Function Utility Package

%>% “pipe” (pass) data from one function to the next magrittr

mutate() create, transform, and re-define columns dplyr

select() keep, remove, select, or re-name columns dplyr

rename() rename columns dplyr

clean\_names() standardize the syntax of column names janitor

as.character(), , , etc.as.numeric()as.Date() convert the class of a column base R

across() transform multiple columns at one time dplyr

tidyselect functions use logic to select columns tidyselect

filter() keep certain rows dplyr

distinct() de-duplicate rows dplyr

rowwise() operations by/within each row dplyr

add\_row() add rows manually tibble

arrange() sort rows dplyr

recode() re-code values in a column dplyr

case\_when() re-code values in a column using more complex logical criteria dplyr

replace\_na(), , na\_if()coalesce() special functions for re-coding tidyr

age\_categories() and cut() create categorical groups from a numeric column epikit and base R

match\_df() re-code/clean values using a data dictionary matchmaker

which() apply logical criteria; return indices base R

######################################################################################################################################################################################################################################

data clean pepline

1.packages

pacman::p\_load(

rio, # importing data

here, # relative file pathways

janitor, # data cleaning and tables

lubridate, # working with dates

matchmaker, # dictionary-based cleaning

epikit, # age\_categories() function

tidyverse # data management and visualization

)

2.import

linelist\_raw <- import("linelist\_raw.xlsx")

3. get an overview of the entire dataframe

skimr::skim(linelist\_raw)

4.Column names

names(linelist\_raw)

linelist <- linelist\_raw %>%

janitor::clean\_names()

replace =replace = c(onset = "date\_of\_onset"))

linelist <- linelist\_raw %>%

janitor::clean\_names() %>%

# NEW name # OLD name

rename(date\_infection = infection\_date,

date\_outcome = date\_of\_outcome)

5 Select or re-order columns

linelist %>%

select(#排好顺序的列名 everything())) %>%

names() # display the column names

select(-c(date\_onset, fever:vomit)) %>% # remove date\_onset and all columns from fever to vomit

names()

6 Deduplication

distinct()

7 Column creation and transformation

new\_col\_demo <- linelist %>%

mutate(

new\_var\_dup = case\_id, # new column = duplicate/copy another existing column

new\_var\_static = 7, # new column = all values the same

new\_var\_static = new\_var\_static + 5, # you can overwrite a column, and it can be a calculation using other variables

new\_var\_paste = stringr::str\_glue("{hospital} on ({date\_hospitalisation})") # new column = pasting together values from other columns

) %>%

select(case\_id, hospital, date\_hospitalisation, contains("new")) # show only new columns, for demonstration purposes

linelist <- linelist %>%

mutate(age = as.numeric(age))%>% #转换类型

group\_by(hospital)

8 Re-code values

# fix incorrect values # old value # new value

linelist <- linelist %>%

mutate(date\_onset = recode(date\_onset, "2014-14-15" = "2014-04-15"))

9 Numeric categories

#check the class of the linelist variable age

class(linelist$age\_years)

# examine the distribution

hist(linelist$age\_years)

10 Add rows

add\_row()

11 Filter rows

linelist <- linelist %>%

filter(gender == "f") # keep only rows where gender is equal to "f"

总的：

#################################################################################################################################################################################################################################

linelist <- linelist\_raw %>%

# standardize column name syntax

janitor::clean\_names() %>%

# manually re-name columns

# NEW name # OLD name

rename(date\_infection = infection\_date,

date\_hospitalisation = hosp\_date,

date\_outcome = date\_of\_outcome) %>%

# remove column

select(-c(row\_num, merged\_header, x28)) %>%

# de-duplicate

distinct(.keep\_all=TRUE) %>%

# add column

mutate(bmi = wt\_kg / (ht\_cm/100)^2) %>%

# convert class of columns

mutate(across(contains("date"), as.Date),

generation = as.numeric(generation),

age = as.numeric(age)) %>%

# add column: delay to hospitalisation

mutate(days\_onset\_hosp = as.numeric(date\_hospitalisation - date\_onset)) %>%

# clean values of hospital column

mutate(hospital = recode(hospital,

# OLD = NEW

"Mitylira Hopital" = "Military Hospital",

"Mitylira Hospital" = "Military Hospital",

"Military Hopital" = "Military Hospital",

"Port Hopital" = "Port Hospital",

"Central Hopital" = "Central Hospital",

"other" = "Other",

"St. Marks Maternity Hopital (SMMH)" = "St. Mark's Maternity Hospital (SMMH)"

)) %>%

mutate(hospital = replace\_na(hospital, "Missing")) %>%

# create age\_years column (from age and age\_unit)

mutate(age\_years = case\_when(

age\_unit == "years" ~ age,

age\_unit == "months" ~ age/12,

is.na(age\_unit) ~ age)) %>%

mutate(

# age categories: custom

age\_cat = epikit::age\_categories(age\_years, breakers = c(0, 5, 10, 15, 20, 30, 50, 70)),

# age categories: 0 to 85 by 5s

age\_cat5 = epikit::age\_categories(age\_years, breakers = seq(0, 85, 5))) %>%

filter(

# keep only rows where case\_id is not missing

!is.na(case\_id),

# also filter to keep only the second outbreak

date\_onset > as.Date("2013-06-01") | (is.na(date\_onset) & !hospital %in% c("Hospital A", "Hospital B")))

简化：

df %>%

na.omit()%>% #删除缺少值的行

mutate(across(where(is.numeric), ~replace\_na(., median(., na.rm=TRUE))))%>% #将缺失值替换为另一个值

distinct(.keep\_all=TRUE) #删除重复行

############################################################################################################################################################################################################################################################################################################################################################################################################################################################################