



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

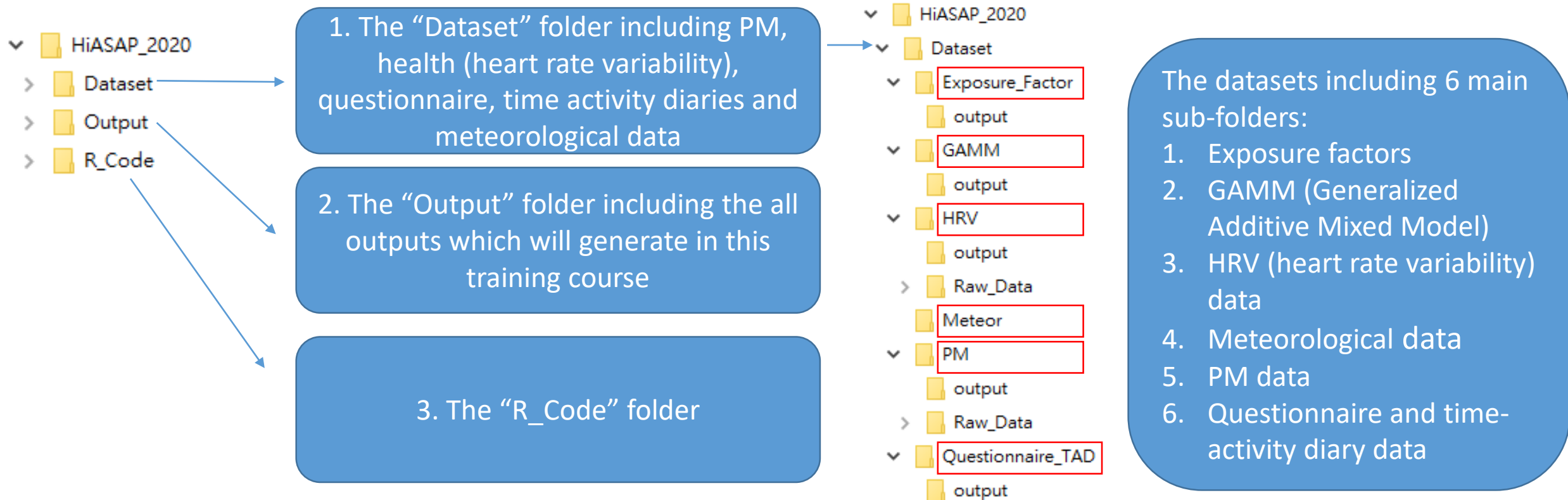
Part 2:
Introduction
of RStudio

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

Part 1:

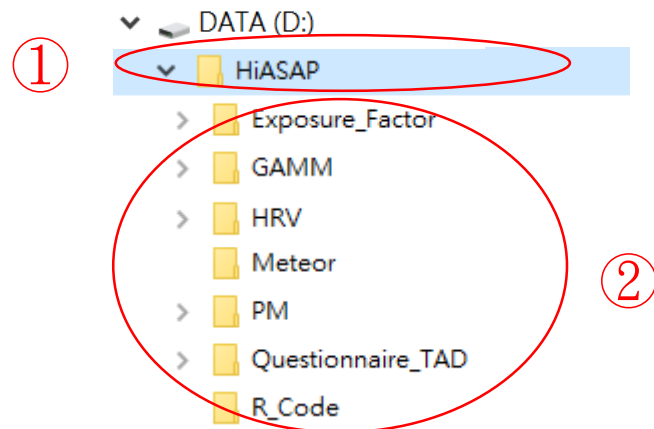
Introduction of dataset

Datasets



Default path of data files in this course

- In order to make sure you can use the R codes provided to analyze the data directly, please follow the steps below:
 1. To create a “HiASAP” folder in D drive
 2. To move all datasets provided to the “HiASAP” folder

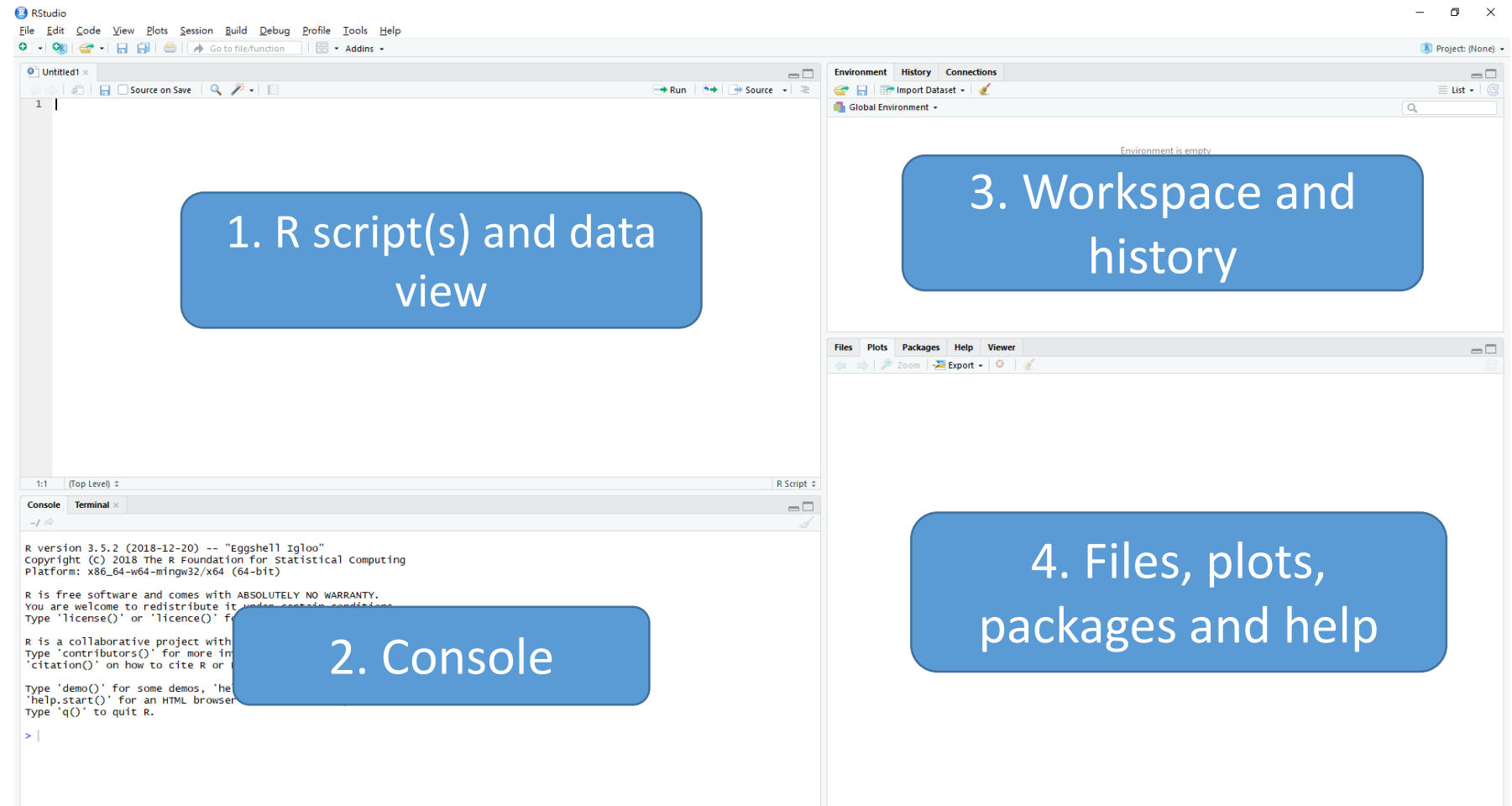


Part 2:

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 (upper left window)

```
QB_Combine.R x 2019_BQ_GAMM_PM_HRV_Awake_P... x GAMM.R* x 2020_Training_PM_by_Rooti_Time.R x 2020_Training_Rooti_HRV.R x 2020_Tra... x
Source on Save Run Source
1 #remove previous memory in R.
2 rm(list=ls())
3 # set a CRAN mirror
4 # Automatic redirection to servers worldwide, currently sponsored by Rstudio
5 local({r = getOption("repos")
6 r["CRAN"] = "https://cloud.r-project.org/"
7 options(repos=r)})
8
9 # update the packages
10 update.packages("lubridate")
11 update.packages("plyr")
12 update.packages("tidyverse")
13 update.packages("jsonlite")
14 update.packages("data.table")
15 update.packages("optparse")
16
17 # set up the path, time zone and location for the subject number
18 library(optparse)
19 opt<-parse_args(option_parser(option_list = list(make_option("--location",default = 'D:/HiA
20 make_option("--output_folder",default = 'o
21 make_option("--rawdata_folder",default = '
22 make_option("--time_zone",default = 'Asia/
23 make_option("--path_number",default = 2)#h
24 )))
25
26 #call out the packages
27 library(lubridate)
28 library(plyr)
29 library(tidyverse)
30 library(jsonlite)
31 library(data.table)
32
33 <
7:19 (Top Level) >
```

The script can be saved as a R file

To type R commands and run them

Output will appear in the Console window below.

	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

To view the data

2. Console (lower left window)

To type commands and
show the outputs

```
Console Terminal x Jobs x
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)
```

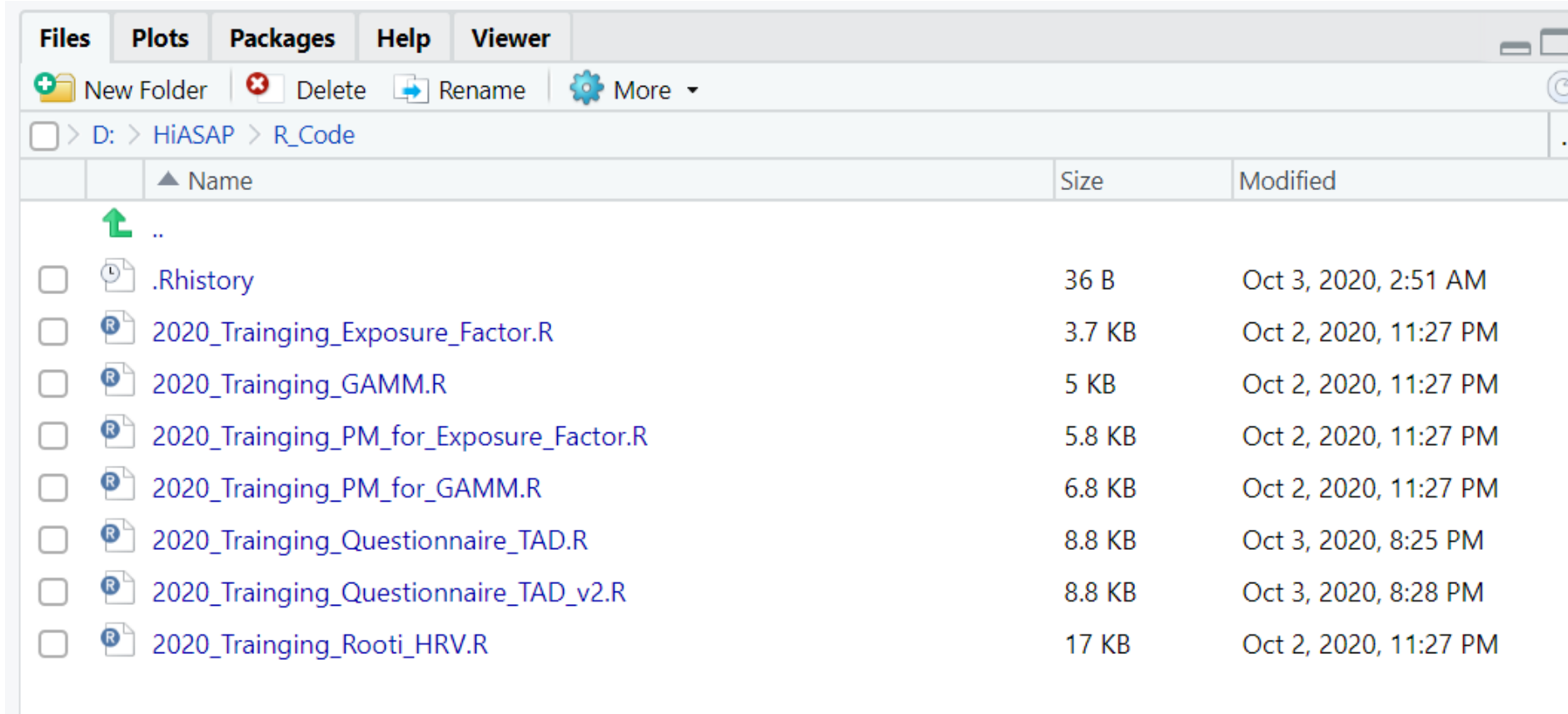
3. Workspace (upper right window)

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

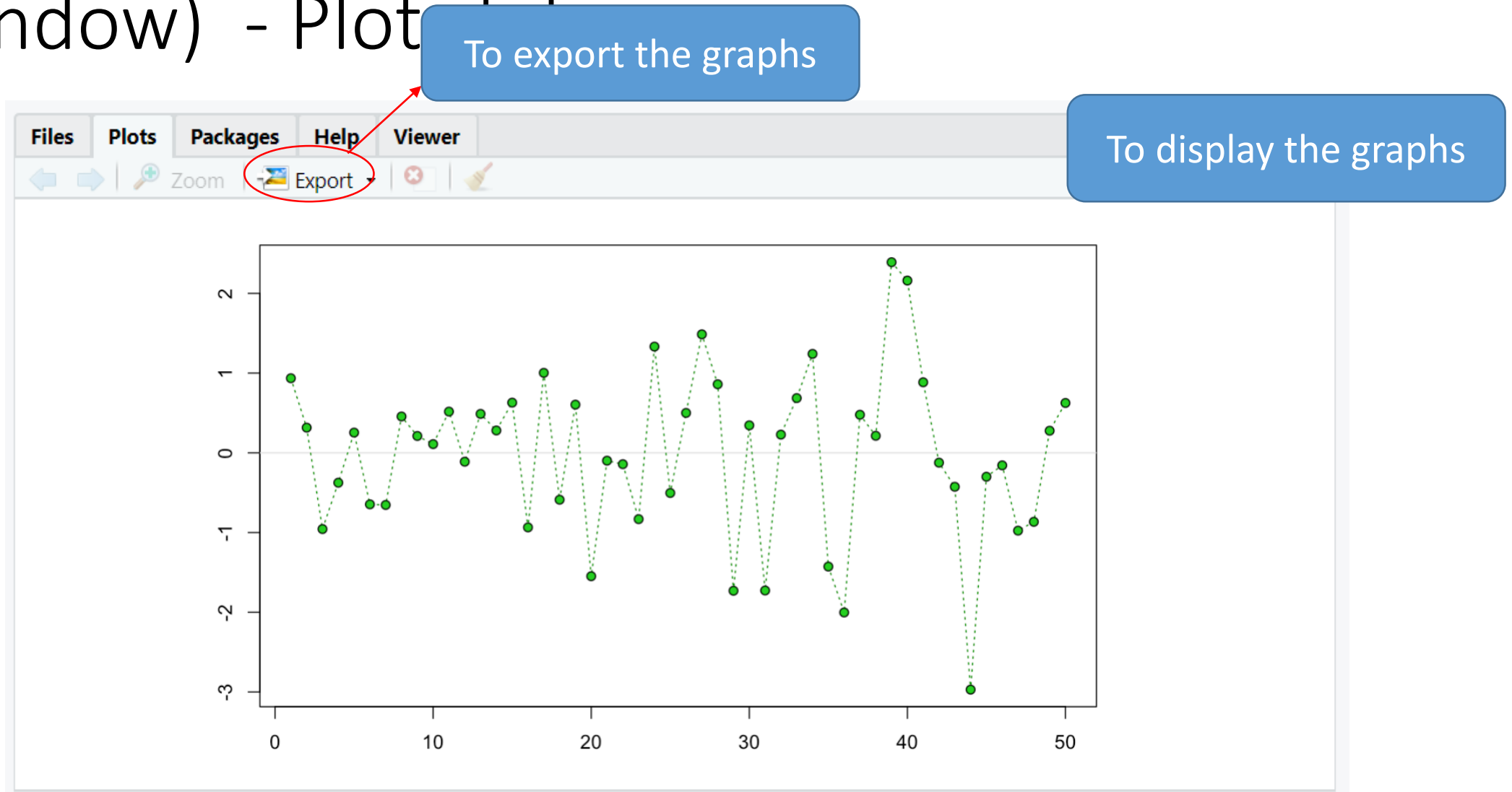
The screenshot shows the RStudio 'Environment' pane. At the top, there are tabs for 'Environment', 'History', 'Connections', and 'Tutorial'. Below the tabs is a toolbar with icons for file operations and a dropdown menu labeled 'Import Dataset'. The main area of the pane is titled 'Global Environment' and contains a list of objects in the workspace. The objects are listed in a table-like format with columns for the object name, its description, and a small icon. The objects are: 'Add_Row' (1 obs. of 11 variables), 'ALFinal' (577 obs. of 7 variables), 'ALFinal2' (575 obs. of 9 variables), and 'ALFinal3' (575 obs. of 9 variables). The 'ALFinal' object is expanded, showing its structure: 'date' (Factor w/ 577 levels), 'TEM' (numeric), 'HUM' (numeric), 'PM1' (numeric), 'PM2.5' (numeric), 'CO2' (numeric), and 'Freq' (integer). The 'ALFinal2' and 'ALFinal3' objects are also expanded, showing their structure: 'date' (Factor w/ 575 levels), 'TEM' (numeric), 'HUM' (numeric), 'PM1' (numeric), 'PM2.5' (numeric), 'CO2' (numeric), and 'Freq' (integer).

Object	Description	Icon
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	
ALFinal3	575 obs. of 9 variables	

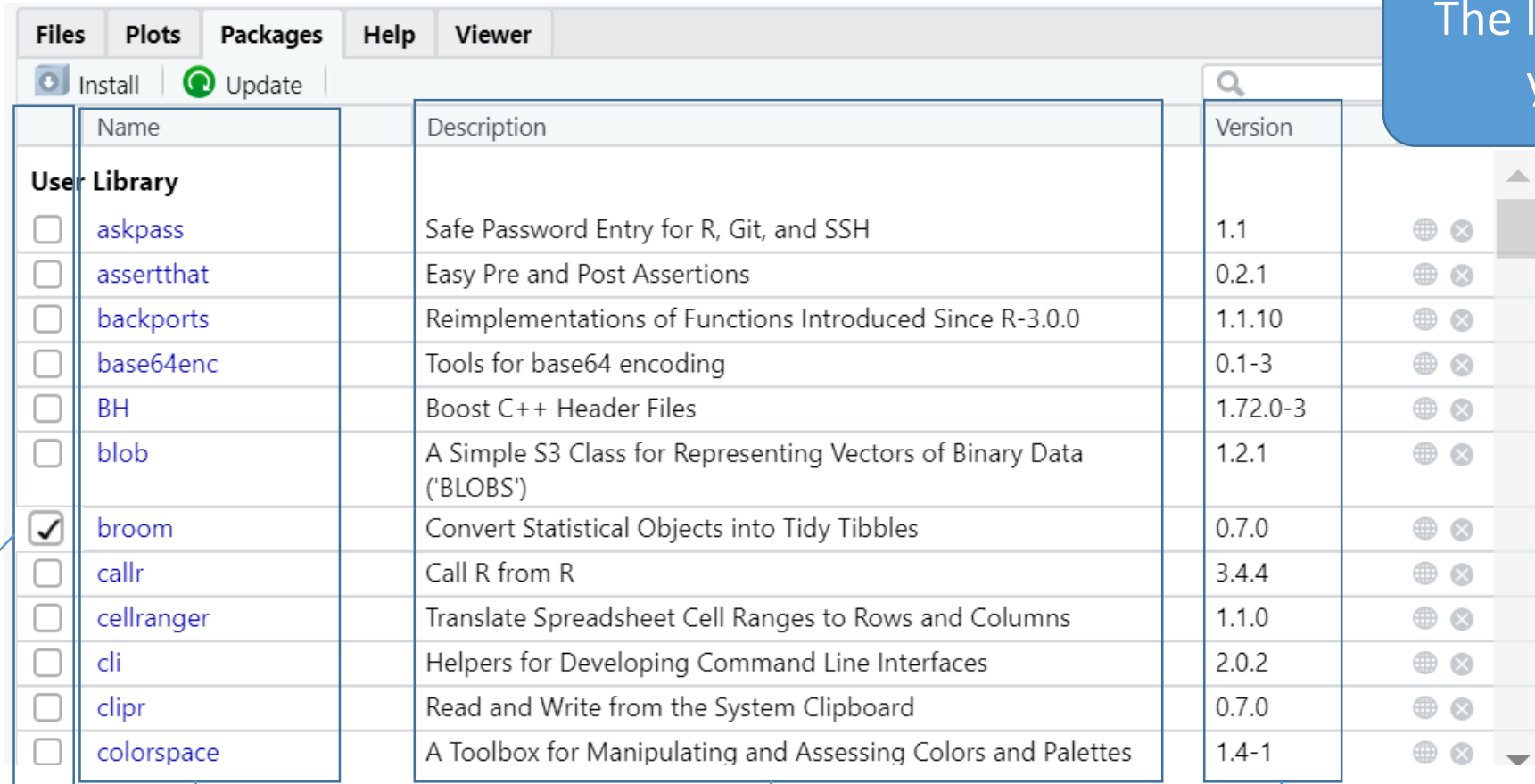
4. Files, plots, packages and help (lower right window) - Files tab



4. Files, plots, packages and help (lower right window) - Plot



4. Files, plots, packages and help (lower right window) - Packages tab

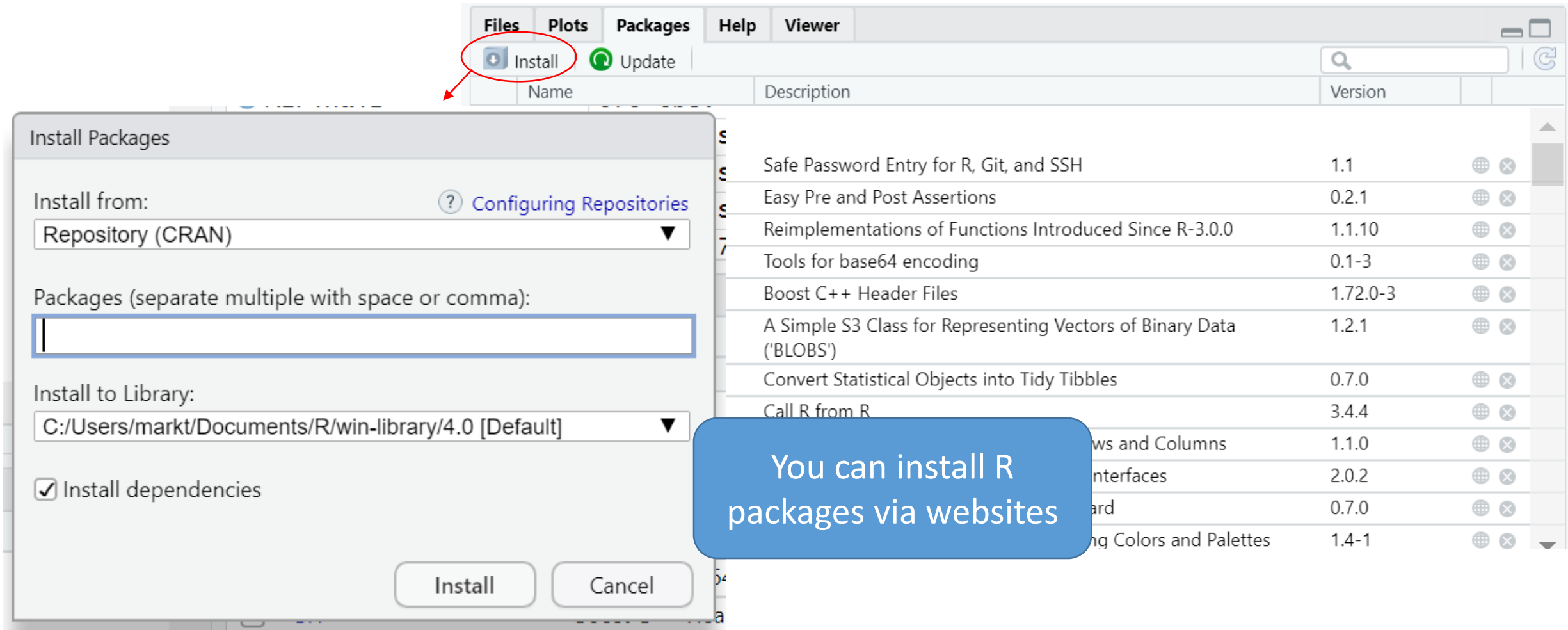


	Name	Description	Version
User Library			
<input type="checkbox"/>	askpass	Safe Password Entry for R, Git, and SSH	1.1
<input type="checkbox"/>	assertthat	Easy Pre and Post Assertions	0.2.1
<input type="checkbox"/>	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.10
<input type="checkbox"/>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

4. Files, plots, packages and help (lower right window) - Packages tab

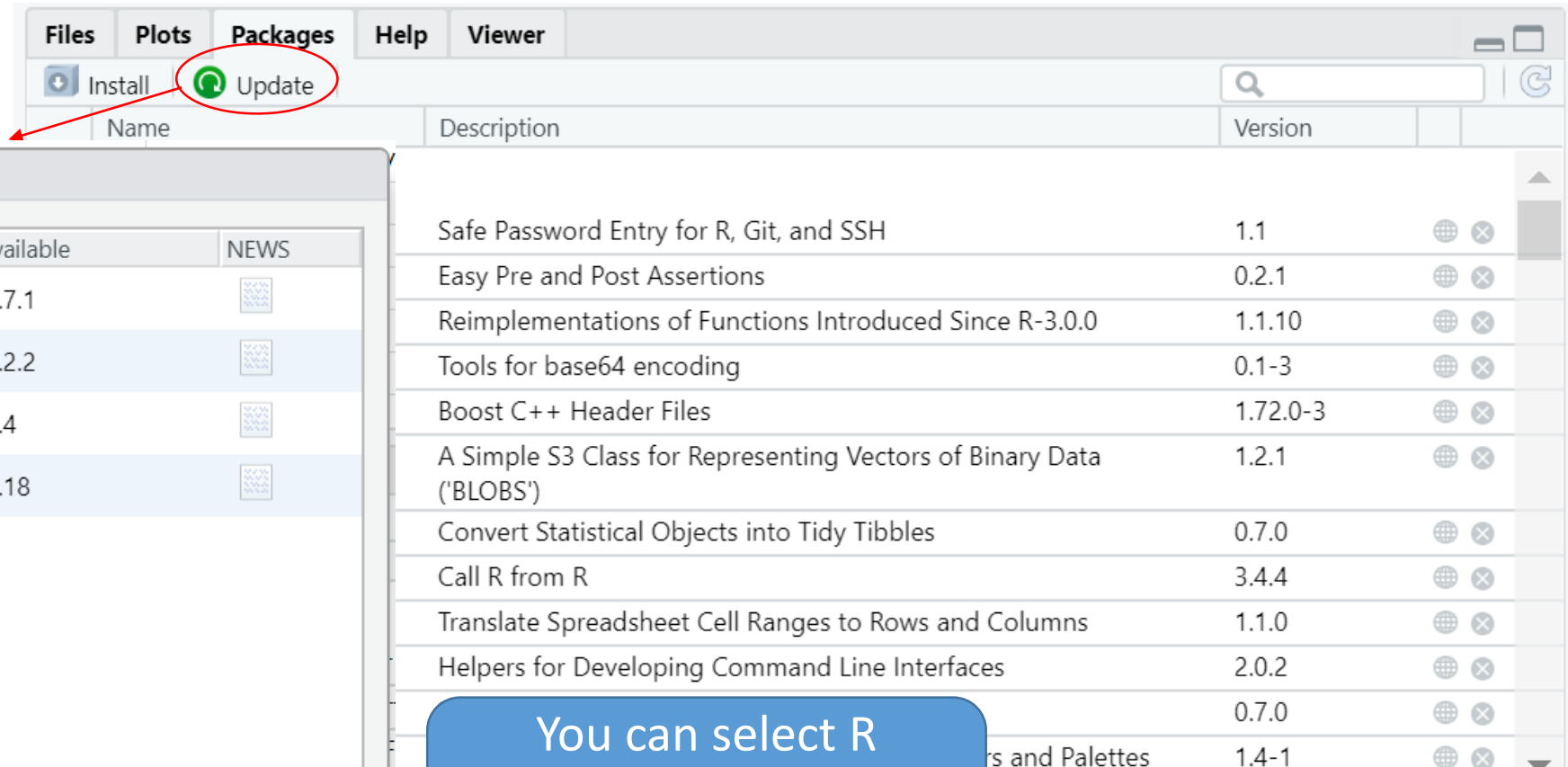


The screenshot shows the RStudio interface with the 'Packages' tab selected. The 'Install Packages' dialog box is open, showing the 'Repository (CRAN)' dropdown and the 'Install to Library' dropdown set to 'C:/Users/mark/Documents/R/win-library/4.0 [Default]'. The 'Install dependencies' checkbox is checked. A blue callout box states: 'You can install R packages via websites'.

The background shows a list of installed packages with columns for Name, Description, and Version. The list includes:

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
Rows and Columns		1.1.0
Interfaces		2.0.2
Card		0.7.0
Ang Colors and Palettes		1.4-1

4. Files, plots, packages and help (lower right window) - Packages tab



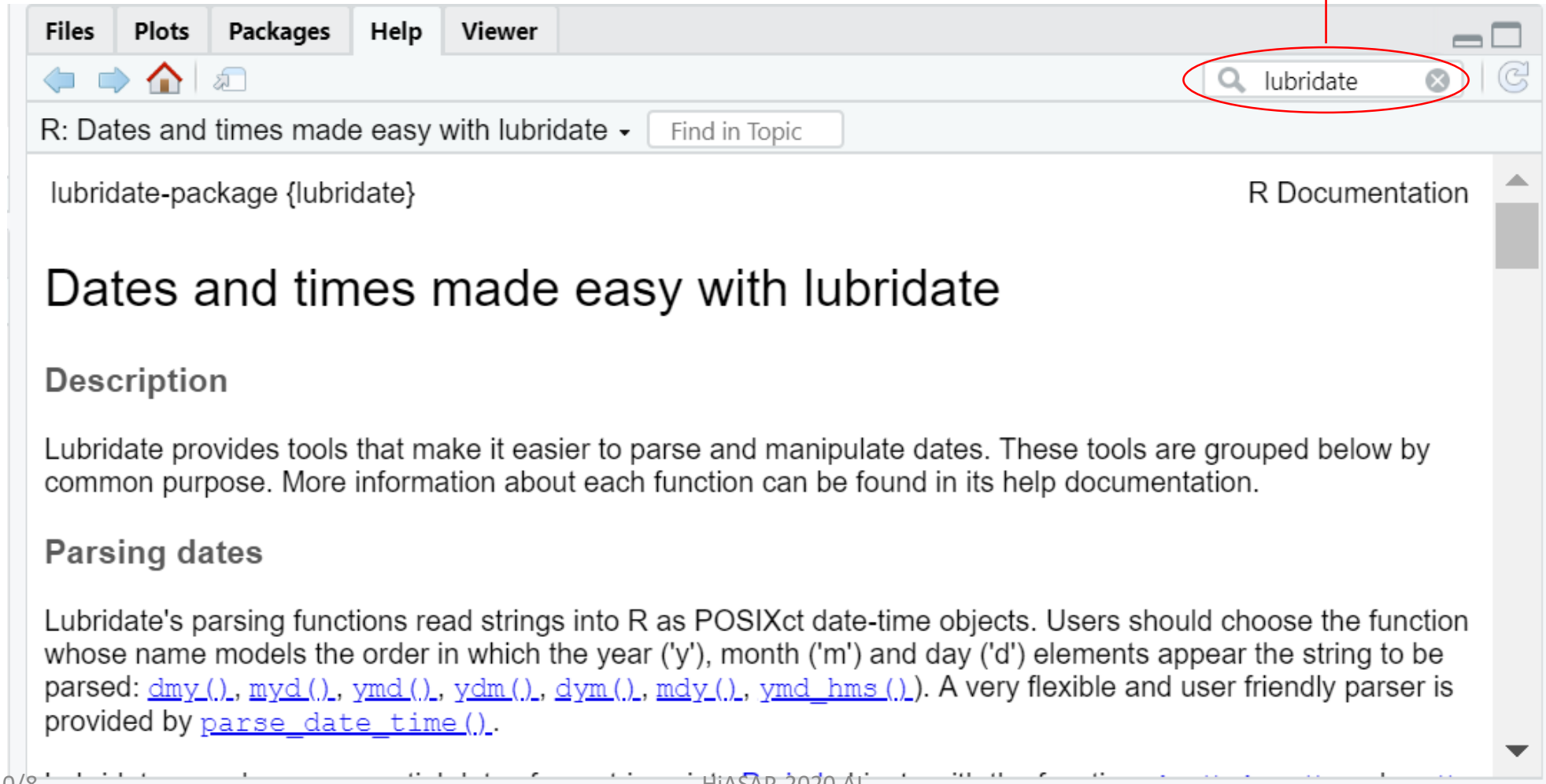
The screenshot shows the RStudio interface with the 'Packages' tab selected. The 'Update' button is circled in red, and an arrow points from it to the 'Update Packages' dialog box. The dialog box displays a table of packages with columns for Package, Installed, Available, and NEWS. The table lists four packages: broom, cpp11, rmarkdown, and xfun. The 'Update Packages' dialog box also has buttons for 'Select All', 'Select None', 'Install Updates', and 'Cancel'.

Package	Installed	Available	NEWS
<input type="checkbox"/> broom	0.7.0	0.7.1	
<input type="checkbox"/> cpp11	0.2.1	0.2.2	
<input type="checkbox"/> rmarkdown	2.3	2.4	
<input type="checkbox"/> xfun	0.17	0.18	

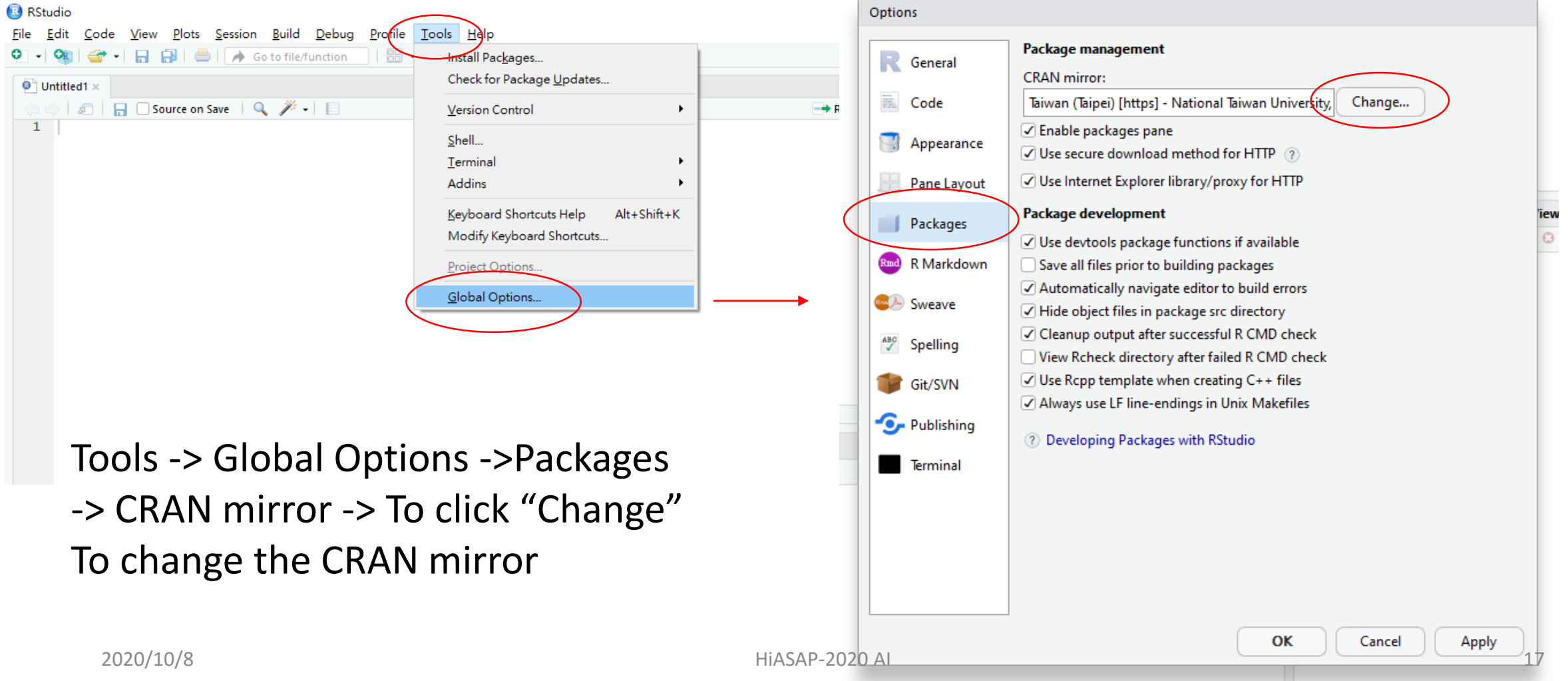
You can select R packages which you want to update

4. Files, plots, packages and help (lower right window) - Help tab

You can search the documents of R packages



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



The image shows the RStudio interface with the 'Tools' menu open. The 'Global Options...' option is selected. The 'Options' dialog box is open, showing the 'Packages' tab. The 'CRAN mirror:' dropdown is set to 'Taiwan (Taipei) [https] - National Taiwan University', and the 'Change...' button is highlighted. A red arrow points from the 'Global Options...' menu item to the 'Options' dialog box.

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

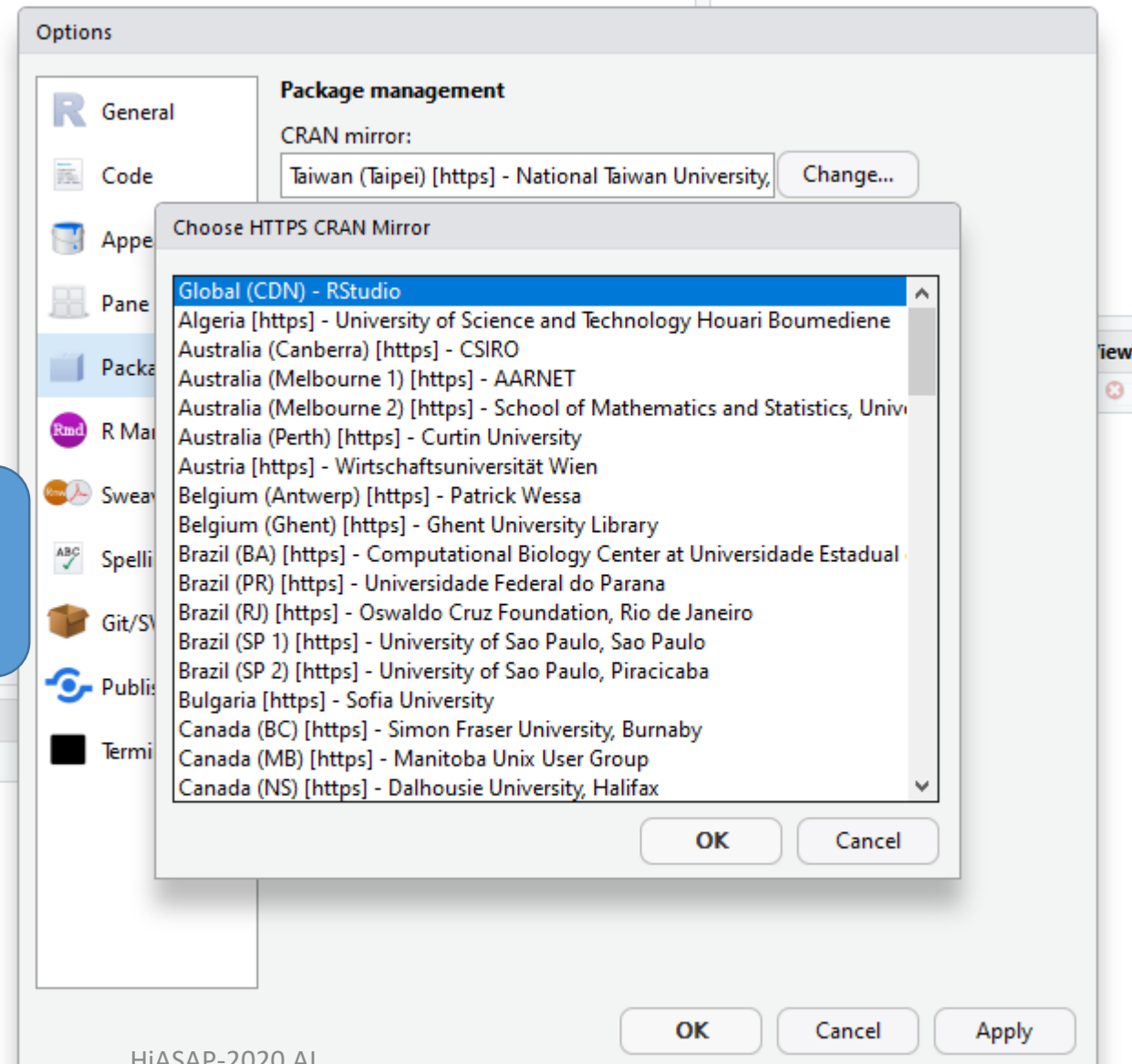
2020/10/8

HiASAP-2020 AI

17

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

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




Part 3:

QA/QC of HRV data

Notices

- All text after the pound sign "#" within the same line in R code
 - To be considered a comment
 - To be not run by R
- In this presentation, the R codes in the red box can be modify according your requirements (mostly the path of data file)
 - For example, you can modify the path of output file by yourself

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



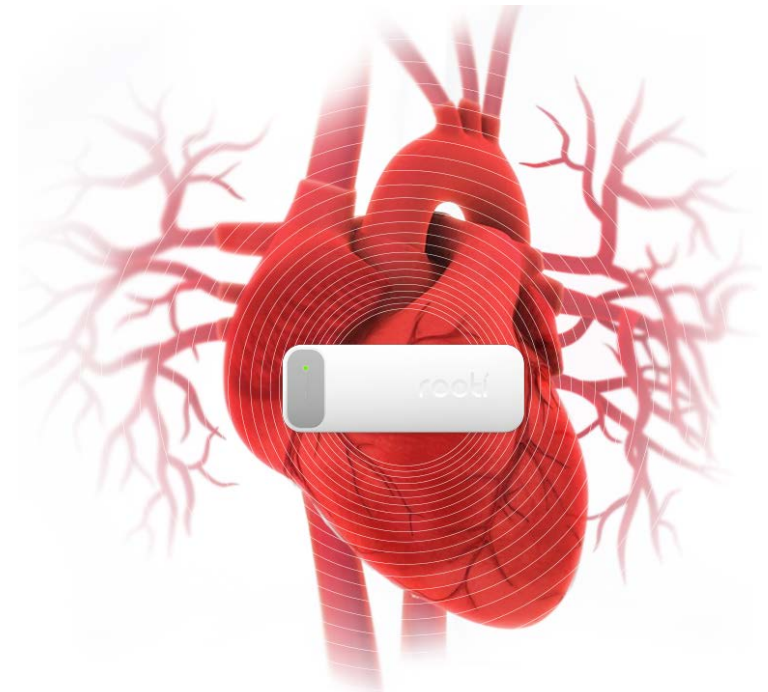
Notices

- To use quotes (") to tell R to interpret something as a string rather than an object.
 - For example,

```
> "Apple"
[1] "Apple"
> Apple
Error: object 'Apple' not found
```
- The uppercase letters (strings) are different from the lowercase letters (strings)
 - For example,
 - $A \neq a$
 - $\text{Apple} \neq \text{apple}$

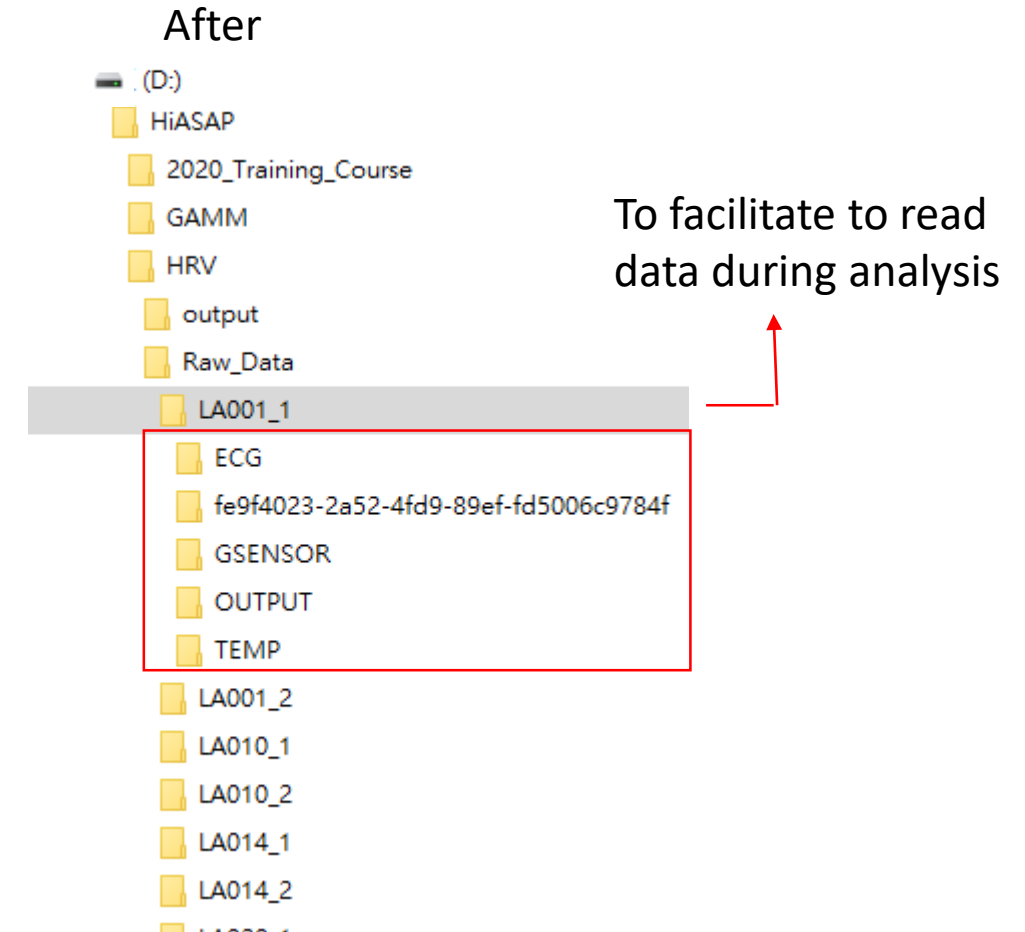
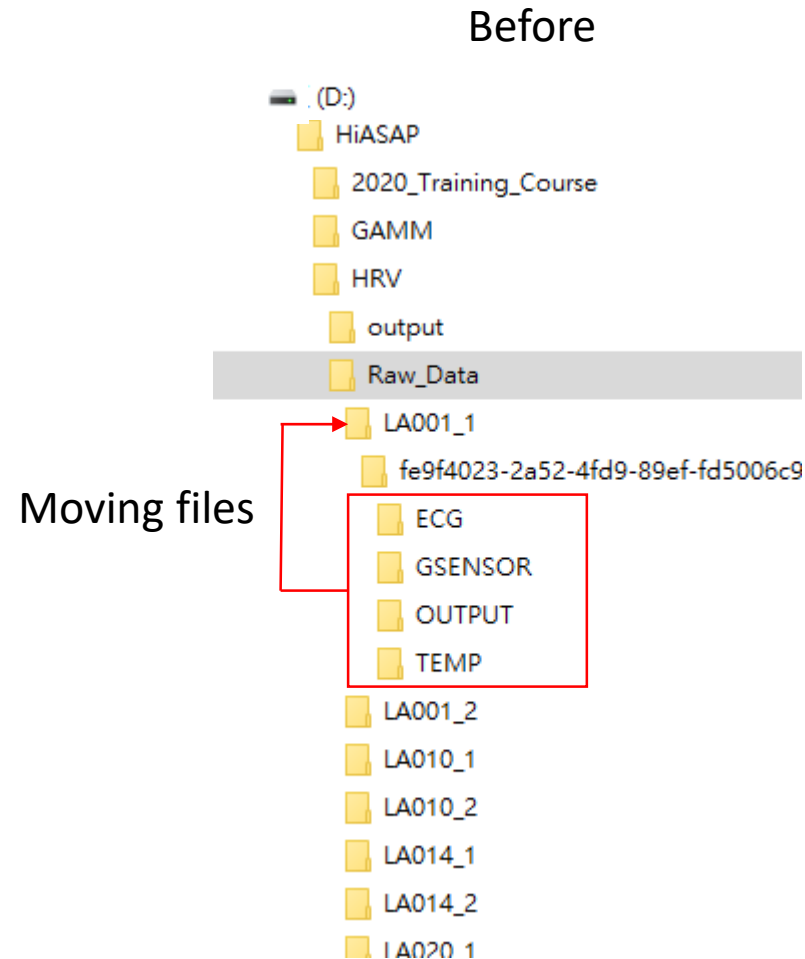
What kind of data we can get from Rooti

- 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

Moving all data files to its own file for every subject



```
1 # To remove previous memory in R.  
2 rm(list=ls())  
3  
4 # To update the R packages  
5 update.packages("lubridate")  
6 update.packages("plyr")  
7 update.packages("tidyverse")  
8 update.packages("jsonlite")  
9 update.packages("data.table")  
10  
11 # To load the R packages  
12 library(lubridate)  
13 library(plyr)  
14 library(tidyverse)  
15 library(jsonlite)  
16 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

```
18 location <- "D:/"
19
20 # To set the output file
21 cmd1 <- paste0("setwd('",location,"HiASAP/HRV/output')")
22 eval(parse(text=cmd1))
```

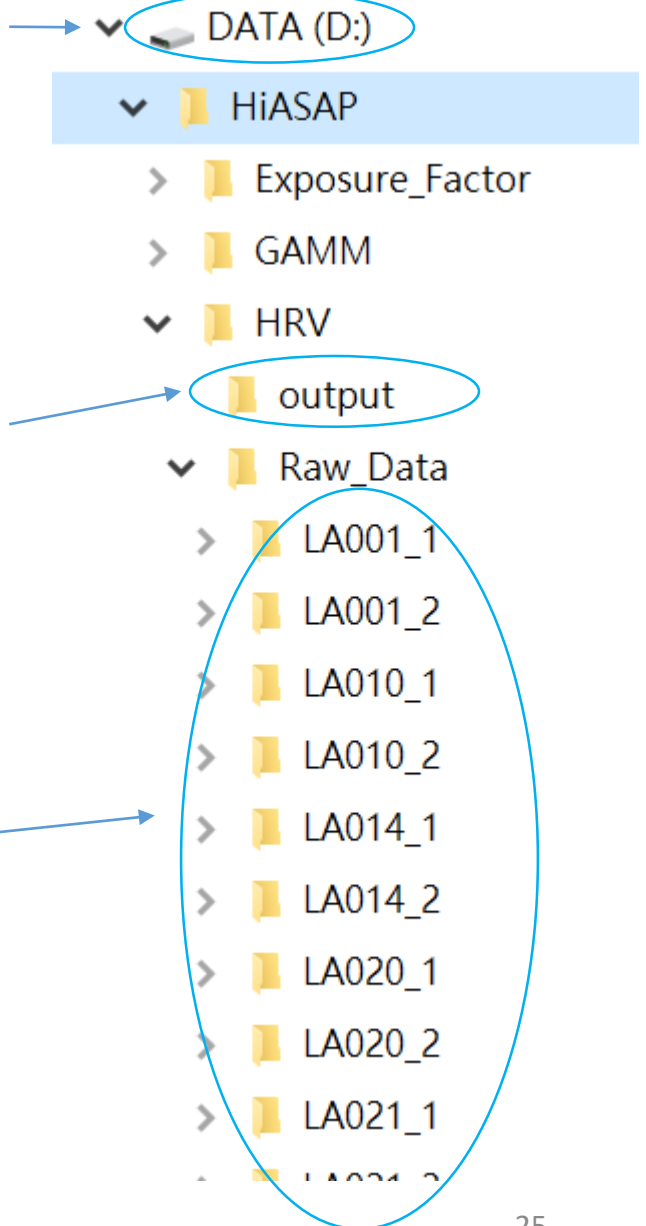
4. To set the default drive of your data

5. To set the path of output file

6. To set the path of HRV data

```
24 # To set the raw data file
25 way <- paste0(location,"HiASAP/HRV/Raw_Data/")
26 # To set the ID list
27 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

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

- DATA (D:)
 - HiASAP
 - Exposure_Factor
 - GAMM
 - HRV
 - output
 - Raw_Data
 - LA001_1
 - ECG
 - fe9f4023-2a52-4
 - GSENSOR
 - OUTPUT

2020/10/8 TEMP

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

Ex: 2020-10-06 08:00:00

The code of time zone

- R_property2010
- R_property2250
- R_property2490
- R_property2730
- result.json
- rr1
- rr2
- SDNN_5
- SDNN1
- SDNN2

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, activity and sleeping data with almost the same procedures

8. To read the results of 5-min SDNN

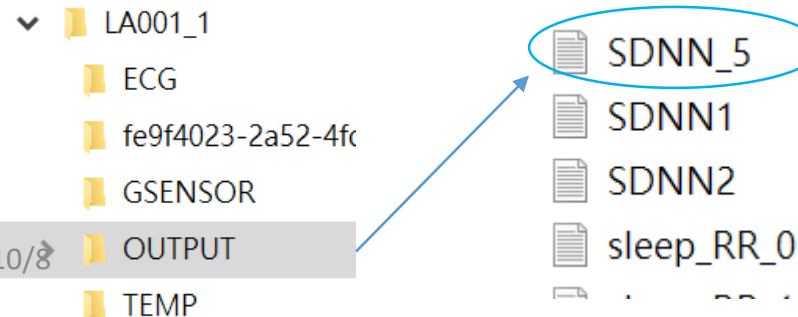
```
37 # To get the 5-min SDNN data
38 aa <- read.table(paste0(way, subject[i], "/OUTPUT/SDNN_5.txt"))
39 test<-substr(subject[i],1,5)
40 Rooti_SDNN5<-data.frame(test,aa)
41 colnames(Rooti_SDNN5)[names(Rooti_SDNN5) == "V1"]<-"SDNN5"
```

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 Datetime<-seq.POSIXt(start_time[[i]][1],start_time[[i]][1]+total_time , by = "5 mins")
45 Datetime<-Datetime[1:(length(Datetime)-1)]
```

```
46
47 Rooti_SDNN5<-data.frame(Datetime, subset(Rooti_SDNN5,select=c(test,SDNN5)))
```



10. To combine data time with SDNN data

Extracting the HRV data

11. To extract the 5-min RMSSD, LF/HF ratio, LF, HF, VLF and TP 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(Datetime, subset(Rooti_RMSSD5,select=c(test,RMSSD5)))
56
57 # To get the 5-min LF/HF data
58 aa 69 # To get the 5-min HF data
59 te 70 aa <- read.table(paste0(way, subject[i], "/OUTPUT/hf_5.txt"))
60 Ro 71 test<-substr(subject[i],1,5)
61 co 72 Rooti_hf5<-data.frame(test,aa)
62 73 colnames(Rooti_hf5)[names(Rooti_hf5) == "V1"]<-"HF5"
63 # 74
64 aa 75 # To get the 5-min VLF data
65 te 76 aa <- read.table(paste0(way, subject[i], "/OUTPUT/vlf_5.txt"))
66 Ro 77 test<-substr(subject[i],1,5)
67 co 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"
```

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



→ You can modify by yourself

Extracting the heart rate (HR) data

The codes for HR data are the same as that for SDNN, except the **temporal resolution of data**

12. To read the 1-min HR data

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

13. To create the variable of data time

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

```
97
98 Rooti_HR<-data.frame(Datatime_HR, Rooti_HR)
99 Rooti_HR<-Rooti_HR %>%
100   group_by(Datatime = cut(Datatime_HR, breaks="300 secs")) %>%
101   summarize(
102     HRsum5 = sum(HR),
103     HRmean5 = floor(mean(HR)))
104 Rooti_HR$Datatime <-ymd_hms(Rooti_HR$Datatime,tz="Asia/Taipei")
```

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

Extracting activity data (variations for three-axis)



→ You can modify by yourself

- 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 Datetime_gsensor<-seq.POSIXt(start_time[[i]][1],start_time[[i]][1]+total_gsensor_time , by = "1 mins")
116 Datetime_gsensor<-Datetime_gsensor[1:(length(Datetime_gsensor)-1)]
117
118 Rooti_gsensor<-data.frame(Datetime_gsensor, Rooti_gsensor)
119 Rooti_gsensor<-Rooti_gsensor %>%
120   group_by(Datetime = cut(Datetime_gsensor, breaks="300 secs")) %>%
121   summarize(Gsensor5 = sum(Gsensor))
122 Rooti_gsensor$Datetime <-ymd_hms(Rooti_gsensor$Datetime,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)

```
124 # To get the 1-min data of accelerations for three-axis
125 aa <- list.files(paste0(way, subject[i], "/GSENSOR/"))
126 bb <- read.table(paste0(way, subject[i], "/GSENSOR/", aa[1]), sep=",")
127 cc <- bb
128 for(j in 2:length(aa)){
129     bb <- read.table(paste0(way, subject[i], "/GSENSOR/", aa[j]), sep=",")
130     cc <- rbind(cc, bb)
131 }
132 test <- substr(subject[i], 1, 5)
133 Rooti_gsensor_raw_data <- data.frame(test, subset(cc, select=c(V1:V5)))
134 colnames(Rooti_gsensor_raw_data) <- c("S_no", "Datatime", "secondpoint", "X", "Y", "Z")
135 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

```
137 # To get the 5-min data of accelerations for three-axis
138 Rooti_gsensor_raw_data <- Rooti_gsensor_raw_data %>%
139   group_by(Datatime = cut(Datatime, breaks="300 secs")) %>%
140   summarize(meanX5 = round(mean(X), 4),
141             meanY5 = round(mean(Y), 4),
142             meanZ5 = round(mean(Z), 4),
143             maxX5 = round(max(X), 4),
144             maxY5 = round(max(Y), 4),
145             maxZ5 = round(max(Z), 4)) %>%
146   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)

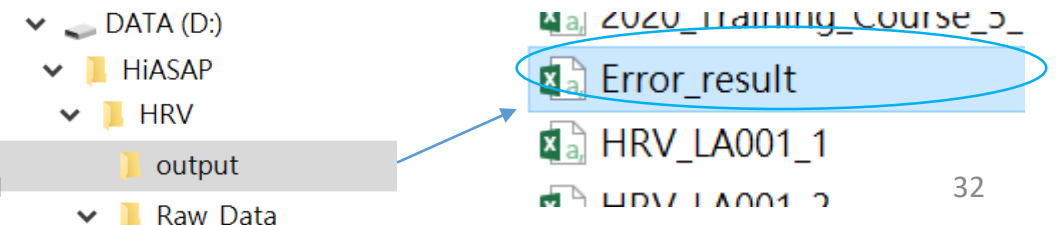
- Because the start time may be different between G-sensor and HRV monitoring ("result.json" file), the difference between two files should be less than 3 seconds.

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

```
149 # To check whether the time of G-sensor is correct
150 gap_of_time<-seconds(Rooti_SDNN5$Datatime[1]-Rooti_gsensor_raw_data$Datatime[1])
151
152 if(abs(gap_of_time)<=3){
153   Rooti_gsensor_raw_data<-Rooti_gsensor_raw_data %>%
154     mutate(Datatime = Datatime +gap_of_time)
155 }else{
156   start_time_error_gsensor<-"Please check the start time of gsensor in GSENSOR folder and start time of activity in result.json."
157   write.csv(start_time_error_gsensor,paste0("Start_time_error_in ",subject[i],"_gsensor.csv"),row.names = F)
158 }
```

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

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

19. To get the start time of sleeping time from "result.json" file

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

```
166 if(!is.null(Rooti_online_result[[i]]$sleep$sleepStartTime)){
167   sleep_start_time[[i]]<-as.POSIXct(sleep_start_time[[i]], origin="1970-01-01", tz="Asia/Taipei")
168   in_bed_time[[i]]<-Rooti_online_result[[i]]$sleep$inBedTime
169   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 7
sleep	list [2 x 12] (S3: data.frame)	A data.frame with 2 rows and 12 columns
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 2



→ You can modify by yourself

Extracting sleeping index

21. To calculate the data time of the sleeping time

```
170 for (s in 1:length(sleep_start_time[[i]])) {  
171   Datetime_sleep_m<-data.frame(Datetime =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

```
172 Datetime_sleep_m<-Datetime_sleep_m[(2:nrow(Datetime_sleep_m)),]  
173 sleep_idx[[i]][[s]]<-sleep_idx[[i]][[s]][(1:length(Datetime_sleep_m))]  
174 Datetime_sleep_m<-data.frame(Datetime =Datetime_sleep_m,sleep_idx=sleep_idx[[i]][[s]])  
175 Datetime_sleep<-rbind(Datetime_sleep_m,Datetime_sleep)  
176 Datetime_sleep$Datetime<-as.POSIXct(Datetime_sleep$Datetime, origin="1970-01-01",tz="Asia/Taipei")  
177 }
```



→ You can modify by yourself

Extracting sleeping index

- Because the start time may be also different between sleeping data and HRV monitoring 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

```
179 # To check whether the time of sleeping is correct
180 Datetime_gsensor<-data.frame(Datetime = Datetime_gsensor)
181 Datetime_gsensor$Datetime<-as.POSIXct(Datetime_gsensor$Datetime, origin="1970-01-01",tz="Asia/Taipei")
182 if(second(Datetime_gsensor$Datetime[1])==0&second(Datetime_sleep$Datetime[1])>55){
183   gap_of_time_sleep<-60-second(Datetime_sleep$Datetime[1])
184 }else if(second(Datetime_gsensor$Datetime[1])==1&second(Datetime_sleep$Datetime[1])>55){
185   gap_of_time_sleep<-61-second(Datetime_sleep$Datetime[1])
186 }else if(second(Datetime_gsensor$Datetime[1])==2&second(Datetime_sleep$Datetime[1])>55){
187   gap_of_time_sleep<-62-second(Datetime_sleep$Datetime[1])
188 }else if(second(Datetime_gsensor$Datetime[1])>55&second(Datetime_sleep$Datetime[1])==0){
189   gap_of_time_sleep<-second(Datetime_gsensor$Datetime[1])-60
190 }else if(second(Datetime_gsensor$Datetime[1])>55&second(Datetime_sleep$Datetime[1])==1){
191   gap_of_time_sleep<-second(Datetime_gsensor$Datetime[1])-61
192 }else if(second(Datetime_gsensor$Datetime[1])>55&second(Datetime_sleep$Datetime[1])==2){
193   gap_of_time_sleep<-second(Datetime_gsensor$Datetime[1])-62
194 }else{
205   gap_of_time_sleep<-second(Datetime_gsensor$Datetime[1])-second(Datetime_sleep$Datetime[1])
196 }
```



→ You can modify by yourself

Extracting sleeping index

```
198 if(abs(gap_of_time_sleep)<=3){
199   Datatime_sleep<-Datatime_sleep %>%
200     mutate(Datatime = Datatime +gap_of_time_sleep) %>%
201     full_join(Datatime_gsensor,by = "Datatime")
202   Datatime_sleep<-Datatime_sleep %>%
203     group_by(Datatime = cut(Datatime, breaks="300 secs")) %>%
204     summarize(sleep_idx5 = max(sleep_idx,na.rm=F)
205   ) %>%
206   mutate(Datatime = ymd_hms(Datatime,tz="Asia/Taipei"))
207 }else{
208   start_time_error_sleep<-"Please check the start time of sleep and start time of activity in result.json."
209   write.csv(start_time_error_sleep,paste0("Start_time_error_in ",subject[i],"_sleep.csv"),row.names = F)
210 }
211 }
```

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 (D:)
HiASAP
HRV
output
Raw_Data

2020_Training_Course_3_
Error_result
HRV_LA001_1
HRV_LA001_2

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, Datetime, 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 = "Datetime")
Rooti_total<-Rooti_total %>%
  full_join(Rooti_gsensor_raw_data,by = "Datetime")
```

25. To combine activity data with HRV data

```
if(!is.null(Rooti_online_result[[i]]$sleep$sleepStartTime)){
  Rooti_total<-Rooti_total %>%
    full_join(Datetime_sleep,by = "Datetime")
}else{
  Rooti_total$sleep_idx5<-NA
}
```

No sleeping data
-> Awake

26. To combine sleeping index with
HRV data

```
Rooti_total<-Rooti_total %>%
  full_join(Rooti_HR,by = "Datetime")
```

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,Datetime,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

```
247 # To exclude data of ineffective time (bad time) and extreme data
248 badtime_path <- list.files(paste0(way, subject[i], "/OUTPUT/bad_time.txt"))
249
250 bad_time<-list()
251 if(length(badtime_path)==0){
252   bad_time[[i]]<-Rooti_online_result[[i]]$Q_factor$bad_min
253 }else{
254   bad_time <- read.table(paste0(way, subject[i], "/OUTPUT/bad_time.txt"))
255   for (j in 1:nrow(bad_time[[i]])) {
256     bad_time[[i]][j]<-start_time[[i]][1]+minutes(bad_time[[i]][j])
257   }
258 }
```

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)

```
260 bad_time[[i]]<-data.frame(V1 = bad_time[[i]][[1]])  
261 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

```
263 Rooti_total$bad_time<-0  
264 for (q in 1:(nrow(bad_time[[i]]))) {  
265   for (k in 1:(nrow(Rooti_total)-1)) {  
266     if(!(Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&!(bad_time[[i]]$V1[q]<Rooti_total$Datatime[k+1])&(Rooti_total$bad_time[k]==0)){  
267       Rooti_total$bad_time[k]<-0  
268     }else if(!(Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&!(bad_time[[i]]$V1[q]<Rooti_total$Datatime[k+1])&(Rooti_total$bad_time[k]==1)){  
269       Rooti_total$bad_time[k]<-1  
270     }else if((Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&(bad_time[[i]]$V1[q]<Rooti_total$Datatime[k+1])&(Rooti_total$bad_time[k]==1)){  
271       Rooti_total$bad_time[k]<-1  
272     }else if((Rooti_total$Datatime[k]<=bad_time[[i]]$V1[q])&(bad_time[[i]]$V1[q]<Rooti_total$Datatime[k+1])&(Rooti_total$bad_time[k]==0)){  
273       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

```
277 for (q in 1:nrow(bad_time[[i]])) {  
278   for (k in nrow(Rooti_total)) {  
279     if(! (Rooti_total$Datatime[k] <= bad_time[[i]]$V1[q]) & ! (bad_time[[i]]$V1[q] <= (Rooti_total$Datatime[k] + minutes(4))) & (Rooti_total$bad_time[k] == 0))  
280       Rooti_total$bad_time[k] <- 0  
281   } else if(! (Rooti_total$Datatime[k] <= bad_time[[i]]$V1[q]) & ! (bad_time[[i]]$V1[q] <= (Rooti_total$Datatime[k] + minutes(4))) & (Rooti_total$bad_time[k] == 1))  
282     Rooti_total$bad_time[k] <- 1  
283   } else if((Rooti_total$Datatime[k] <= bad_time[[i]]$V1[q]) & (bad_time[[i]]$V1[q] <= (Rooti_total$Datatime[k] + minutes(4))) & (Rooti_total$bad_time[k] == 0))  
284     Rooti_total$bad_time[k] <- 1  
285   } else if((Rooti_total$Datatime[k] <= bad_time[[i]]$V1[q]) & (bad_time[[i]]$V1[q] <= (Rooti_total$Datatime[k] + minutes(4))) & (Rooti_total$bad_time[k] == 1))  
286     Rooti_total$bad_time[k] <- 1  
287   }  
288 }
```

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

```
290 Rooti_total <- Rooti_total %>%  
291   filter(!(bad_time == 1)) %>%  
292   filter(SDNN5 > 0 & SDNN5 < 400 & LFHF5 > 0.01 & HRmean5 > 40 & HRmean5 < 200) %>%  
293   select(-c(bad_time))  
294  
295 write.csv(Rooti_total, paste0("HRV_", subject[i], ".csv"), row.names = F)
```

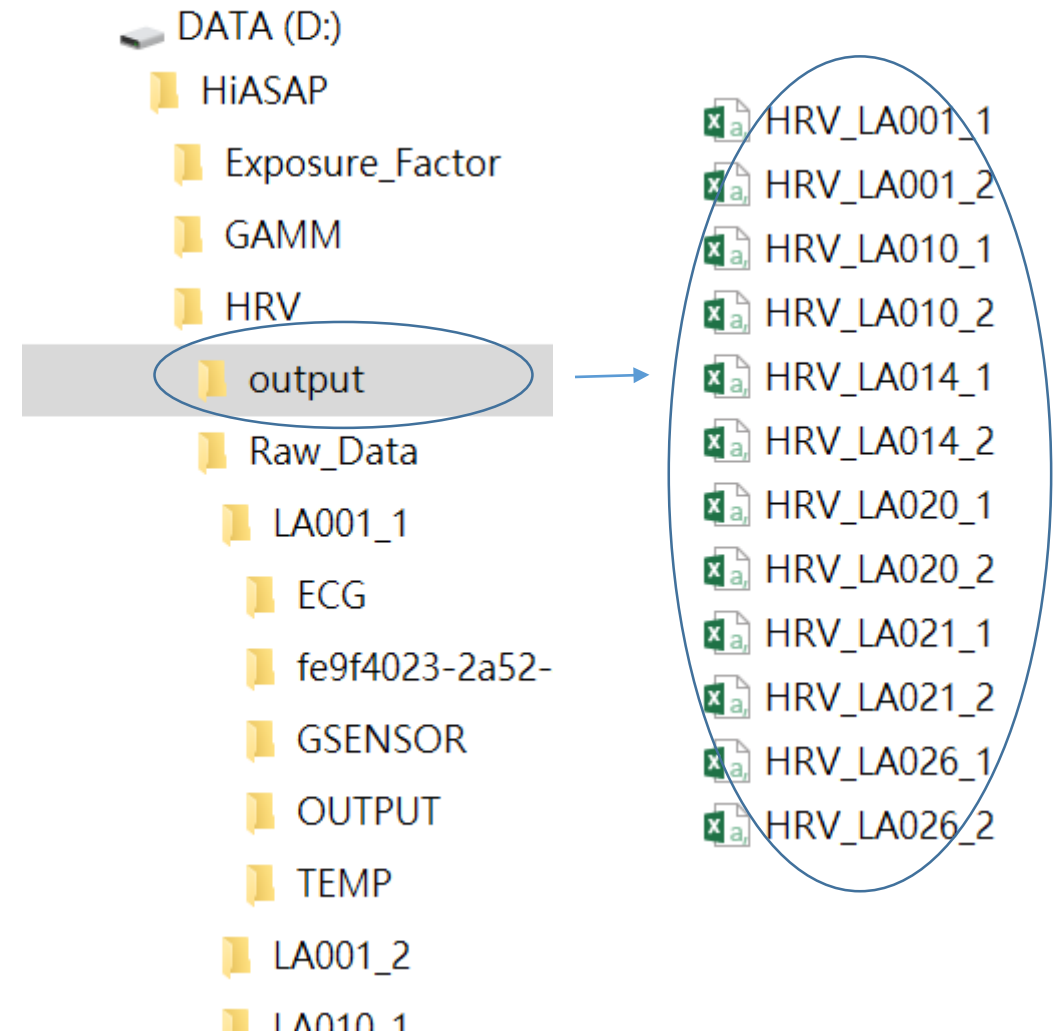
To exclude bad time

To exclude abnormal signals

Data combination for all subjects

37. To combine HRV data for all subjects

```
299 # To combine HRV data for all subjects
300 way2 <- paste0(location,"HiASAP/HRV/output")
301 aa1 <- list.files(way2,pattern="HRV")
302
303 HRV <- data.frame()
304 filename <- paste0(way2,"/",aa1[1])
305 cc <- read.csv(filename)
306 HRV <- cc
307 for(k in 2:length(aa1)){
308     filename <- paste0(way2,"/",aa1[k])
309     cc <- read.csv(filename)
310     HRV <- rbind(HRV,cc)
311 }
```



```

313 # To create the time variables (year, month, day, hour and minute) for the following data matching
314 library(lubridate)
315 yy<-c()
316 mm<-c()
317 dd<-c()
318 hh<-c()
319 mn<-c()
320 for(l in 1:dim(HRV)[1]){
321   if(nchar(as.character(HRV$Datatime[l]))==14){
322     yy[l] <- as.numeric(substr(HRV$Datatime[l],1,4))
323     mm[l] <- as.numeric(substr(HRV$Datatime[l],6,6))
324     dd[l] <- as.numeric(substr(HRV$Datatime[l],8,8))
325     hh[l] <- as.numeric(substr(HRV$Datatime[l],10,11))
326     mn[l] <- as.numeric(substr(HRV$Datatime[l],13,14))
327   }else{
328     if(nchar(as.character(HRV$Datatime[l]))==15){
329       yy[l] <- as.numeric(substr(HRV$Datatime[l],1,4))
330       mm[l] <- as.numeric(substr(HRV$Datatime[l],6,6))
331       dd[l] <- as.numeric(substr(HRV$Datatime[l],8,9))
332       hh[l] <- as.numeric(substr(HRV$Datatime[l],11,12))
333       mn[l] <- as.numeric(substr(HRV$Datatime[l],14,15))
334     }else{
335       yy[l] <- as.numeric(substr(HRV$Datatime[l],1,4))
336       mm[l] <- as.numeric(substr(HRV$Datatime[l],6,7))
337       dd[l] <- as.numeric(substr(HRV$Datatime[l],9,10))
338       hh[l] <- as.numeric(substr(HRV$Datatime[l],12,13))
339       mn[l] <- as.numeric(substr(HRV$Datatime[l],15,16))
340     }
  }
}

```

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

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

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

```
343 mn_30 <- c()
344 for (l in 1:length(mn)) {
345   if(mn[l] < 30){
346     mn_30[l] <- 1
347   }else{
348     mn_30[l] <- 2
349   }
350 }
```

Year	Month	Day	Hour	Minute	Minute_30
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

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

40. To format the date variable

Ex: 2020-01-01 14:11

```

352 HRVfinal <- data.frame()
353 for(j in 1:dim(HRV)[1]){
354     HRVfinal[j,1]<-date_1[j]
355     HRVfinal[j,2]<-yy[j]
356     HRVfinal[j,3]<-mm[j]
357     HRVfinal[j,4]<-dd[j]
358     HRVfinal[j,5]<-hh[j]
359     HRVfinal[j,6]<-mn[j]
360     HRVfinal[j,7]<-mn_30[j]
361     HRVfinal[j,8]<-HRV[j,1]
362     HRVfinal[j,9]<-HRV[j,3]
363     HRVfinal[j,10]<-HRV[j,4]
364     HRVfinal[j,11]<-HRV[j,5]
365     HRVfinal[j,12]<-HRV[j,6]
366     HRVfinal[j,13]<-HRV[j,7]
367     HRVfinal[j,14]<-HRV[j,8]
368     HRVfinal[j,15]<-HRV[j,9]
369     HRVfinal[j,16]<-HRV[j,10]
370     HRVfinal[j,17]<-HRV[j,11]
371     HRVfinal[j,18]<-HRV[j,12]
372     HRVfinal[j,19]<-HRV[j,13]
373     HRVfinal[j,20]<-HRV[j,14]
374     HRVfinal[j,21]<-HRV[j,15]
375     HRVfinal[j,22]<-HRV[j,16]
376     HRVfinal[j,23]<-HRV[j,17]
377     HRVfinal[j,24]<-HRV[j,18]
378     HRVfinal[j,25]<-HRV[j,19]
379 }

```

41. To combine time-related variables with HRV data

Data export

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

```
380 colnames(HRVfinal)<-c("Date", "Year", "Month", "Day", "Hour", "Minute", "Minute_30", "S_no", "HRsum5", "HRmean5", "RMSSD5", "SDNN5", "LFHF5", "LF5", "HF5", "VLF5", "TP5", "Gsensor5", "Mean5")
381 outputname<-"HRV_5 minute_All.csv"
382 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_30	S_no	HRsum5	HRmean5	RMSSD5	SDNN5	LFHF5	LF5	HF5	VLF5	TP5	Gsensor5	Mean5
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.
12	2019/3/8 21:39	2019	3	8	21	39	2	LA001	377	75	22	10	2.02821	140.2082	46.08022	264.4425	450.7175	126872	-50.

Thank you for your attention