数据集清理

00a-prescriptions

读取和处理PIS OST处方数据，将变量重命名为统一的名称，并保存处理后的数据。

```
# Name of file - 05-PIS_OST_all_prescriptions.R ####

# Original author(s) - Morven Millar

# Original date - 15/12/2022

# Latest update author - Morven Millar

# Latest update date -

# Latest update description -

# Type of script - Data preparation of unaggregated file of PIS OST prescriptions

# Written/run on - R Studio Server

# Version of R - 3.6.1


# Description of content

# Script to call in the unaggregated file of all PIS OST prescriptions from Paid FY 2009/10 - 2021/22.

# 该脚本用于调用2009/10财年至2021/22财年所有PIS OST处方的未聚合文件。

# Approximate run time - <5mins

#  清除环境变量

rm(list = ls())

gc()


ok

ok# install packages

install.packages('tidyverse')

install.packages('dplyr')
```

```
# Load packages
library(tidyverse)
library(dplyr)
#设置工作目录,根据目录找到位置
setwd('')
getwd()
ost <- read_rds('OST_prescriptions.rds')
df01_OST_cohort <- readRDS("/PHI_conf/SubstanceMisuse2/Topics/Prevalence/Opioids/Projects/2025-05-01-FanZhang-MSc-OAT-cocaine/Analysis/PHS_raw_data/cohort_OST_prescriptions.rds")
#Converting Vars to match all files

ost <- ost %>%
  rename(PaidDate = paid_month_end_date,
         PrescrDate = prescribed_full_date,
         DispDate = dispensed_full_date,
         DOB = patient_date_of_birth,
         Gender = patient_sex,
         PatientsResidentsNHSBoard  =  patient_health_board_name,
         ItemCode = item_code)
range(ost$PaidDate)

write_rds(ost,  "refined_data/OST_prescriptions.rds")
```

- 变量重命名:
  - 使用 `dplyr` 包的 `rename` 函数，将 `ost` 数据框中的变量重命名为统一的名称，以便与所有文件匹配。
  - 重命名的变量包括:
    - `PaidDate` （付账日期）
    - `PrescrDate` （处方日期）

    - `DispDate`（发药日期）
    - `DOB`（患者出生日期）
    - `Gender`（患者性别）
    - `PatientsResidentsNHSBoard`（患者所在NHS委员会名称）
    - `ItemCode`（项目代码）
- 检查日期范围: 使用 `range(ost$PaidDate)` 检查 `PaidDate` 变量的范围。
- 保存处理后的数据: 将处理后的 `ost` 数据保存为 `refined_data/OST_prescriptions.rds` 文件。

00b-strong_opioids_short

合并和处理与阿片类药物相关的数据集，标记强阿片类药物，并保存处理后的数据。

输入文件

1. 药物相关死亡（DRDs）数据集：
    - 文件名：`NRS_drug_related_deaths.rds`
    - 用途：包含药物相关死亡的详细信息。代码中将其与额外的案例数据集合并。
2. 额外的药物相关死亡案例数据集：
    - 文件名：`NRS_drug_related_deaths_ADDITIONAL_CASES.rds`
    - 用途：包含额外的药物相关死亡案例，用于扩展主数据集。
3. 强阿片类药物数据集：
    - 文件名：`lookup_strong_opioids.rds`
    - 用途：包含强阿片类药物的相关信息，用于标记主数据集中的强阿片类药物案例。
4. 额外的强阿片类药物数据集：
    - 文件名：`lookup_strong_opioids_ADDITIONAL_CASES.rds`
    - 用途：包含额外的强阿片类药物案例，用于扩展主强阿片类药物数据集。

输出文件

这段代码生成了以下输出文件：

1. 处理后的药物相关死亡数据集：
   - 文件名：`refined_data/allDRDs.rds`
   - 内容：经过变量转换、合并和标记后的药物相关死亡数据集。这个文件包含了主数据集和额外案例数据集的合并结果，以及强阿片类药物的标记。

Code block

```
# 清除环境变量 加载多个R包
rm(list=ls())
library(readr)
library(retroharmonize)
library(data.table)
library(tidyverse)
library(phsmethods)

# 设置工作目录
setwd('')
OpioidCohort <- read_rds("NRS_drug_related_deaths.rds")
ExtraOpioidCohort <- read_rds("NRS_drug_related_deaths_ADDITIONAL_CASES.rds")
# 检查NRSdrd_DOD（死亡日期）的范围。
range(OpioidCohort$NRSdrd_DOD)
# 检查两个数据集的列名是否一致，确保合并。
colnames(OpioidCohort)==colnames(ExtraOpioidCohort)
OpioidCohort <- rbind(OpioidCohort,ExtraOpioidCohort)

# 检查合并
StrongOpioidCohort <- read_rds('lookup_strong_opioids.rds')
ExtraStrongOpioidCohort <-
read_rds('lookup_strong_opioids_ADDITIONAL_CASES.rds')
colnames(StrongOpioidCohort)==colnames(ExtraStrongOpioidCohort)
StrongOpioidCohort <- rbind(StrongOpioidCohort,ExtraStrongOpioidCohort)


# 将OpioidCohort数据集中的IAIN列转换为字符类型。
OpioidCohort <- OpioidCohort %>%
                  mutate(across(c("IAIN"), as.character))

# 将StrongOpioidCohort数据集中的IAIN列转换为字符类型，
# 并添加一个新列strong_opioid_flag，其值为1，用于标记强阿片类药物。
StrongOpioidCohort <- StrongOpioidCohort %>%
                        mutate(across(c("IAIN"), as.character)) %>%
                        mutate(strong_opioid_flag=1)


# 数据合并与标记
OpioidCohort <- OpioidCohort %>%
```

```
                    left_join(StrongOpioidCohort %>% select(IAIN,
strong_opioid_flag)) %>%

mutate(strong_opioid_flag=ifelse(is.na(strong_opioid_flag),0,strong_opioid_flag
))
# 统计标记数量
OpioidCohort %>% count(strong_opioid_flag)

getwd()
write_rds(OpioidCohort, "refined_data/allDRDs.rds")

```

00c-create_deaths_file

处理与药物相关死亡（DRDs）和全因死亡（ACM）的数据集。

输入文件

1. **药物相关死亡（DRDs）数据集**：
   - 文件名：`refined_data/allDRDs.rds`
   - 用途：这是主要的输入数据集，包含了药物相关死亡的详细信息。
2. **全因死亡（ACM）数据集**：
   - 文件名：`lookup_NRS_all_cause_mortality.rds` 和 `lookup_NRS_all_cause_mortality_ADDITIONAL_CASES.rds`
   - 用途：这两个文件包含了全因死亡的数据。代码中将这两个文件合并后，用于进一步的分析和比较。

输出文件

1. **处理后的DRDs数据集**：
   - 文件名：`refined_data/allDRDs.rds`
   - 内容：经过变量重命名、筛选和转换后的DRDs数据集。这个文件包含了财年、年龄等新计算的变量。

2. **死亡计数数据集**：
   - 文件名：`refined_data/death_counts.csv`（注释掉了，未实际生成）

- 内容：包含不同财年、不同条件下的死亡计数。例如，特定财年内的总死亡数、特定类型的死亡数等。

3. **处理后的ACM数据集**：
   - 文件名：`refined_data/ACM_drds.rds`
   - 内容：经过变量重命名、筛选和转换后的ACM数据集。这个文件包含了财年、年龄等新计算的变量。

4. **所有HMMB死亡数据集**：
   - 文件名：`refined_data/allHMMBs.rds`
   - 内容：筛选出的与HMMB（可能是指某种特定药物或药物类别）相关的死亡数据。这个数据集经过了详细的筛选和标记，包括特定的死亡类型、年龄范围、药物标记等。

5. **所有ACM数据集**：
   - 文件名：`refined_data/allACMs.rds`
   - 内容：经过处理后的全因死亡数据集，包含了标记和筛选后的数据。

代码

Code block

```
# Create cohort; alive & with prescriptions in 2011-2020
# Last updated 30/08/2022

# Libraries and data ----------------------------------------------------
rm(list=ls())
library(tidyverse)
library(tidylog)
library(janitor)
library(lubridate)
library(retroharmonize)
library(data.table)
library(phsmethods)


#load DRDs file
setwd('')
getwd()
drds <- read_rds('refined_data/allDRDs.rds')
```

```r
#load secondary codes file

# 指定路径读取药物相关死亡（DRDs）数据集。重命名变量
drds <- drds %>% rename(SMR99_HMMBimplic = NRSdrd_HMMBimplic) %>%
rename(SMR99_DOB = NRSdrd_DOB) %>%
rename(SMR99_Sex = NRSdrd_SEX) %>%
rename(SMR99_DOD = NRSdrd_DOD) %>%
rename(SMR99_HBres = NRSdrd_HBres) %>%
rename(SMR99_UNDERLYING_CAUSE_OF_DEATH = NRSdrd_UNDERLYING_CAUSE_OF_DEATH) %>%
rename(type_of_death = type_of_death) %>%
rename(SMR99_poison = NRSdrd_poison) %>%
rename(SMR99_alsopres = NRSdrd_alsopres)

# 转换变量 选择特定的列。
# 将IAIN列转换为字符类型。
# 使用phsmethods::extract_fin_year函数提取财年。
# 计算年龄。
drds <- drds %>%
          select(IAIN, SMR99_HMMBimplic, SMR99_DOB, SMR99_Sex, SMR99_DOD,
                 SMR99_HBres, SMR99_UNDERLYING_CAUSE_OF_DEATH, type_of_death,
                 SMR99_poison,SMR99_alsopres, strong_opioid_flag) %>%
          mutate(across(c("IAIN"), as.character)) %>%
          mutate(FY := phsmethods::extract_fin_year(lubridate::ymd(SMR99_DOD))) %>%
          mutate(age= floor(as.duration(SMR99_DOB %--% SMR99_DOD) / ddays(365.25)))
range(drds$FY)
range(drds$age)

# 检查数据的范围和唯一值
range(drds$SMR99_DOD)
length(unique(drds$IAIN))
#7385
length(unique(drds$IAIN[which(drds$SMR99_HMMBimplic==1)]))
#5514 (HMMBs)
length(unique(drds$IAIN[which(drds$SMR99_HMMBimplic==1 & drds$type_of_death==3)]))
#4831 (accidental HMMBs)


#write_rds(drds, "refined_data/allDRDs.rds")
##############################################################################
##############################################################################

#  计算特定财年内的死亡计数。
death_counts <- drds %>%
  filter(SMR99_HMMBimplic==1) %>%
```

```r
  filter(FY %in% c('2014/15','2015/16','2016/17','2017/18',
                   '2018/19','2019/20','2020/21','2021/22',
                   '2022/23')) %>%
  count(FY)


death_counts <- death_counts %>%
  rename(n0=n)


death_counts2 <- drds %>%
  filter(SMR99_HMMBimplic==1) %>%
  filter(FY %in% c('2014/15','2015/16','2016/17','2017/18',
                   '2018/19','2019/20','2020/21','2021/22',
                   '2022/23')) %>%
  filter(type_of_death==3) %>%
  count(FY)

# 将不同条件下的死亡计数合并到一个数据框中
death_counts <- death_counts %>%
  left_join(death_counts2 %>% rename(n2=n))

death_counts2 <- drds %>%
  filter(SMR99_HMMBimplic==1) %>%
  filter(FY %in% c('2014/15','2015/16','2016/17','2017/18',
                   '2018/19','2019/20','2020/21','2021/22',
                   '2022/23')) %>%
  filter(type_of_death==3) %>%
  filter(age>=15 & age<65) %>%
#  filter(age>65 & age<70) %>%
  count(FY)

death_counts <- death_counts %>%
  left_join(death_counts2 %>% rename(n3=n))




death_counts2 <- drds %>%
  filter(SMR99_HMMBimplic==1) %>%
  filter(FY %in% c('2014/15','2015/16','2016/17','2017/18',
                   '2018/19','2019/20','2020/21','2021/22',
                   '2022/23')) %>%
  filter(type_of_death==3) %>%
  filter(age>=15 & age<65) %>%
#  filter(age>65 & age<70) %>%
    filter(SMR99_UNDERLYING_CAUSE_OF_DEATH %in% c('X42','F112','F192')) %>%
```

```
  count(FY)

death_counts <- death_counts %>%
  left_join(death_counts2 %>% rename(n4=n))



death_counts2 <- drds %>%
  filter(SMR99_HMMBimplic==1) %>%
  filter(FY %in% c('2014/15','2015/16','2016/17','2017/18',
                   '2018/19','2019/20','2020/21','2021/22',
                   '2022/23')) %>%
  filter(type_of_death==3) %>%
  filter(age>=15 & age<65) %>%
#  filter(age>65 & age<70) %>%
  filter(SMR99_UNDERLYING_CAUSE_OF_DEATH %in% c('X42','F112','F192')) %>%
  filter(strong_opioid_flag==0) %>%
  count(FY)


death_counts <- death_counts %>%
  left_join(death_counts2 %>% rename(n5=n))


#write.csv(death_counts, "refined_data/death_counts.csv")

#####################################################
#####################################################
#####################################################
#####################################################
#####################################################
#####################################################
#####################################################
#####################################################
#####################################################
#####################################################
#####################################################
# add ACM ----


ACM <- read_rds("lookup_NRS_all_cause_mortality.rds")
ExtraACM <- read_rds("lookup_NRS_all_cause_mortality_ADDITIONAL_CASES.rds")
colnames(ACM)==colnames(ExtraACM)
ACM <- rbind(ACM,ExtraACM)

colnames(ACM) <- c('rowid','IAIN', 'SMR99_DOB', 'SMR99_DOD', 'SMR99_Sex',
```

```
                    'record_type','date_of_registration',
                    'SMR99_UNDERLYING_CAUSE_OF_DEATH',
                    'SMR99_CAUSE_OF_DEATH_CODE_0',
                    'SMR99_CAUSE_OF_DEATH_CODE_1',
                    'SMR99_CAUSE_OF_DEATH_CODE_2',
                    'SMR99_CAUSE_OF_DEATH_CODE_3',
                    'SMR99_CAUSE_OF_DEATH_CODE_4',
                    'SMR99_CAUSE_OF_DEATH_CODE_5',
                    'SMR99_CAUSE_OF_DEATH_CODE_6',
                    'SMR99_CAUSE_OF_DEATH_CODE_7',
                    'SMR99_CAUSE_OF_DEATH_CODE_8',
                    'SMR99_CAUSE_OF_DEATH_CODE_9',
                    'SMR99_HBres','SMR99_Council_Area',
                    'SMR99_HBres_current','SMR99_Council_Area_current',
                    'SMR992_flag')




ACM <- ACM %>%
  select(IAIN, SMR99_DOB, SMR99_Sex, SMR99_DOD,
         SMR99_HBres, SMR99_UNDERLYING_CAUSE_OF_DEATH, ) %>%
  mutate(across(c("IAIN"), as.character)) %>%
  mutate(FY := phsmethods::extract_fin_year(lubridate::ymd(SMR99_DOD))) %>%
  mutate(age_at_death = floor(as.duration(SMR99_DOB %--% SMR99_DOD) /
ddays(365.25)))



# Check overlap between ACM and DRDs 重叠?

length(unique(ACM$IAIN))

length(intersect(unique(ACM$IAIN),unique(drds$IAIN)))


#write_rds(ACM,  "refined_data/ACM_drds.rds")
#筛选年龄在15-64岁、有具体原因且无强效阿片类药物的HMMB意外死亡

all_HMMB_deaths <- drds[which(drds$SMR99_HMMBimplic==1),]



allHMMBs <- all_HMMB_deaths %>%
  filter(type_of_death==3) %>%
  mutate(FY := phsmethods::extract_fin_year(lubridate::ymd(SMR99_DOD))) %>%
```

```r
  mutate(age_at_death = floor(as.duration(SMR99_DOB %--% SMR99_DOD) / ddays(365.25))) %>%
  filter(age_at_death>=15 & age_at_death<=64) %>%
  mutate(type_of_death_long = case_when(
    type_of_death==3 ~ "Accidental",
    type_of_death==1 ~ "Suicide",
    type_of_death==2 ~ "Undetermined")) %>%
  mutate(age_group = case_when(age_at_death>=15 & age_at_death<35 ~ '15-34',
                               age_at_death>=35 & age_at_death<50 ~ '35-49',
                               age_at_death>=50 & age_at_death<65 ~ '50-64')) %>%
  mutate(across(c("IAIN"), as.character)) %>%
  mutate(sex= ifelse(SMR99_Sex==1, 'M','F')) %>%
  filter(strong_opioid_flag==0) %>%
  filter(SMR99_UNDERLYING_CAUSE_OF_DEATH %in% c('X42','F112','F192')) %>%
  mutate(day=SMR99_DOD) %>%
  mutate(DOB=SMR99_DOB) %>%
  mutate(hmmb_flag=1) %>%
  filter(type_of_death==3)



#allHMMBs <- allHMMBs %>%
#filter(FY %in% c('2014/15','2015/16','2016/17',
#                 '2017/18','2018/19','2019/20',
#                 '2020/21','2021/22','2022/23'))
#allHMMBs <- allHMMBs %>% filter(strong_opioid_flag==0)


# Prepare all-cause mortality dataset for export

all_deaths <- read_rds("refined_data/ACM_drds.rds")
all_deaths <- all_deaths %>%
  mutate(across(c("IAIN"), as.character))

allACM <- all_deaths %>%
  mutate(DOB=SMR99_DOB) %>%
  mutate(day=SMR99_DOD) %>%
  mutate(acm_flag=1) %>%
  select(IAIN,day,DOB,acm_flag)


write_rds(allHMMBs, "refined_data/allHMMBs.rds")
write_rds(allACM, "refined_data/allACMs.rds")
```

00d-Hospital_Admissions

输入文件

- `SMR0104_drug_related_hosp_stays.rds`：这是一个RDS文件，包含与药物相关的医院住院数据。该文件被加载到 `hospital_admissions` 变量中。

输出文件

- `refined_data/hospitalisations_T.rds`：这是一个RDS文件，包含经过**筛选和处理后的与特定ICD-10代码（T400, T401, T403）相关的住院数据**。该文件存储在 `refined_data` 目录中。

数据加载和预处理

1. **加载必要的库**：

2. **设置工作目录**：
   - 使用 `setwd('')` 设置工作目录（需要用户指定路径）。
   - 使用 `getwd()` 确认当前工作目录。

3. **加载数据**：
   - 使用 `read_rds` 函数加载 `SMR0104_drug_related_hosp_stays.rds` 文件到 `hospital_admissions` 变量中。

4. **数据预处理**：
   - 检查 `ADMISSION_DATE` 和 `flag_discharge_death` 字段的范围。
   - 将 `IAIN` 字段转换为字符类型。
   - 提取财政年度（`FY`）。
   - 计算入院时的年龄（`age_at_entry`）。

- 过滤掉 `flag_discharge_death` 为1的记录（即死亡记录）。
- 过滤年龄在15到64岁之间的记录。
- 去重（`distinct`）。

数据筛选和统计

1. **统计财政年度的记录数**：
   - 使用 `count(FY)` 统计每个财政年度的记录数。

2. **筛选特定ICD-10代码的记录**：
   - 筛选 `RECORD_TYPE` 为 `01B` 的记录。
   - 筛选主诊断或次要诊断为 `T400`，`T401`，`T403` 的记录。
   - 过滤年龄在15到64岁之间的记录。
   - 过滤掉 `flag_discharge_death` 为1的记录。

3. **统计性别分布**：
   - 对2014/15到2021/22财政年度的性别分布进行统计。
   - 对2014/15到2022/23财政年度的性别分布进行统计。

数据保存

1. **保存处理后的数据**：
   - 使用 `write_rds` 函数将处理后的数据保存到 `refined_data/hospitalisations_T.rds` 文件中。

总结

这段代码的主要功能是：

1. 加载和预处理与药物相关的医院住院数据。
2. 筛选出特定ICD-10代码（T400, T401, T403）的记录。
3. 统计不同财政年度和性别的记录数。
4. 保存处理后的数据到指定文件中。

代码

Code block

```
# Split dataset by age_grp and year_grp
# Calculate CMRs for on/off by year_grp, and by age_grp and year_grp
# Plot figures

# Libraries and data ----------------------------------------------------
rm(list=ls())
library(tidyverse)
library(tidylog)
library(lubridate)
library(glue)
library(data.table)
library(janitor)
#library(epitools)
#library(haven)
library(xtable)
library(phsmethods)


setwd('')
getwd()
hospital_admissions <- read_rds('SMR0104_drug_related_hosp_stays.rds')


# Check date ranges and discharge death flag

range(hospital_admissions$ADMISSION_DATE)
range(hospital_admissions$flag_discharge_death)


# Prepare dataset: convert ID to character, extract financial year, calculate age at entry

hospital_admissions <- hospital_admissions %>%
  mutate(across(c("IAIN"), as.character)) %>%
  mutate(FY := phsmethods::extract_fin_year(lubridate::ymd(ADMISSION_DATE))) %>%
  mutate(age_at_entry = floor(as.duration(DOB %--% ADMISSION_DATE) / ddays(365.25))) %>%
  filter(flag_discharge_death==0)
  #mutate(age_at_entry = round(as.duration(DOB %--% ADMISSION_DATE) / ddays(365.25)))
```

```r


# Count admissions by financial year

hospital_admissions %>% count(FY)


# Filter for working-age adults (15-64) and remove duplicates

hospital_admissions <- hospital_admissions %>%
                    filter(age_at_entry>=15 & age_at_entry<=64) %>%
                    filter(flag_discharge_death==0)


# Count by record type with totals

hospital_admissions %>% count(RECORD_TYPE) %>% adorn_totals()
hospital_admissions <- hospital_admissions %>% distinct()

#length(which(hospital_admissions$EPI_1_MAIN_CONDITION=='F112'))


# Count T-code admissions by sex for selected financial years

hosp_T_codes <- hospital_admissions %>%
  filter(RECORD_TYPE %in% c('01B')) %>%
  filter(EPI_1_MAIN_CONDITION %in% c('T400','T401','T403') |
           EPI_1_OTHER_CONDITION_1 %in% c('T400','T401','T403') |
           EPI_1_OTHER_CONDITION_2 %in% c('T400','T401','T403') |
           EPI_1_OTHER_CONDITION_3 %in% c('T400','T401','T403') |
           EPI_1_OTHER_CONDITION_4 %in% c('T400','T401','T403') |
           EPI_1_OTHER_CONDITION_5 %in% c('T400','T401','T403')) %>%
  filter(age_at_entry>=15 & age_at_entry<=64) %>%
  filter(flag_discharge_death==0)

hosp_T_codes  %>%
  filter(FY %in% c('2014/15','2015/16','2016/17','2017/18',
                   '2018/19','2019/20','2020/21','2021/22')) %>%
  count(SEX) %>% adorn_totals()

hosp_T_codes  %>%
  filter(FY %in% c('2014/15','2015/16','2016/17','2017/18',
                   '2018/19','2019/20','2020/21','2021/22','2022/23')) %>%
  count(SEX) %>% adorn_totals()

 write_rds(hosp_T_codes, "refined_data/hospitalisations_T.rds")


```

00e_create_demographics

输入文件

- `/OST_prescriptions.rds`：这是一个RDS文件，包含阿片类药物替代治疗（ORT）的处方数据。该文件被加载到 `orts` 变量中。

输出文件

- `refined_data/demographics.rds`：这是一个RDS文件，包含经过处理和合并后的人口统计学数据（包括性别、出生日期和NHS健康委员会信息）。该文件存储在 `refined_data` 目录中。

代码

2️⃣ 读入原始处方数据（ORT）

Code block

```
setwd('')                          # 设置项目工作目录（已留空，需手动指定）
orts <- readRDS(.../filtered_OST_prescriptions.rds)  # 读入 2011-2020 的 OST 处方
orts <- orts %>% mutate(across("IAIN", as.character))  # 保证 ID 为字符型
```

- **注意：** `setwd('')` 需改成实际路径，否则后续读写会报错。

3️⃣ 把日期列统一转成 `Date`

Code block

```
z_orts <- orts %>%
  mutate_at(vars(contains("Date")), ~as.Date(., "%Y/%m/%d %H:%M:%S"))
z_orts <- z_orts %>% mutate(IAIN = as.character(IAIN))
```

- **目的**：所有含 `Date` 的列（如 `PaidDate`、`PrescriptionStartDate` 等）从字符转成真正的日期，便于后续按时间筛选与排序。

4️⃣ Health Board（HB）名称转换函数

```
hb_recode <- function(hb_in, recode_to = "long", and_choice = "and") { ... }
```

- **功能**：把 NHS Health Board 的代码（如 `S08000007`）或简写（如 `GC`）映射为 **完整名称**（`Greater Glasgow and Clyde`）或 **简写**（`GC`）。
- **参数**：
  - `recode_to = "long"` → 返回完整名称（如 `"Greater Glasgow and Clyde"`）
  - `recode_to = "short"` → 返回两字母简写（如 `"GC"`）
- **注意**：函数里用 `case_when` 做了大量手工映射，保证新旧编码兼容。

5️⃣ 提取人口学变量（Gender + DOB）

```
demographics <- orts %>% select(IAIN, Gender, DOB)
demographics <- demographics %>% tidylog::distinct()    # 去重
demographics <- demographics %>%
  mutate(DOB = as.Date(DOB, "%Y/%m/%d %H:%M:%S")) %>%
  mutate(IAIN = as.character(IAIN))
```

- **目的**：把 `Gender`（1=M, 2=F）和出生日期单独存一份，后面与死亡、HB 合并。

6️⃣ 为每个患者每年生成“最近居住 HB”

6-a）先拿 **按年+人** 的最近记录

Code block

```
NHSboards2 <- orts %>%
  mutate(year = year(PaidDate)) %>%
  select(IAIN, PatientsResidentsNHSBoard, PaidDate, year) %>%
  filter(!is.na(PatientsResidentsNHSBoard)) %>%
  group_by(IAIN, year) %>%
  arrange(desc(PaidDate)) %>%     # 同一年里按处方日期倒序
  slice(1) %>%                    # 取最近一条处方
  ungroup() %>% distinct()
```

- **结果**：得到 **每人每年** 的最新 HB 代码（用于后续时间序列插补）。

6-b）再拿 **全时期** 的最近 HB（备用）

Code block

```
NHSboards <- orts %>%
  select(IAIN, PatientsResidentsNHSBoard, PaidDate) %>%
  filter(!is.na(...)) %>%
  group_by(IAIN) %>% arrange(desc(PaidDate)) %>% slice(1) %>%
  ungroup() %>% mutate(year = year(PaidDate)) %>% distinct()
```

- **用途**：做稳健性检查，也可作为“缺失年份”的 fallback。

7️⃣ 把 HB 缺失年份 **向前/向后填充**

Code block

```
n_years <- length(2009:2024)   # 16 年
n_IAIN  <- length(unique(NHSboards2$IAIN))

NHSBoards_by_year <- expand_grid(
  IAIN = unique(NHSboards2$IAIN),
  year = 2009:2024
```

```r
  ) %>% left_join(NHSboards2)          # 先左连，很多缺失

  # 用 locf 两次：先向后、再向前填充
  NHSBoards_by_year2 <- NHSBoards_by_year %>%
    arrange(IAIN, year) %>%
    group_by(IAIN) %>%
    mutate(PatientsResidentsNHSBoard = zoo::na.locf(PatientsResidentsNHSBoard, na.rm = FALSE, fromLast = TRUE)) %>%
    mutate(PatientsResidentsNHSBoard = zoo::na.locf(PatientsResidentsNHSBoard, na.rm = FALSE, fromLast = FALSE)) %>%
    ungroup()
```

- **目的**：即使某年没有处方，也能用 **最近已知 HB** 填充，确保 2009–2024 每年每人都有值。

8️⃣ 合并人口学 + HB，并保存

Code block

```r
demographics_and_NHSBoard2 <- NHSBoards_by_year2 %>%
  select(IAIN, PatientsResidentsNHSBoard, year) %>%
  left_join(demographics) %>% distinct()

demographics_and_NHSBoard <- demographics %>%
  left_join(NHSboards %>% select(IAIN, PatientsResidentsNHSBoard))

# 写盘
write_rds(demographics_and_NHSBoard, "refined_data/demographics.rds")
```

- **最终产物**：`demographics.rds` 包含 **每人每年的**
  - IAIN
  - Gender
  - DOB
  - 最近居住 NHS Board（已插补）