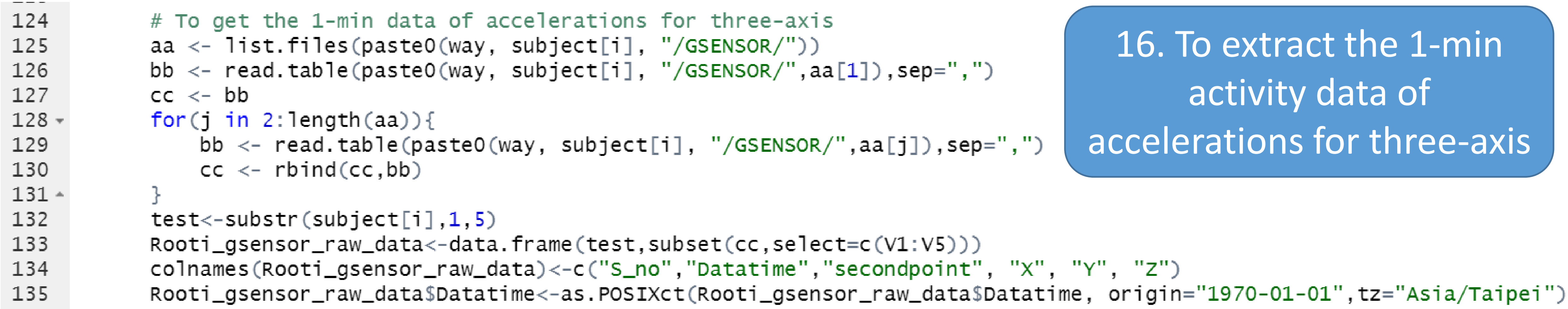


```
106        # To get the activity data form G-sensor
107        # To get the 1-min data of variations for three-axis
108        aa <- read.table(paste0(way, subject[i], "/OUTPUT/Avg_XYZsum.txt"))
109        test<-substr(subject[i],1,5)
110        Rooti_gsensor<-data.frame(test,aa)
111        colnames(Rooti_gsensor)[names(Rooti_gsensor) == "V1"]<-"Gsensor"
112
113        # To calculate the 5-min data of variations for three-axis
114        total_gsensor_time<-minutes(dim(Rooti_gsensor)[1]*1)
115        Datatime_gsensor<-seq.POSIXt(start_time[[i]][1],start_time[[i]][1]+total_gsensor_time , by = "1 mins")
116        Datatime_gsensor<-Datatime_gsensor[1:(length(Datatime_gsensor)-1)]
117
118        Rooti_gsensor<-data.frame(Datatime_gsensor, Rooti_gsensor)
119        Rooti_gsensor<-Rooti_gsensor %>%
120          group_by(Datatime = cut(Datatime_gsensor, breaks="300 secs")) %>%
121          summarize(Gsensor5 = sum(Gsensor))
122        Rooti_gsensor$Datatime <-ymd_hms(Rooti_gsensor$Datatime,tz="Asia/Taipei")
```

- From Line 106 to 122
- The same procedure as HR

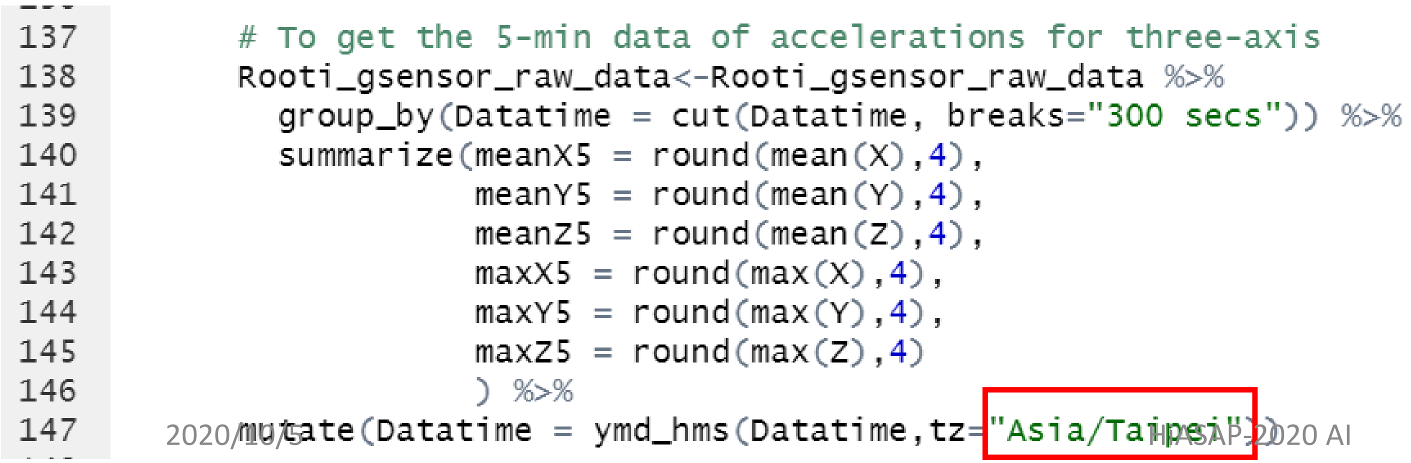
You can modify by yourself

Extracting activity data (accelerations for three-axis)



```
# To get the 1-min data of accelerations for three-axis
aa <- list.files(paste0(way, subject[i], "/GSENSOR/"))
bb <- read.table(paste0(way, subject[i], "/GSENSOR/",aa[1]),sep=",")
cc <- bb
for(j in 2:length(aa)){
    bb <- read.table(paste0(way, subject[i], "/GSENSOR/",aa[j]),sep=",")
    cc <- rbind(cc,bb)
}
test<-substr(subject[i],1,5)
Rooti_gsensor_raw_data<-data.frame(test,subset(cc,select=c(V1:V5)))
colnames(Rooti_gsensor_raw_data)<-c("S_no","Datatime","secondpoint", "X", "Y", "Z")
Rooti_gsensor_raw_data$Datatime<-as.POSIXct(Rooti_gsensor_raw_data$Datatime, origin="1970-01-01",tz="Asia/Taipei")
```

16. To extract the 1-min activity data of accelerations for three-axis



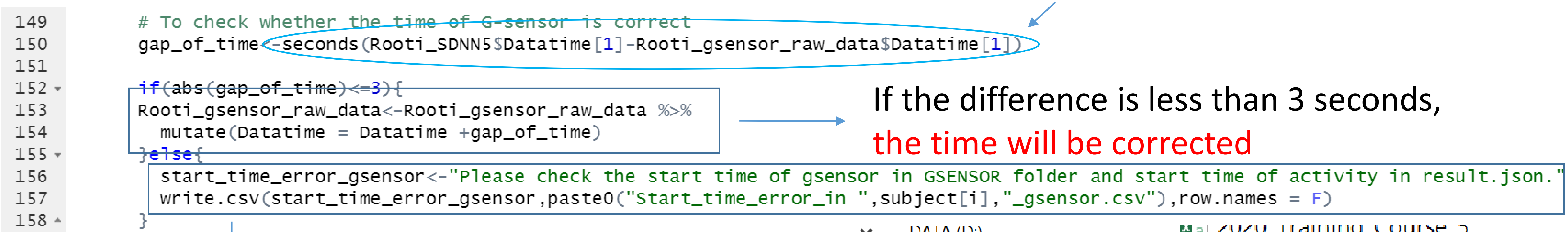
```
# To get the 5-min data of accelerations for three-axis
Rooti_gsensor_raw_data<-Rooti_gsensor_raw_data %>%
  group_by(Datatime = cut(Datatime, breaks="300 secs")) %>%
  summarize(meanX5 = round(mean(X),4),
            meanY5 = round(mean(Y),4),
            meanZ5 = round(mean(Z),4),
            maxX5 = round(max(X),4),
            maxY5 = round(max(Y),4),
            maxZ5 = round(max(Z),4)
            ) %>%
mutate(Datatime = ymd_hms(Datatime,tz="Asia/Taipei"))
```

17. To calculate the mean and maximum of accelerations for three-axis for 5-min intervals

Extracting activity data (accelerations for three-axis)

- Because the start time may be different between G-sensor and HRV monitoring ("result.json" file), the difference between two files should be less than 3 seconds.

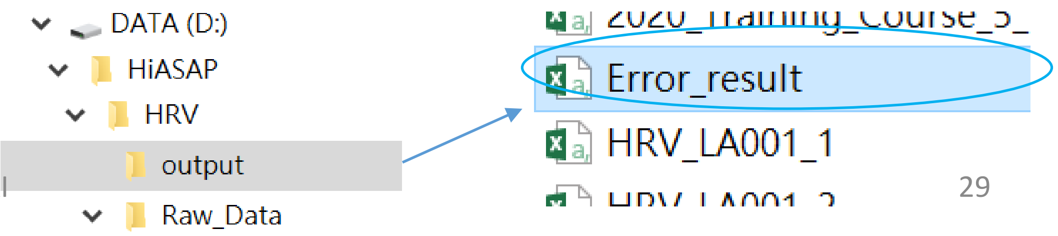
18. To determine the difference of start time between G-sensor and "result.json" file



```r
# To check whether the time of G-sensor is correct
gap_of_time<-seconds(Rooti_SDNN5$Datatime[1]-Rooti_gsensor_raw_data$Datatime[1])

if(abs(gap_of_time)<=3){
Rooti_gsensor_raw_data<-Rooti_gsensor_raw_data %>%
  mutate(Datatime = Datatime +gap_of_time)
}else{
  start_time_error_gsensor<-"Please check the start time of gsensor in GSENSOR folder and start time of activity in result.json."
  write.csv(start_time_error_gsensor,paste0("Start_time_error_in ",subject[i],"_gsensor.csv"),row.names = F)
}
```

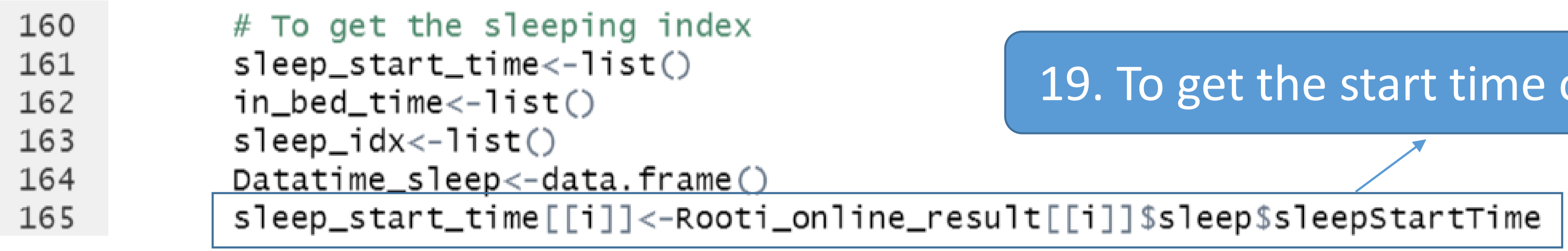
If the difference is less than 3 seconds, the time will be corrected

If the difference is more than 3 seconds, the process will be terminated



□ → You can modify by yourself

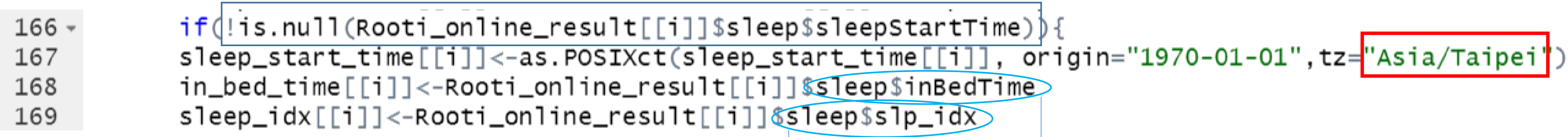
Extracting sleeping index



```
# To get the sleeping index
sleep_start_time<-list()
in_bed_time<-list()
sleep_idx<-list()
Datatime_sleep<-data.frame()
sleep_start_time[[i]]<-Rooti_online_result[[i]]$sleep$sleepStartTime
```

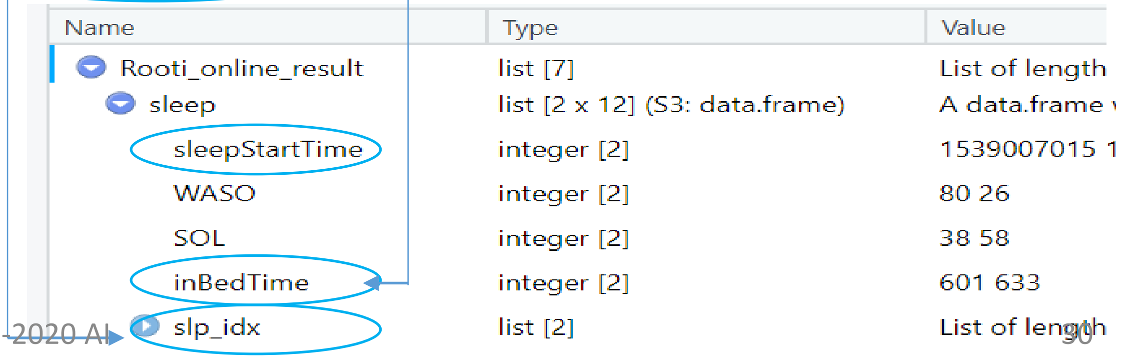
19. To get the start time of sleeping time

If subjects do not have sleeping data, it will not run the following code of sleeping index



```
if(!is.null(Rooti_online_result[[i]]$sleep$sleepStartTime)){
sleep_start_time[[i]]<-as.POSIXct(sleep_start_time[[i]], origin="1970-01-01",tz="Asia/Taipei")
in_bed_time[[i]]<-Rooti_online_result[[i]]$sleep$inBedTime
sleep_idx[[i]]<-Rooti_online_result[[i]]$sleep$slp_idx
```

20. To get the time and sleeping index during the sleeping period

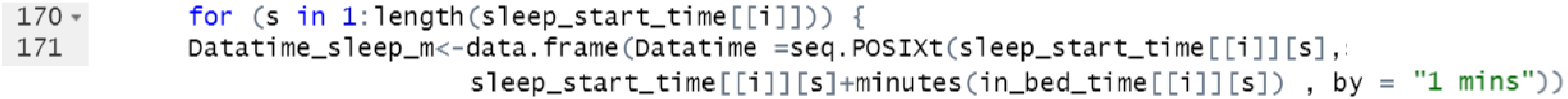


Name	Type	Value
Rooti_online_result	list [7]	List of length
sleep	list [2 x 12] (S3: data.frame)	A data.frame \
sleepStartTime	integer [2]	1539007015 1
WASO	integer [2]	80 26
SOL	integer [2]	38 58
inBedTime	integer [2]	601 633
slp_idx	list [2]	List of length

□ → You can modify by yourself

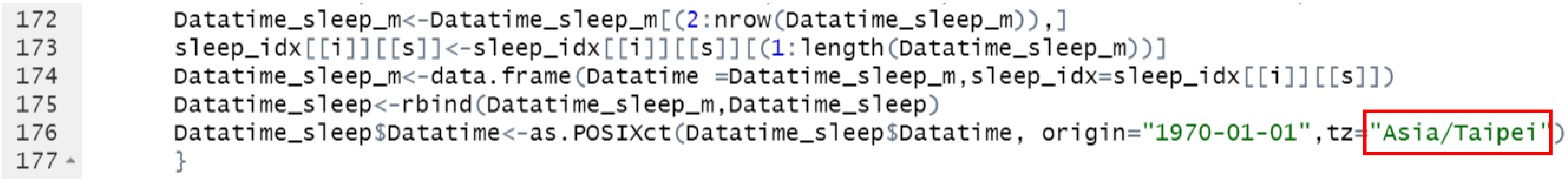
Extracting sleeping index

21. To calculated the data time of the sleeping time



```
for (s in 1:length(sleep_start_time[[i]])) {
Datatime_sleep_m<-data.frame(Datatime =seq.POSIXt(sleep_start_time[[i]][s],
                    sleep_start_time[[i]][s]+minutes(in_bed_time[[i]][s]) , by = "1 mins"))
```

22. To combine the sleeping index with data time



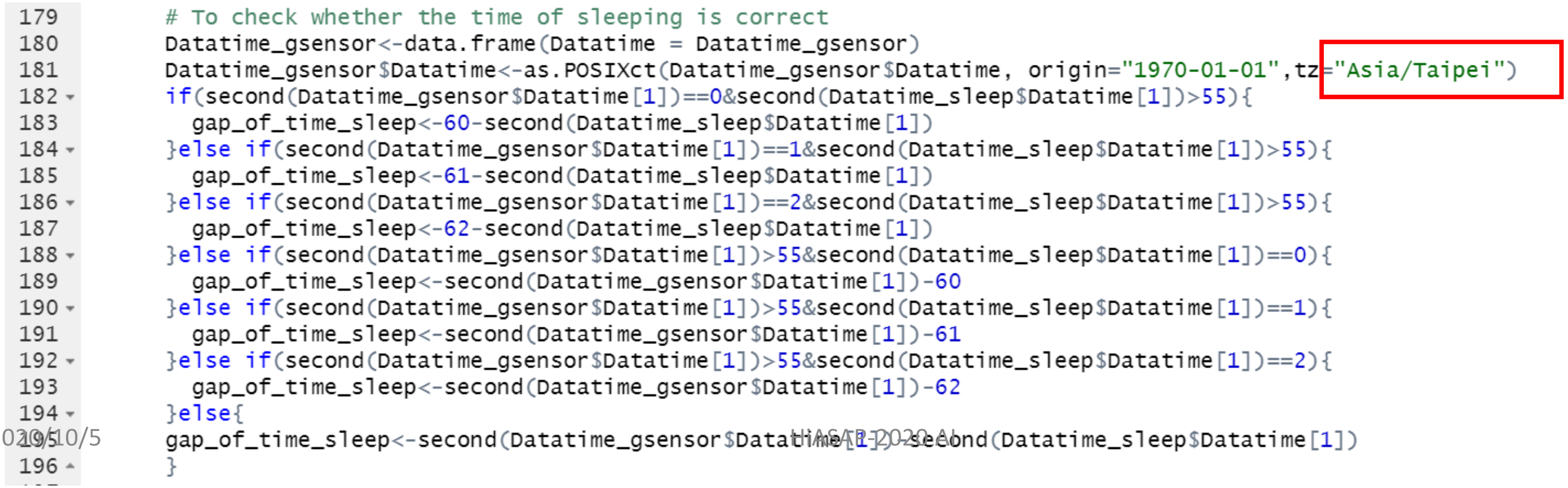
```
Datatime_sleep_m<-Datatime_sleep_m[(2:nrow(Datatime_sleep_m)),]
sleep_idx[[i]][[s]]<-sleep_idx[[i]][[s]][(1:length(Datatime_sleep_m))]
Datatime_sleep_m<-data.frame(Datatime =Datatime_sleep_m,sleep_idx=sleep_idx[[i]][[s]])
Datatime_sleep<-rbind(Datatime_sleep_m,Datatime_sleep)
Datatime_sleep$Datatime<-as.POSIXct(Datatime_sleep$Datatime, origin="1970-01-01",tz="Asia/Taipei")
}
```

□ → You can modify by yourself

Extracting sleeping index

- Because the start time may be also different between sleeping data and G-sensor data, the difference between two files should be less than 3 seconds.

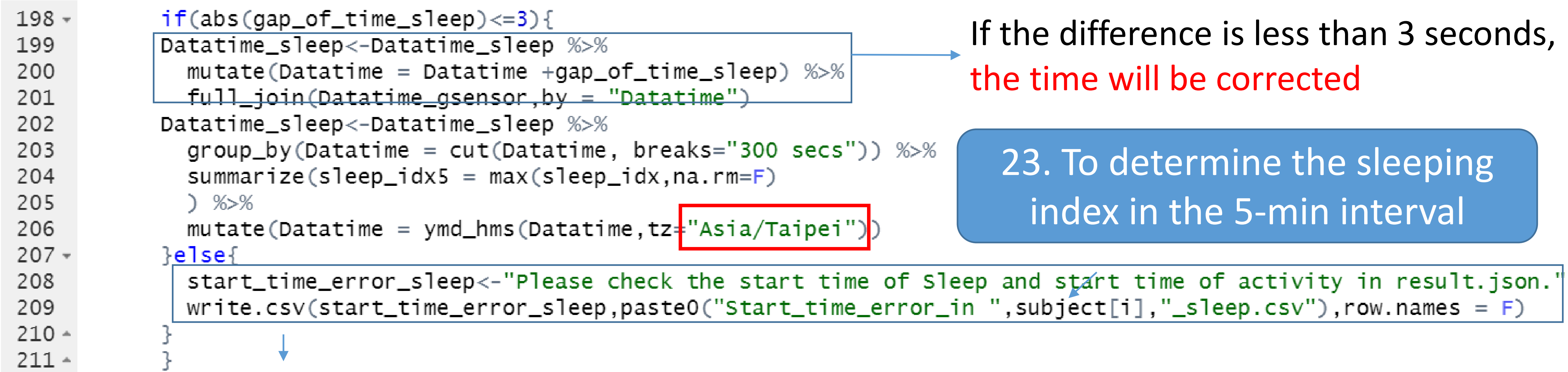
22. To determine the difference of start time between sleeping data and G-sensor data



```r
# To check whether the time of sleeping is correct
Datatime_gsensor<-data.frame(Datatime = Datatime_gsensor)
Datatime_gsensor$Datatime<-as.POSIXct(Datatime_gsensor$Datatime, origin="1970-01-01",tz="Asia/Taipei")
if(second(Datatime_gsensor$Datatime[1])==0&second(Datatime_sleep$Datatime[1])>55){
  gap_of_time_sleep<-60-second(Datatime_sleep$Datatime[1])
}else if(second(Datatime_gsensor$Datatime[1])==1&second(Datatime_sleep$Datatime[1])>55){
  gap_of_time_sleep<-61-second(Datatime_sleep$Datatime[1])
}else if(second(Datatime_gsensor$Datatime[1])==2&second(Datatime_sleep$Datatime[1])>55){
  gap_of_time_sleep<-62-second(Datatime_sleep$Datatime[1])
}else if(second(Datatime_gsensor$Datatime[1])>55&second(Datatime_sleep$Datatime[1])==0){
  gap_of_time_sleep<-second(Datatime_gsensor$Datatime[1])-60
}else if(second(Datatime_gsensor$Datatime[1])>55&second(Datatime_sleep$Datatime[1])==1){
  gap_of_time_sleep<-second(Datatime_gsensor$Datatime[1])-61
}else if(second(Datatime_gsensor$Datatime[1])>55&second(Datatime_sleep$Datatime[1])==2){
  gap_of_time_sleep<-second(Datatime_gsensor$Datatime[1])-62
}else{
gap_of_time_sleep<-second(Datatime_gsensor$Datatime[1])-second(Datatime_sleep$Datatime[1])
}
```

□ → You can modify by yourself

Extracting sleeping index

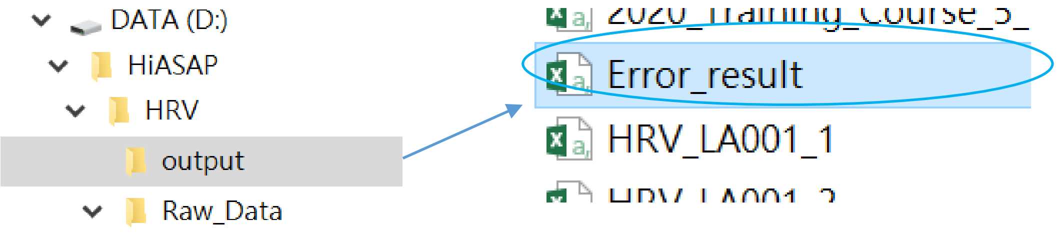


```
if(abs(gap_of_time_sleep)<=3){
Datatime_sleep<-Datatime_sleep %>%
  mutate(Datatime = Datatime +gap_of_time_sleep) %>%
  full_join(Datatime_gsensor,by = "Datatime")
Datatime_sleep<-Datatime_sleep %>%
  group_by(Datatime = cut(Datatime, breaks="300 secs")) %>%
  summarize(sleep_idx5 = max(sleep_idx,na.rm=F)
  ) %>%
  mutate(Datatime = ymd_hms(Datatime,tz="Asia/Taipei"))
}else{
  start_time_error_sleep<-"Please check the start time of Sleep and start time of activity in result.json."
  write.csv(start_time_error_sleep,paste0("Start_time_error_in ",subject[i],"_sleep.csv"),row.names = F)
}
}
```

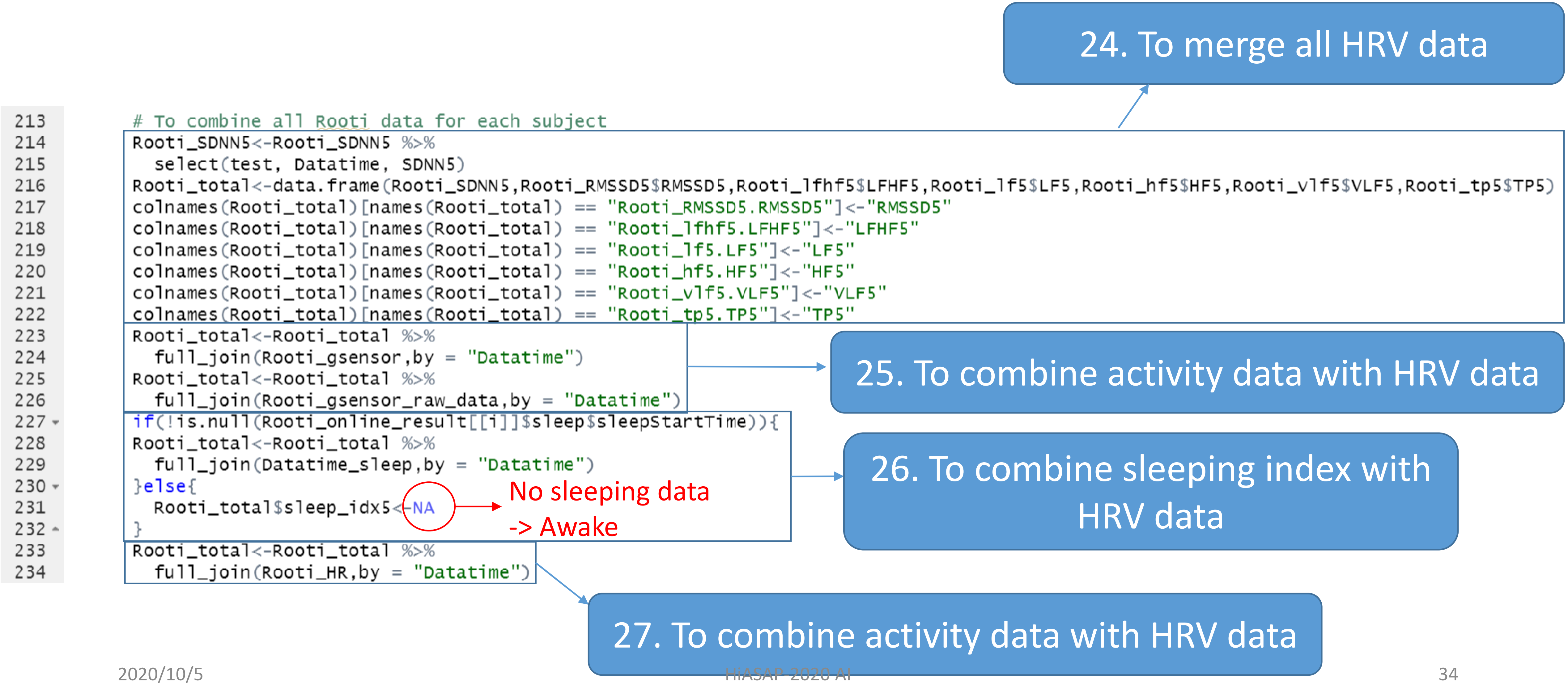
If the difference is less than 3 seconds, the time will be corrected

23. To determine the sleeping index in the 5-min interval

If the difference is more than 3 seconds, the process will be terminated



Data combination for each subject



24. To merge all HRV data

```
# To combine all Rooti data for each subject
Rooti_SDNN5<-Rooti_SDNN5 %>%
  select(test, Datatime, SDNN5)
Rooti_total<-data.frame(Rooti_SDNN5,Rooti_RMSSD5$RMSSD5,Rooti_lfhf5$LFHF5,Rooti_lf5$LF5,Rooti_hf5$HF5,Rooti_vlf5$VLF5,Rooti_tp5$TP5)
colnames(Rooti_total)[names(Rooti_total) == "Rooti_RMSSD5.RMSSD5"]<-"RMSSD5"
colnames(Rooti_total)[names(Rooti_total) == "Rooti_lfhf5.LFHF5"]<-"LFHF5"
colnames(Rooti_total)[names(Rooti_total) == "Rooti_lf5.LF5"]<-"LF5"
colnames(Rooti_total)[names(Rooti_total) == "Rooti_hf5.HF5"]<-"HF5"
colnames(Rooti_total)[names(Rooti_total) == "Rooti_vlf5.VLF5"]<-"VLF5"
colnames(Rooti_total)[names(Rooti_total) == "Rooti_tp5.TP5"]<-"TP5"
Rooti_total<-Rooti_total %>%
  full_join(Rooti_gsensor,by = "Datatime")
Rooti_total<-Rooti_total %>%
  full_join(Rooti_gsensor_raw_data,by = "Datatime")
if(!is.null(Rooti_online_result[[i]]$sleep$sleepStartTime)){
Rooti_total<-Rooti_total %>%
  full_join(Datatime_sleep,by = "Datatime")
}else{
  Rooti_total$sleep_idx5<-NA
}
Rooti_total<-Rooti_total %>%
  full_join(Rooti_HR,by = "Datatime")
```

25. To combine activity data with HRV data

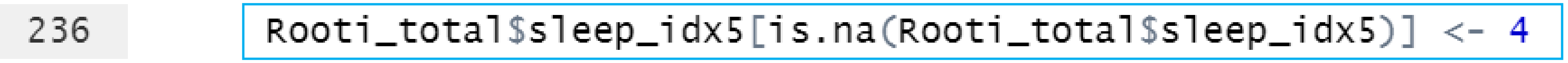
26. To combine sleeping index with HRV data

No sleeping data -> Awake

27. To combine activity data with HRV data

Data combination for each subject

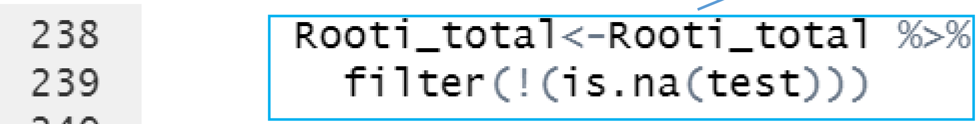
28. To re-code the “NA” as “4” for sleeping index



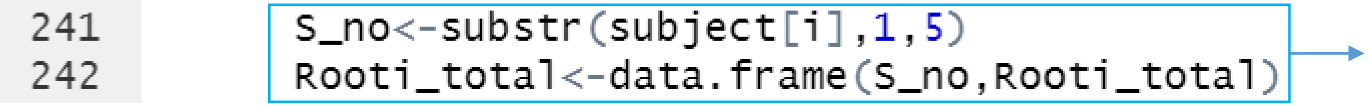
```
Rooti_total$sleep_idx5[is.na(Rooti_total$sleep_idx5)] <- 4
```

Sleeping index	Sleeping status
1	Deep sleep
2	Light sleep
3	Rapid Eye Movement (REM)
4	Awake

29. To exclude the time without HRV data

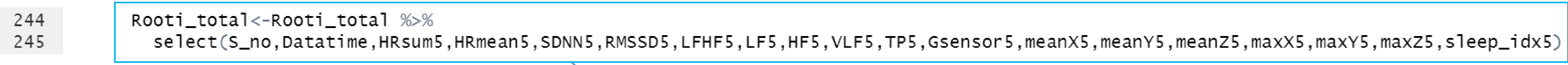


```
Rooti_total<-Rooti_total %>%
  filter(!(is.na(test)))
```



```
S_no<-substr(subject[i],1,5)
Rooti_total<-data.frame(S_no,Rooti_total)
```

30. To add the variable of subjects’ ID to the HRV data

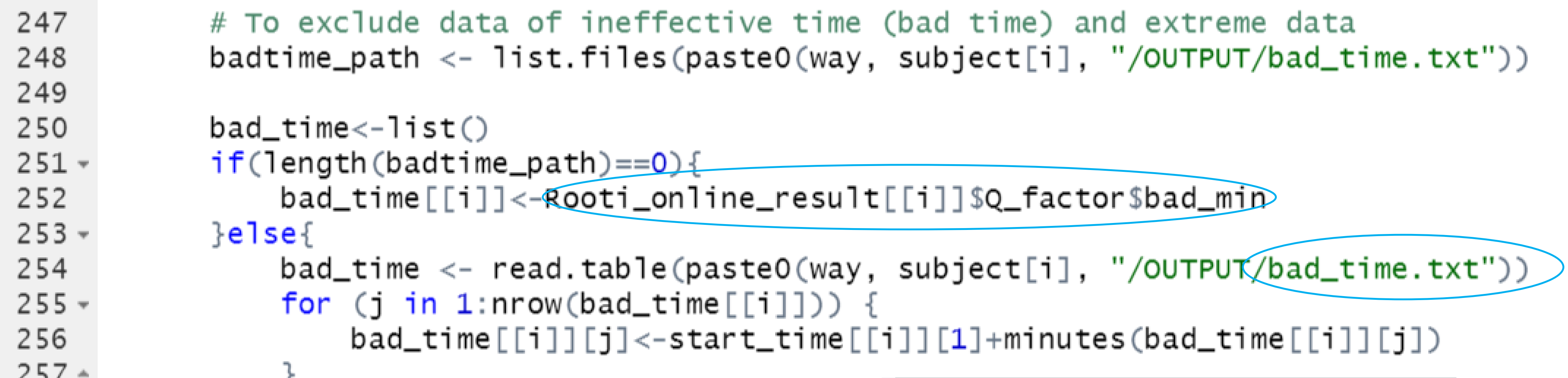


```
Rooti_total<-Rooti_total %>%
  select(S_no,Datatime,HRsum5,HRmean5,SDNN5,RMSSD5,LFHF5,LF5,HF5,VLF5,TP5,Gsensor5,meanX5,meanY5,meanZ5,maxX5,maxY5,maxZ5,sleep_idx5)
```

31. To select variables which will use in the following analysis

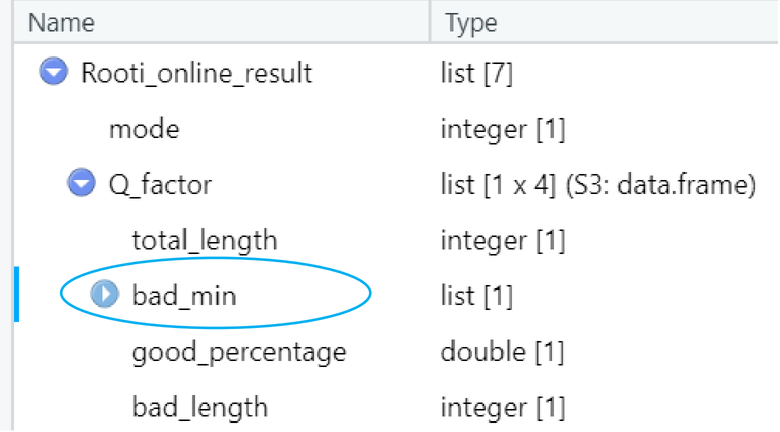
Excluding the ineffective time (bad time)

32. To read the bad time of HRV monitoring



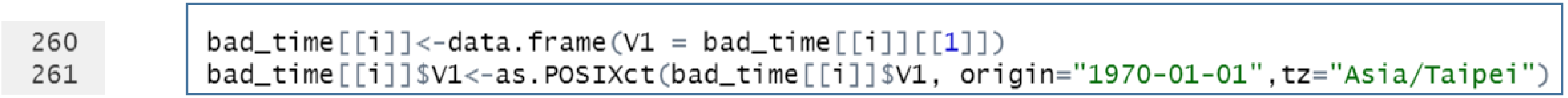
```r
        # To exclude data of ineffective time (bad time) and extreme data
        badtime_path <- list.files(paste0(way, subject[i], "/OUTPUT/bad_time.txt"))

        bad_time<-list()
        if(length(badtime_path)==0){
            bad_time[[i]]<-Rooti_online_result[[i]]$Q_factor$bad_min
        }else{
            bad_time <- read.table(paste0(way, subject[i], "/OUTPUT/bad_time.txt"))
            for (j in 1:nrow(bad_time[[i]])) {
                bad_time[[i]][j]<-start_time[[i]][1]+minutes(bad_time[[i]][j])
            }
        }
```

To automatically select the bad time from “bad_time.txt” file or “result.json” file



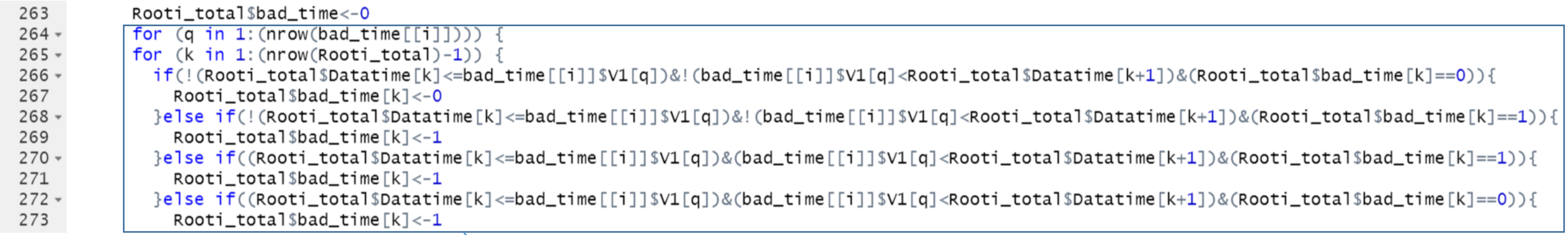
Name	Type
Rooti_online_result	list [7]
mode	integer [1]
Q_factor	list [1 x 4] (S3: data.frame)
total_length	integer [1]
bad_min	list [1]
good_percentage	double [1]
bad_length	integer [1]

Excluding the ineffective time (bad time)



```r
bad_time[[i]]<-data.frame(V1 = bad_time[[i]][[1]])
bad_time[[i]]$V1<-as.POSIXct(bad_time[[i]]$V1, origin="1970-01-01",tz="Asia/Taipei")
```

33. To combine bad time to HRV data

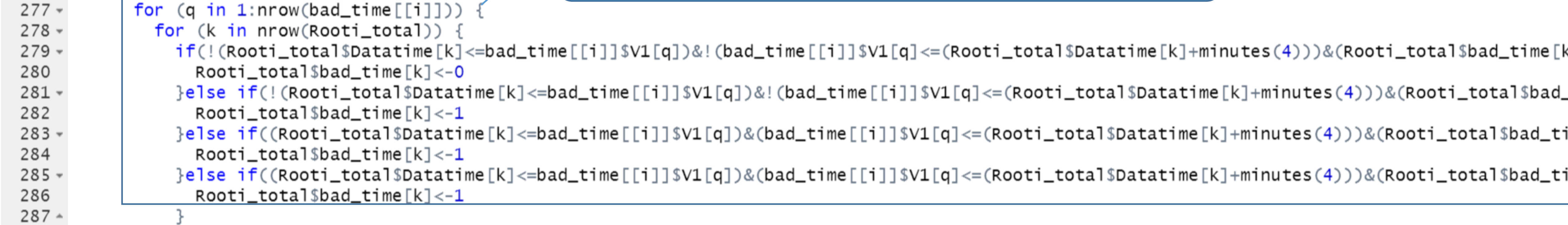


```r
Rooti_total$bad_time<-0
for (q in 1:(nrow(bad_time[[i]]))) {
for (k in 1:(nrow(Rooti_total)-1)) {
  if(!(Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&!(bad_time[[i]]$V1[q]<Rooti_total$Datatime[k+1])&(Rooti_total$bad_time[k]==0)){
    Rooti_total$bad_time[k]<-0
  }else if(!(Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&!(bad_time[[i]]$V1[q]<Rooti_total$Datatime[k+1])&(Rooti_total$bad_time[k]==1)){
    Rooti_total$bad_time[k]<-1
  }else if((Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&(bad_time[[i]]$V1[q]<Rooti_total$Datatime[k+1])&(Rooti_total$bad_time[k]==1)){
    Rooti_total$bad_time[k]<-1
  }else if((Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&(bad_time[[i]]$V1[q]<Rooti_total$Datatime[k+1])&(Rooti_total$bad_time[k]==0)){
    Rooti_total$bad_time[k]<-1
```

34. To determine whether the data time is bad time in minute

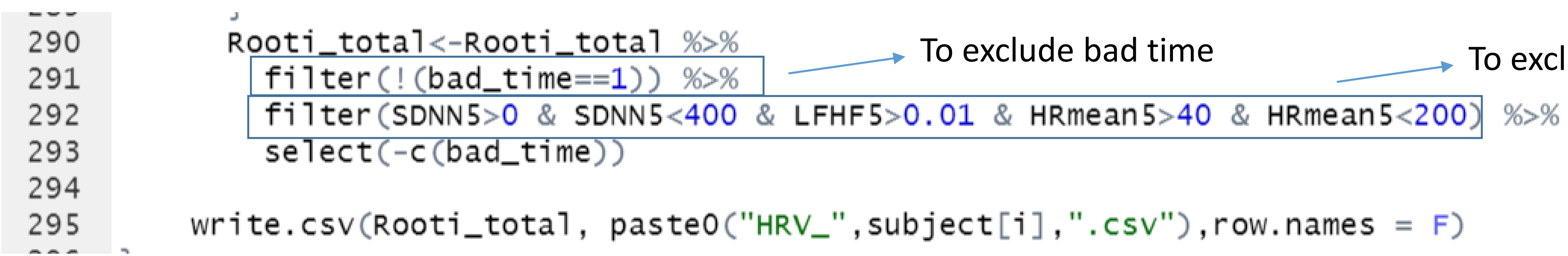
Excluding the ineffective time (bad time) and

35. To determine whether the data time is bad time in 5-min interval



```
for (q in 1:nrow(bad_time[[i]])) {
  for (k in nrow(Rooti_total)) {
    if(!(Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&!(bad_time[[i]]$V1[q]<=(Rooti_total$Datatime[k]+minutes(4)))&(Rooti_total$bad_time[k
      Rooti_total$bad_time[k]<-0
    }else if(!(Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&!(bad_time[[i]]$V1[q]<=(Rooti_total$Datatime[k]+minutes(4)))&(Rooti_total$bad_
      Rooti_total$bad_time[k]<-1
    }else if((Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&(bad_time[[i]]$V1[q]<=(Rooti_total$Datatime[k]+minutes(4)))&(Rooti_total$bad_ti
      Rooti_total$bad_time[k]<-1
    }else if((Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&(bad_time[[i]]$V1[q]<=(Rooti_total$Datatime[k]+minutes(4)))&(Rooti_total$bad_ti
      Rooti_total$bad_time[k]<-1
    }
```

36. To exclude the 5-min intervals contained bad time and the data with abnormal signals



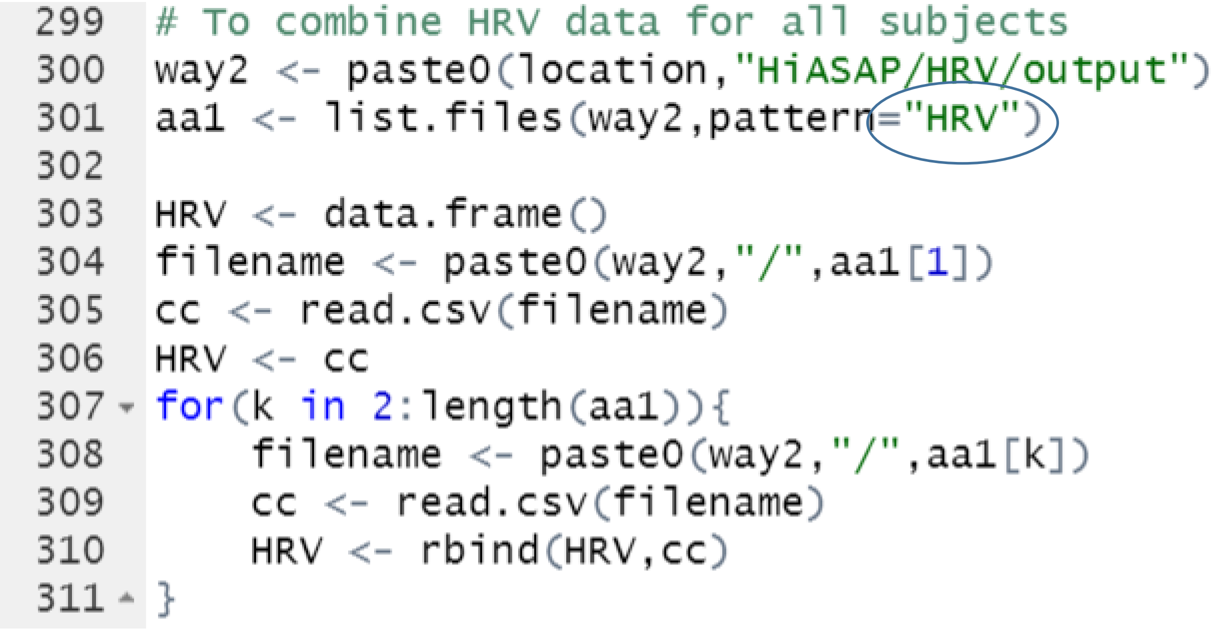
```
   Rooti_total<-Rooti_total %>%
    filter(!(bad_time==1)) %>%
    filter(SDNN5>0 & SDNN5<400 & LFHF5>0.01 & HRmean5>40 & HRmean5<200) %>%
    select(-c(bad_time))

  write.csv(Rooti_total, paste0("HRV_",subject[i],".csv"),row.names = F)
```

To exclude bad time

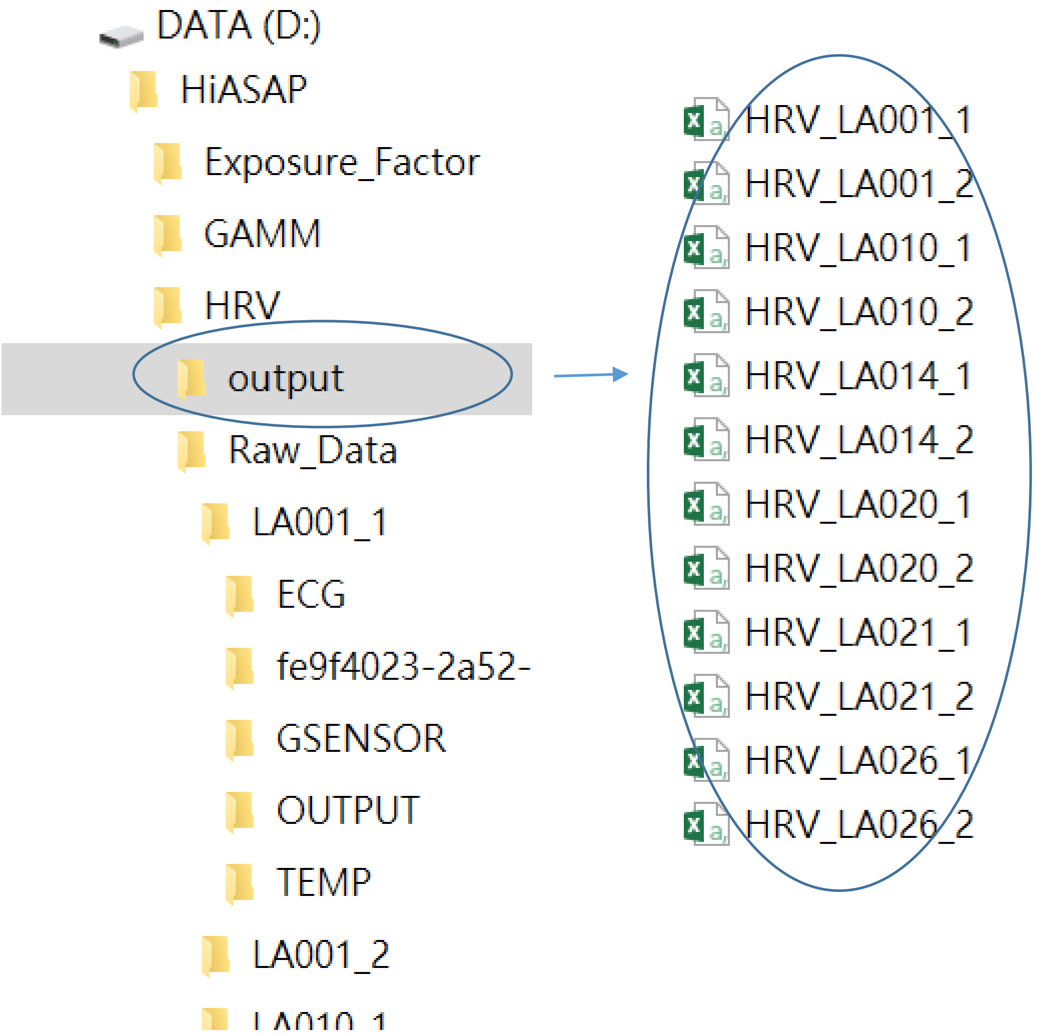
To exclude abnormal signals

Data combination for all subjects

37. To combine HRV data for all subjects



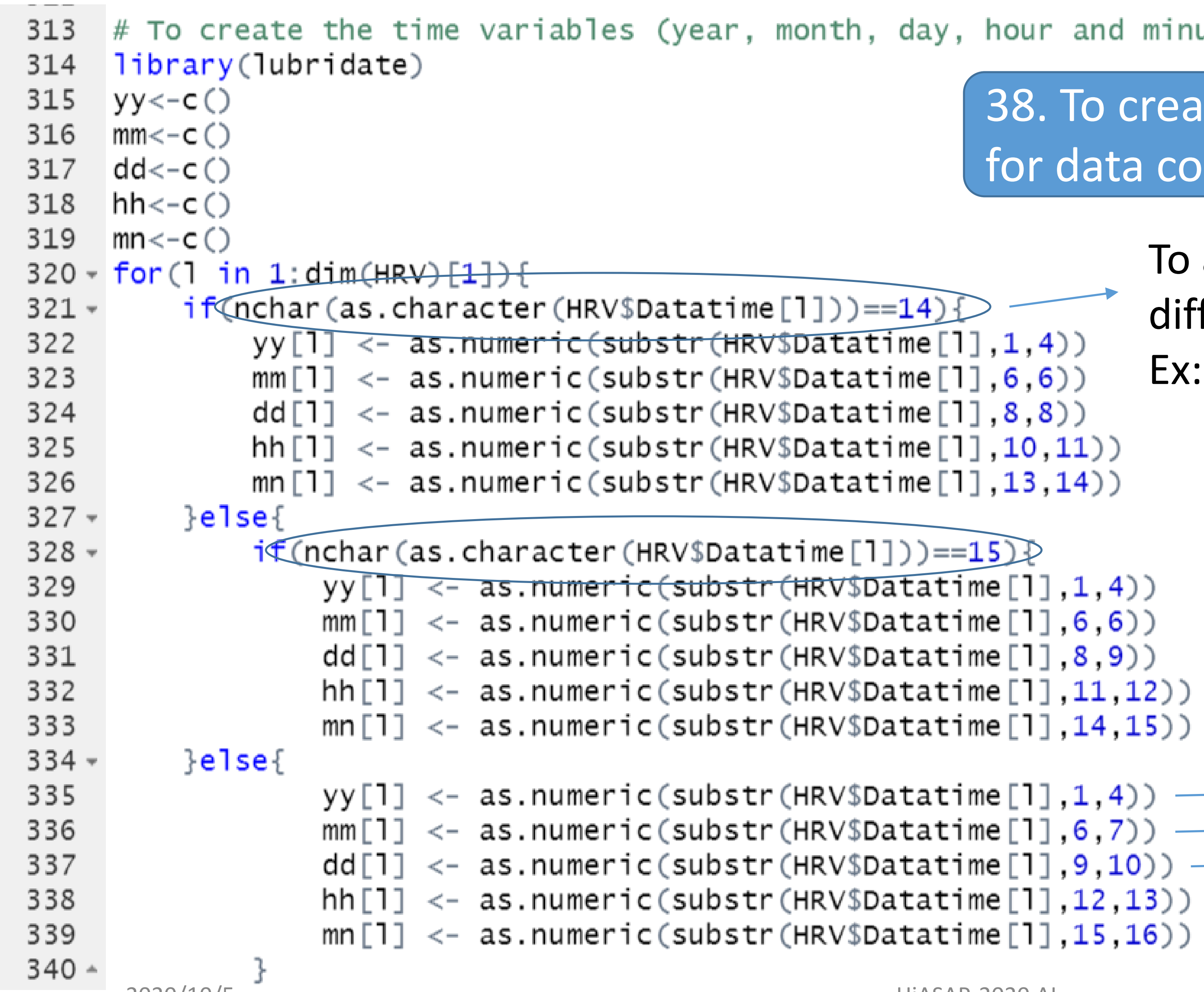
```r
# To combine HRV data for all subjects
way2 <- paste0(location,"HiASAP/HRV/output")
aa1 <- list.files(way2,pattern="HRV")

HRV <- data.frame()
filename <- paste0(way2,"/",aa1[1])
cc <- read.csv(filename)
HRV <- cc
for(k in 2:length(aa1)){
    filename <- paste0(way2,"/",aa1[k])
    cc <- read.csv(filename)
    HRV <- rbind(HRV,cc)
}
```



DATA (D:)
HiASAP
Exposure_Factor
GAMM
HRV
output
Raw_Data
LA001_1
ECG
fe9f4023-2a52-
GSENSOR
OUTPUT
TEMP
LA001_2
LA010_1

HRV_LA001_1
HRV_LA001_2
HRV_LA010_1
HRV_LA010_2
HRV_LA014_1
HRV_LA014_2
HRV_LA020_1
HRV_LA020_2
HRV_LA021_1
HRV_LA021_2
HRV_LA026_1
HRV_LA026_2

38. To create the time-related variables for data combination

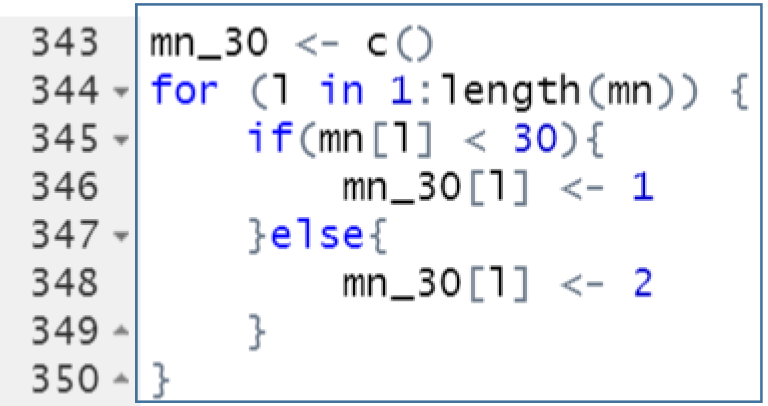


```r
# To create the time variables (year, month, day, hour and minute) for the following data matching
library(lubridate)
yy<-c()
mm<-c()
dd<-c()
hh<-c()
mn<-c()
for(l in 1:dim(HRV)[1]){
    if(nchar(as.character(HRV$Datatime[l]))==14){
        yy[l] <- as.numeric(substr(HRV$Datatime[l],1,4))
        mm[l] <- as.numeric(substr(HRV$Datatime[l],6,6))
        dd[l] <- as.numeric(substr(HRV$Datatime[l],8,8))
        hh[l] <- as.numeric(substr(HRV$Datatime[l],10,11))
        mn[l] <- as.numeric(substr(HRV$Datatime[l],13,14))
    }else{
        if(nchar(as.character(HRV$Datatime[l]))==15){
            yy[l] <- as.numeric(substr(HRV$Datatime[l],1,4))
            mm[l] <- as.numeric(substr(HRV$Datatime[l],6,6))
            dd[l] <- as.numeric(substr(HRV$Datatime[l],8,9))
            hh[l] <- as.numeric(substr(HRV$Datatime[l],11,12))
            mn[l] <- as.numeric(substr(HRV$Datatime[l],14,15))
    }else{
            yy[l] <- as.numeric(substr(HRV$Datatime[l],1,4))
            mm[l] <- as.numeric(substr(HRV$Datatime[l],6,7))
            dd[l] <- as.numeric(substr(HRV$Datatime[l],9,10))
            hh[l] <- as.numeric(substr(HRV$Datatime[l],12,13))
            mn[l] <- as.numeric(substr(HRV$Datatime[l],15,16))
        }
```

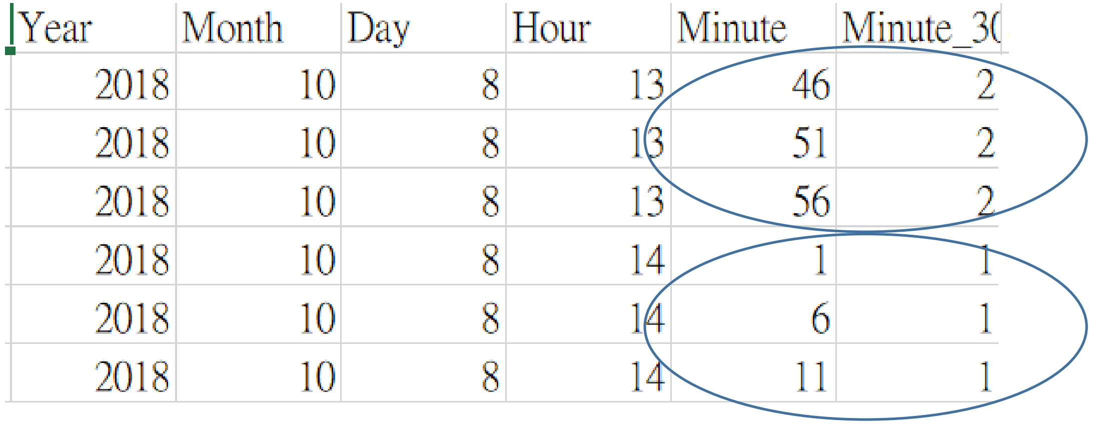
To avoid the formats of date may be different
Ex: 2020-03-03 / 2020-3-10 / 2020-3-1 ….

Year
Month
Day
Hour
Minute

39. To create a variable of 30-minute interval of each hour for merging data with TAD



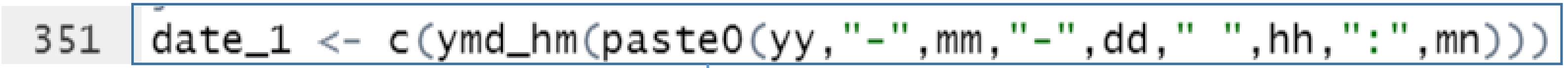
```r
mn_30 <- c()
for (l in 1:length(mn)) {
    if(mn[l] < 30){
        mn_30[l] <- 1
    }else{
        mn_30[l] <- 2
    }
}
```



Year	Month	Day	Hour	Minute	Minute_3(
2018	10	8	13	46	2
2018	10	8	13	51	2
2018	10	8	13	56	2
2018	10	8	14	1	1
2018	10	8	14	6	1
2018	10	8	14	11	1

Time between 30 to 59 minutes -> 2

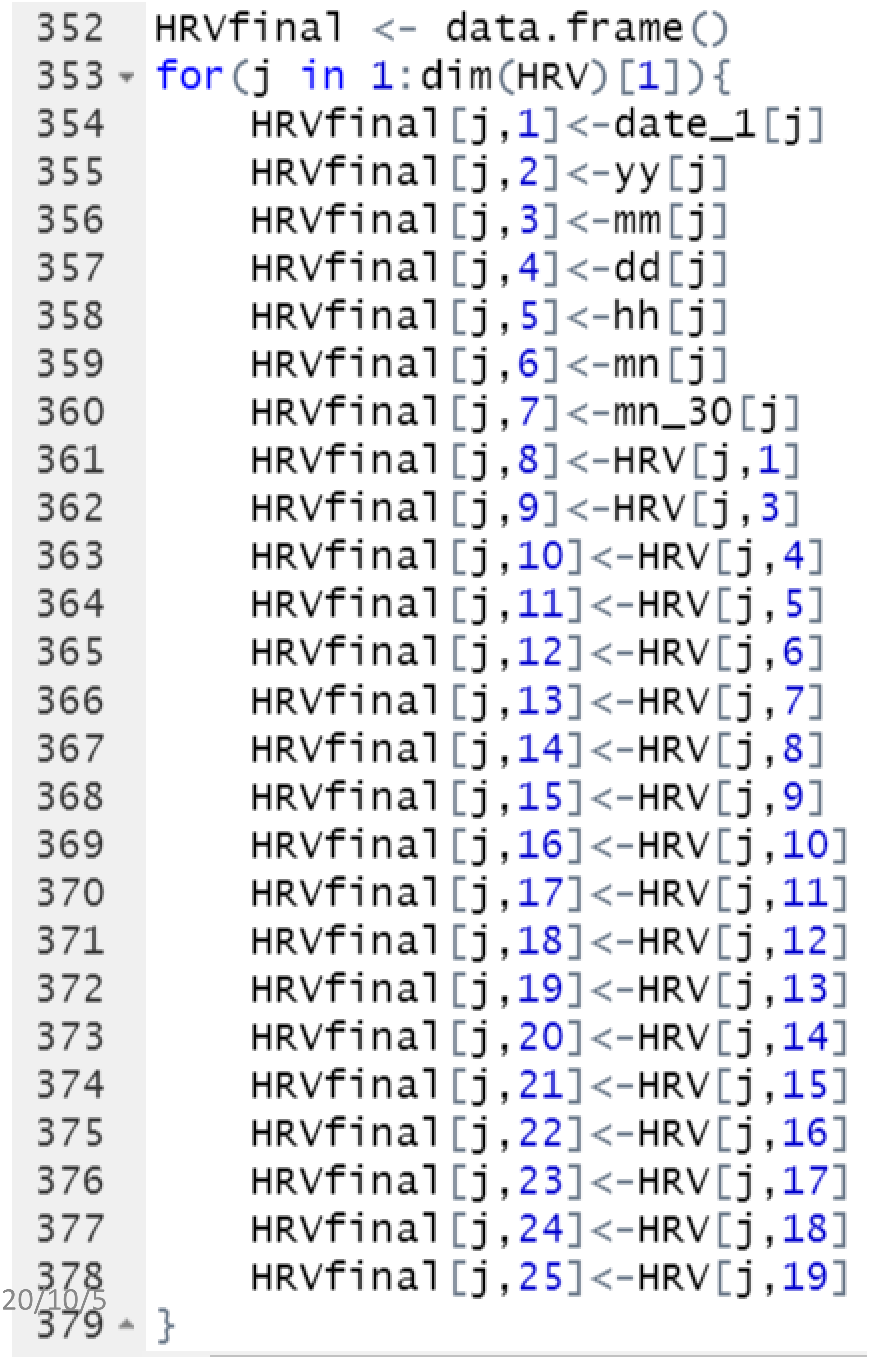
Time between 0 to 29 minutes -> 1



```r
date_1 <- c(ymd_hm(paste0(yy,"-",mm,"-",dd," ",hh,":",mn)))
```

40. To format the date variable

Ex: 2020-01-01 14:11

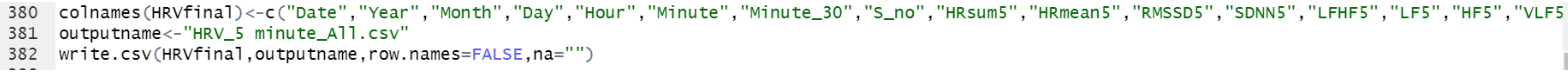


```
HRVfinal <- data.frame()
for(j in 1:dim(HRV)[1]){
      HRVfinal[j,1]<-date_1[j]
      HRVfinal[j,2]<-yy[j]
      HRVfinal[j,3]<-mm[j]
      HRVfinal[j,4]<-dd[j]
      HRVfinal[j,5]<-hh[j]
      HRVfinal[j,6]<-mn[j]
      HRVfinal[j,7]<-mn_30[j]
      HRVfinal[j,8]<-HRV[j,1]
      HRVfinal[j,9]<-HRV[j,3]
      HRVfinal[j,10]<-HRV[j,4]
      HRVfinal[j,11]<-HRV[j,5]
      HRVfinal[j,12]<-HRV[j,6]
      HRVfinal[j,13]<-HRV[j,7]
      HRVfinal[j,14]<-HRV[j,8]
      HRVfinal[j,15]<-HRV[j,9]
      HRVfinal[j,16]<-HRV[j,10]
      HRVfinal[j,17]<-HRV[j,11]
      HRVfinal[j,18]<-HRV[j,12]
      HRVfinal[j,19]<-HRV[j,13]
      HRVfinal[j,20]<-HRV[j,14]
      HRVfinal[j,21]<-HRV[j,15]
      HRVfinal[j,22]<-HRV[j,16]
      HRVfinal[j,23]<-HRV[j,17]
      HRVfinal[j,24]<-HRV[j,18]
      HRVfinal[j,25]<-HRV[j,19]
}
```

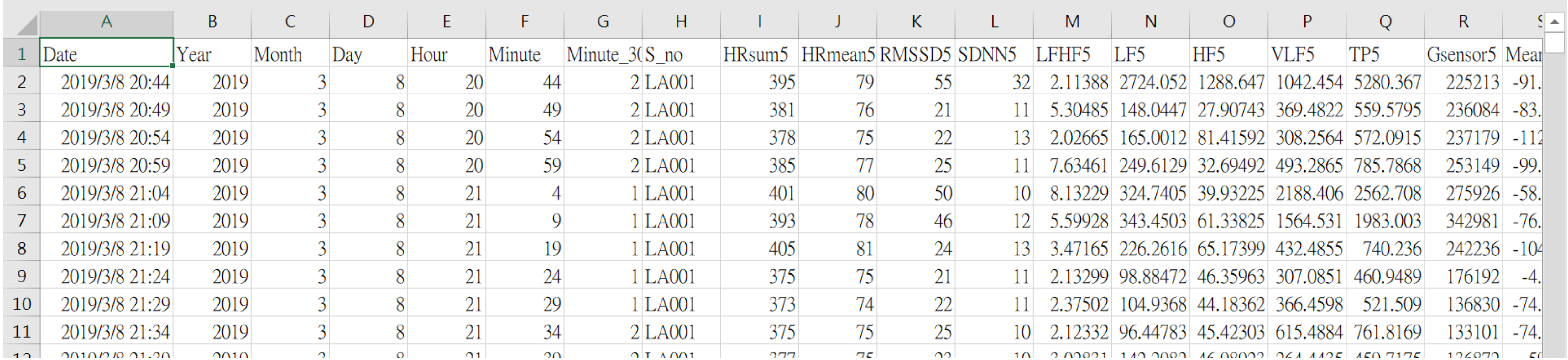
41. To combine time-related variables with HRV data

Data export

42. To export the final dataset of HRV data for all subject



```
colnames(HRVfinal)<-c("Date","Year","Month","Day","Hour","Minute","Minute_30","S_no","HRsum5","HRmean5","RMSSD5","SDNN5","LFHF5","LF5","HF5","VLF5
outputname<-"HRV_5 minute_All.csv"
write.csv(HRVfinal,outputname,row.names=FALSE,na="")
```



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Date	Year	Month	Day	Hour	Minute	Minute_3(	S_no	HRsum5	HRmean5	RMSSD5	SDNN5	LFHF5	LF5	HF5	VLF5	TP5	Gsensor5	Mear
2	2019/3/8 20:44	2019	3	8	20	44	2	LA001	395	79	55	32	2.11388	2724.052	1288.647	1042.454	5280.367	225213	-91.
3	2019/3/8 20:49	2019	3	8	20	49	2	LA001	381	76	21	11	5.30485	148.0447	27.90743	369.4822	559.5795	236084	-83.
4	2019/3/8 20:54	2019	3	8	20	54	2	LA001	378	75	22	13	2.02665	165.0012	81.41592	308.2564	572.0915	237179	-112
5	2019/3/8 20:59	2019	3	8	20	59	2	LA001	385	77	25	11	7.63461	249.6129	32.69492	493.2865	785.7868	253149	-99.
6	2019/3/8 21:04	2019	3	8	21	4	1	LA001	401	80	50	10	8.13229	324.7405	39.93225	2188.406	2562.708	275926	-58.
7	2019/3/8 21:09	2019	3	8	21	9	1	LA001	393	78	46	12	5.59928	343.4503	61.33825	1564.531	1983.003	342981	-76.
8	2019/3/8 21:19	2019	3	8	21	19	1	LA001	405	81	24	13	3.47165	226.2616	65.17399	432.4855	740.236	242236	-104
9	2019/3/8 21:24	2019	3	8	21	24	1	LA001	375	75	21	11	2.13299	98.88472	46.35963	307.0851	460.9489	176192	-4.
10	2019/3/8 21:29	2019	3	8	21	29	1	LA001	373	74	22	11	2.37502	104.9368	44.18362	366.4598	521.509	136830	-74.
11	2019/3/8 21:34	2019	3	8	21	34	2	LA001	375	75	25	10	2.12332	96.44783	45.42303	615.4884	761.8169	133101	-74.

Thank you for your attention