内在能力建模

Ruifu Kang & Yirou Niu

2024-12-26

# 建模步骤

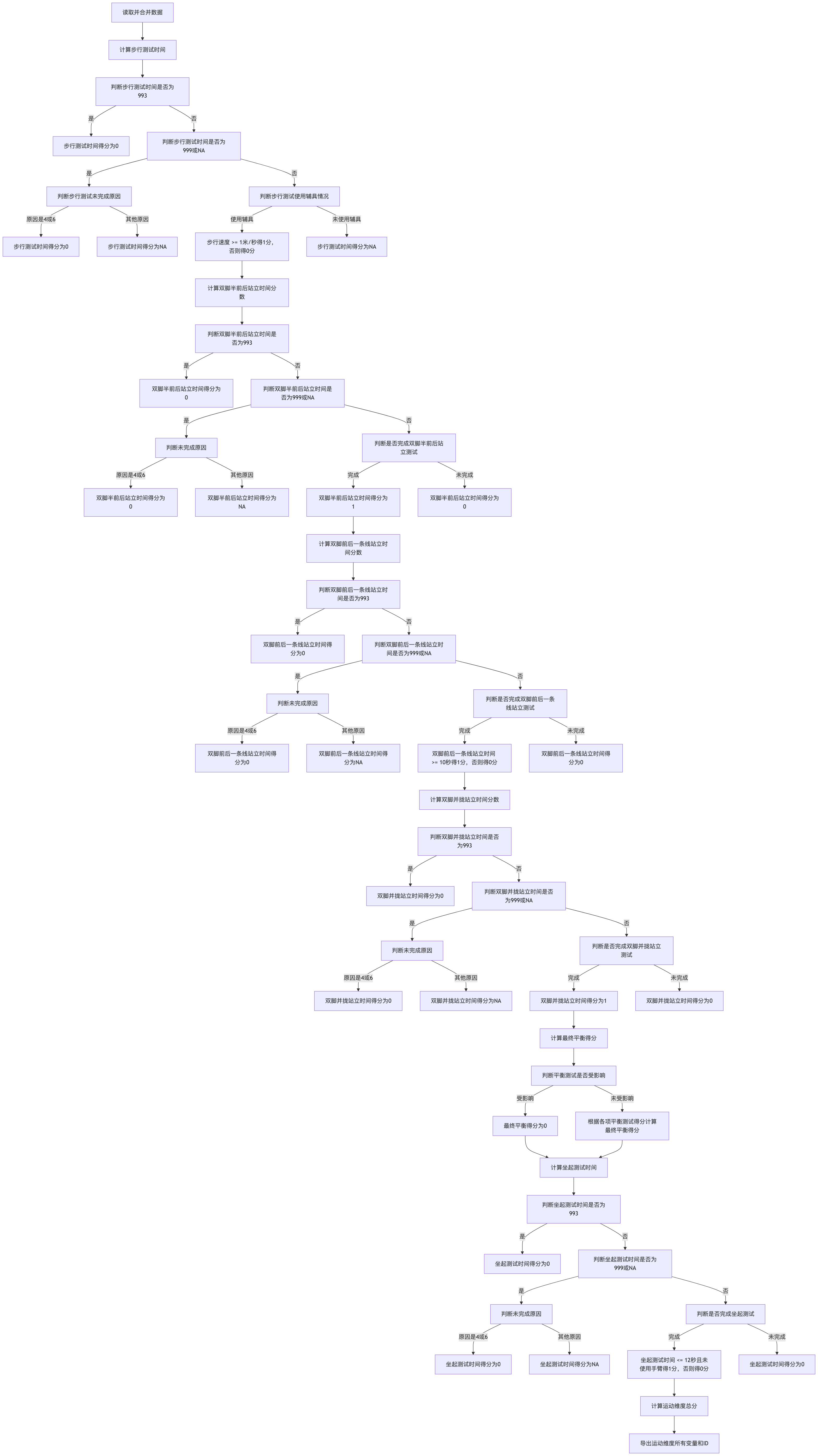
## 数据初步清洗

1. 读取并合并数据
2. 变量重命名：内在能力及其相关变量，包括出生年份、性别等都重命名
3. 年龄数据清洗：去掉非老年人的个体之后9982人

## 内在能力赋值

### 1. 运动

* 2.5米步行速度≥ 1米/秒得1分；
* 重复坐下5次≤12秒得1分；
* 平衡：3个10秒完成2个及以上得1分。
* 满分3分



### 2. 认知：

* 情景记忆（基于延迟回忆得分）：0-10分；
* 减7测试：5分；
* 日期、月份、年份和季节：5分；
* 绘画：1分。
* 满分21分
* 进一步：范围18-20：得3分；范围14-17：得2分；范围7-13：得1分；范围0-6：得0分。
* 满分3分

### 3. 心理

* CES-D评分为0到9分：得1分；
* 总睡眠时间在5到10.5小时之间，得1分；
* 睡眠质量：一周内睡眠不安的频率0到2天之间得1分。
* 满分3分

### 4. 感官

* 听力回答非常好、好、一般：得1分；
* 视力回答非常好、好、一般：得1分（远和近两项）
* 满分3分

### 5. 活力

* 握力：男性≥ 35kg得1分，女性≥ 25kg得1分；
* FEV：男性≥ 400得1分，女性≥ 290得1分;
* 血红蛋白：男性≥ 120g/L得1分，女性≥ 110g/L得1分。
* 满分3分

## 内在能力筛选

* 根据内在能力计算情况筛选出覆盖内在能力指标范围大于100%的个体
* 保存筛选后的数据到一个新的数据框中

## 自变量筛选与清洗

## 数据拆分

## 模型构建

## 模型验证

# 代码

## 读取并合并数据

# 安装并加载所需要的包  
if (!requireNamespace("haven", quietly = TRUE)) {  
 install.packages("haven")  
}  
if (!requireNamespace("dplyr", quietly = TRUE)) {  
 install.packages("dplyr")  
}  
if (!requireNamespace("ggplot2", quietly = TRUE)) {  
 install.packages("ggplot2")  
}  
library(haven)  
library(dplyr)

Attaching package: 'dplyr'

The following objects are masked from 'package:stats':  
  
 filter, lag

The following objects are masked from 'package:base':  
  
 intersect, setdiff, setequal, union

library(ggplot2)  
  
# 读取并合并数据  
files <- list.files("data\_raw/self",  
 pattern = "\\.dta$",  
 full.names = TRUE  
)  
data\_list <- lapply(files, read\_dta)  
my\_data\_raw <- Reduce(function(x, y) full\_join(x, y, by = "ID"), data\_list)  
print(my\_data\_raw)

# A tibble: 21,805 × 7,595  
 ID householdID.x communityID.x pa001 pa002 qa001s1 qa001s2 qa001s3  
 <chr> <chr> <chr> <dbl+l> <dbl+l> <dbl+l> <dbl+l> <dbl+l>  
 1 09400410… 0940041030 0940041 5 [5 N… 1 [1 Y… NA NA NA   
 2 09400411… 0940041100 0940041 5 [5 N… 1 [1 Y… NA NA NA   
 3 09400410… 0940041080 0940041 5 [5 N… 1 [1 Y… NA NA NA   
 4 09400411… 0940041120 0940041 5 [5 N… 1 [1 Y… NA NA NA   
 5 09400411… 0940041120 0940041 5 [5 N… 1 [1 Y… NA NA NA   
 6 09400411… 0940041140 0940041 5 [5 N… 1 [1 Y… NA NA NA   
 7 09400411… 0940041190 0940041 5 [5 N… 1 [1 Y… NA NA NA   
 8 09400411… 0940041170 0940041 5 [5 N… 1 [1 Y… NA NA NA   
 9 09400411… 0940041170 0940041 5 [5 N… 1 [1 Y… NA NA NA   
10 09400431… 0940043100 0940043 5 [5 N… 1 [1 Y… NA NA NA   
# ℹ 21,795 more rows  
# ℹ 7,587 more variables: qa001s4 <dbl+lbl>, qa001s5 <dbl+lbl>,  
# qa001s6 <dbl+lbl>, qa001s7 <dbl+lbl>, qa001s8 <dbl+lbl>,  
# qa001s97 <dbl+lbl>, qa002 <chr>, qa002\_1 <chr>, qa003 <dbl>, qa004 <dbl>,  
# qa005 <dbl>, qa006 <chr>, qa006\_1 <chr>, qa007 <dbl>, qa008 <dbl>,  
# qa009 <dbl>, qa010 <chr>, qa010\_1 <chr>, qa011 <dbl>, qa012 <dbl>,  
# qa013 <dbl>, qa014 <dbl+lbl>, qa015 <dbl+lbl>, qa016 <dbl+lbl>, …

## 变量重命名

# 变量重命名  
my\_data\_rename <- my\_data\_raw %>%  
 rename(  
 # 基本信息重命名  
 id\_birth\_year = ba004\_w3\_1, # ID出生年份  
 actual\_birth\_year = ba002\_1, # 实际出生年份   
 death = died, # 是否死亡  
 gender = ba000\_w2\_3, # 性别  
  
 # 内在能力运动维度重命名  
 balance\_test\_affected = pd001, # 是否影响完成平衡测试  
 understand\_and\_willing\_semi\_tandem = pd002, # 是否明白且愿意参加双脚前后半站立测试  
 stand\_test\_semi\_tandem\_reason\_1 = qd001s1, # 未完成双脚半前后站立测试的第一个原因  
 stand\_test\_semi\_tandem\_reason\_2 = qd001s2, # 未完成双脚半前后站立测试的第二个原因  
 stand\_test\_semi\_tandem\_reason\_3 = qd001s3, # 未完成双脚半前后站立测试的第三个原因  
 stand\_test\_semi\_tandem\_reason\_4 = qd001s4, # 未完成双脚半前后站立测试的第四个原因  
 stand\_test\_semi\_tandem\_reason\_5 = qd001s5, # 未完成双脚半前后站立测试的第五个原因  
 stand\_test\_semi\_tandem\_reason\_6 = qd001s6, # 未完成双脚半前后站立测试的第六个原因  
 stand\_test\_semi\_tandem\_reason\_7 = qd001s7, # 未完成双脚半前后站立测试的第七个原因  
 stand\_test\_semi\_tandem\_reason\_8 = qd001s8, # 未完成双脚半前后站立测试的第八个原因  
 stand\_test\_semi\_tandem\_reason\_other = qd001s97, # 未完成双脚半前后站立测试的其他原因  
 stand\_test\_semi\_tandem = qd002, # 是否完成双脚半前后站立测试  
  
 understand\_and\_willing\_tandem = pe002, # 是否明白且愿意参加双脚一条线站立测试  
 stand\_test\_tandem\_reason\_1 = qe001s1, # 未完成双脚一条线站立测试的第一个原因  
 stand\_test\_tandem\_reason\_2 = qe001s2, # 未完成双脚一条线站立测试的第二个原因  
 stand\_test\_tandem\_reason\_3 = qe001s3, # 未完成双脚一条线站立测试的第三个原因  
 stand\_test\_tandem\_reason\_4 = qe001s4, # 未完成双脚一条线站立测试的第四个原因  
 stand\_test\_tandem\_reason\_5 = qe001s5, # 未完成双脚一条线站立测试的第五个原因  
 stand\_test\_tandem\_reason\_6 = qe001s6, # 未完成双脚一条线站立测试的第六个原因  
 stand\_test\_tandem\_reason\_7 = qe001s7, # 未完成双脚一条线站立测试的第七个原因  
 stand\_test\_tandem\_reason\_8 = qe001s8, # 未完成双脚一条线站立测试的第八个原因  
 stand\_test\_tandem\_reason\_other = qe001s97, # 未完成双脚一条线站立测试的其他原因  
 stand\_test\_tandem = qe002, # 双脚一条线站立测试  
 stand\_test\_tandem\_time = qe003, # 双脚一条线站立测试时间  
  
 understand\_and\_willing\_feet\_together = pf001, # 是否明白且愿意参加双脚并拢站立测试  
 stand\_test\_feet\_together\_reason\_1 = qf001s1, # 未完成双脚并拢站立测试的第一个原因  
 stand\_test\_feet\_together\_reason\_2 = qf001s2, # 未完成双脚并拢站立测试的第二个原因  
 stand\_test\_feet\_together\_reason\_3 = qf001s3, # 未完成双脚并拢站立测试的第三个原因  
 stand\_test\_feet\_together\_reason\_4 = qf001s4, # 未完成双脚并拢站立测试的第四个原因  
 stand\_test\_feet\_together\_reason\_5 = qf001s5, # 未完成双脚并拢站立测试的第五个原因  
 stand\_test\_feet\_together\_reason\_6 = qf001s6, # 未完成双脚并拢站立测试的第六个原因  
 stand\_test\_feet\_together\_reason\_7 = qf001s7, # 未完成双脚并拢站立测试的第七个原因  
 stand\_test\_feet\_together\_reason\_8 = qf001s8, # 未完成双脚并拢站立测试的第八个原因  
 stand\_test\_feet\_together\_reason\_other = qf001s97, # 未完成双脚并拢站立测试的其他原因  
 stand\_test\_feet\_together = qf002, # 双脚并拢站立测试  
  
 walk\_test\_reason\_1 = qg001s1, # 未完成步行测试的第一个原因  
 walk\_test\_reason\_2 = qg001s2, # 未完成步行测试的第二个原因  
 walk\_test\_reason\_3 = qg001s3, # 未完成步行测试的第三个原因  
 walk\_test\_reason\_4 = qg001s4, # 未完成步行测试的第四个原因  
 walk\_test\_reason\_5 = qg001s5, # 未完成步行测试的第五个原因  
 walk\_test\_reason\_6 = qg001s6, # 未完成步行测试的第六个原因  
 walk\_test\_reason\_7 = qg001s7, # 未完成步行测试的第七个原因  
 walk\_test\_reason\_8 = qg001s8, # 未完成步行测试的第八个原因  
 walk\_test\_reason\_other = qg001s97, # 未完成步行测试的其他原因  
 walk\_time\_first = qg002, # 步行测试时间1  
 walk\_time\_second = qg003, # 步行测试时间2  
 walk\_test\_aid = qg005, # 步行测试使用辅具情况  
  
 sit\_stand\_test\_reason\_1 = qh001s1, # 未完成五次坐下起来测试的第一个原因  
 sit\_stand\_test\_reason\_2 = qh001s2, # 未完成五次坐下起来测试的第二个原因  
 sit\_stand\_test\_reason\_3 = qh001s3, # 未完成五次坐下起来测试的第三个原因  
 sit\_stand\_test\_reason\_4 = qh001s4, # 未完成五次坐下起来测试的第四个原因  
 sit\_stand\_test\_reason\_5 = qh001s5, # 未完成五次坐下起来测试的第五个原因  
 sit\_stand\_test\_reason\_6 = qh001s6, # 未完成五次坐下起来测试的第六个原因  
 sit\_stand\_test\_reason\_7 = qh001s7, # 未完成五次坐下起来测试的第七个原因  
 sit\_stand\_test\_reason\_8 = qh001s8, # 未完成五次坐下起来测试的第八个原因  
 sit\_stand\_test\_reason\_other = qh001s97, # 未完成五次坐下起来测试的其他原因  
 sit\_stand\_test\_completed = qh002, # 是否完成五次坐下起来测试  
 sit\_stand\_test\_time = qh003, # 完成五次坐下起来测试的时间  
 sit\_stand\_test\_arm\_use = qh007, # 坐起测试中使用手臂的情况  
  
   
  
 # 内在能力认知维度重命名  
 # 内在能力认知维度重命名  
 proxy\_answer = db032, # 是否由别人代答  
 recall\_word\_1 = dc006s1, # 回忆词汇1  
 recall\_word\_2 = dc006s2, # 回忆词汇2  
 recall\_word\_3 = dc006s3, # 回忆词汇3  
 recall\_word\_4 = dc006s4, # 回忆词汇4  
 recall\_word\_5 = dc006s5, # 回忆词汇5  
 recall\_word\_6 = dc006s6, # 回忆词汇6  
 recall\_word\_7 = dc006s7, # 回忆词汇7  
 recall\_word\_8 = dc006s8, # 回忆词汇8  
 recall\_word\_9 = dc006s9, # 回忆词汇9  
 recall\_word\_10 = dc006s10, # 回忆词汇10  
 recall\_none = dc006s11, # 是否一个都没回忆起来  
 recall\_refused = dc006s12, # 是否拒绝回忆  
  
 recall\_year = dc001s1, # 回忆年份  
 recall\_month = dc001s2, # 回忆月份  
 recall\_day = dc001s3, # 回忆日期  
 recall\_weekday = dc002, # 回忆星期  
 recall\_season = dc003, # 回忆季节  
  
 subtraction\_test\_1 = dc019, # 减法测试1  
 subtraction\_test\_2 = dc020, # 减法测试2  
 subtraction\_test\_3 = dc021, # 减法测试3  
 subtraction\_test\_4 = dc022, # 减法测试4  
 subtraction\_test\_5 = dc023, # 减法测试5  
 subtraction\_tool\_use = dc024, # 是否使用工具计算  
  
 drawing\_test = dc025, # 画图测试  
  
 # 内在能力心理维度重命名  
 depression\_scale\_1 = dc009, # 心理量表问题1  
 depression\_scale\_2 = dc010, # 心理量表问题2  
 depression\_scale\_3 = dc011, # 心理量表问题3  
 depression\_scale\_4 = dc012, # 心理量表问题4  
 depression\_scale\_5 = dc013, # 心理量表问题5  
 depression\_scale\_6 = dc014, # 心理量表问题6  
 depression\_scale\_7 = dc016, # 心理量表问题7  
 depression\_scale\_8 = dc017, # 心理量表问题8  
 depression\_scale\_9 = dc018, # 心理量表问题9  
 night\_sleep\_time = da049, # 夜晚睡眠时间  
 poor\_sleep\_frequency = dc015, # 睡眠不佳频率  
 nap\_time = da050, # 午睡时间  
  
 # 内在能力感官维度重命名  
 wearing\_glasses = da032, # 是否佩戴眼镜  
 far\_vision = da033, # 看远处视力情况  
 near\_vision = da034, # 看近处视力情况  
 hearing\_aid = da038, # 是否佩戴助听器  
 hearing\_status = da039, # 听力情况  
   
   
 # 内在能力活力维度重命名  
 grip\_test\_reason\_1 = qc001s1, # 未完成握力测量的第一个原因  
 grip\_test\_reason\_2 = qc001s2, # 未完成握力测量的第二个原因  
 grip\_test\_reason\_3 = qc001s3, # 未完成握力测量的第三个原因  
 grip\_test\_reason\_4 = qc001s4, # 未完成握力测量的第四个原因  
 grip\_test\_reason\_5 = qc001s5, # 未完成握力测量的第五个原因  
 grip\_test\_reason\_6 = qc001s6, # 未完成握力测量的第六个原因  
 grip\_test\_reason\_7 = qc001s7, # 未完成握力测量的第七个原因  
 grip\_test\_reason\_8 = qc001s8, # 未完成握力测量的第八个原因  
 grip\_test\_reason\_other = qc001s97, # 未完成握力测量的其他原因  
 left\_hand\_grip\_1 = qc003, # 第一次左手握力测量  
 right\_hand\_grip\_1 = qc004, # 第一次右手握力测量  
 left\_hand\_grip\_2 = qc005, # 第二次左手握力测量  
 right\_hand\_grip\_2 = qc006, # 第二次右手握力测量  
  
 breath\_test\_reason\_1 = qb001s1, # 未完成呼吸测试的第一个原因  
 breath\_test\_reason\_2 = qb001s2, # 未完成呼吸测试的第二个原因  
 breath\_test\_reason\_3 = qb001s3, # 未完成呼吸测试的第三个原因  
 breath\_test\_reason\_4 = qb001s4, # 未完成呼吸测试的第四个原因  
 breath\_test\_reason\_5 = qb001s5, # 未完成呼吸测试的第五个原因  
 breath\_test\_reason\_6 = qb001s6, # 未完成呼吸测试的第六个原因  
 breath\_test\_reason\_7 = qb001s7, # 未完成呼吸测试的第七个原因  
 breath\_test\_reason\_8 = qb001s8, # 未完成呼吸测试的第八个原因  
 breath\_test\_reason\_other = qb001s97, # 未完成呼吸测试的其他原因  
 breath\_test\_1 = qb002, # 第一次呼吸功能测定  
 breath\_test\_2 = qb003, # 第二次呼吸功能测定  
 breath\_test\_3 = qb004, # 第三次呼吸功能测定  
  
 hemoglobin = bl\_hgb # 血红蛋白含量  
 )  
 # 导出包含所有重命名列的数据  
 renamed\_columns <- names(my\_data\_rename)  
 original\_columns <- names(my\_data\_raw)  
 renamed\_only\_columns <- setdiff(renamed\_columns, original\_columns)  
 my\_data\_renamed\_only <- my\_data\_rename %>%  
 select(all\_of(renamed\_only\_columns))  
 write.csv(my\_data\_renamed\_only, "renamed\_data.csv", row.names = FALSE)

## 年龄数据清洗

# 去除实际出生年份和ID出生年份同时缺失的数据  
my\_data\_year\_na <- my\_data\_rename %>%  
 filter(!(is.na(actual\_birth\_year) & is.na(id\_birth\_year)))  
  
# 计算年龄值  
my\_data\_age <- my\_data\_year\_na %>%  
 mutate(age = 2015 - coalesce(actual\_birth\_year, id\_birth\_year))  
  
# 年龄清洗  
my\_data\_elder <- my\_data\_age %>%  
 filter(age >= 60)  
 # 输出现在总共多少人  
total\_individuals <- nrow(my\_data\_elder)  
print(paste("总共人数:", total\_individuals))

[1] "总共人数: 9982"

my\_data\_wash <- my\_data\_elder

## 内在能力分数计算

### 运动维度计算（无误）

# 计算步行测试时间（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_walk\_time = case\_when(  
 walk\_time\_first %in% c(993) ~ 0,  
 walk\_time\_first %in% c(999, NA) ~ case\_when(  
 rowSums(select(., starts\_with("walk\_test\_reason\_")) == 4, na.rm = TRUE) > 0 ~ 0,  
 rowSums(select(., starts\_with("walk\_test\_reason\_")) == 6, na.rm = TRUE) > 0 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 TRUE ~ case\_when(  
 walk\_test\_aid == 1 ~ if\_else(2.5 / pmin(walk\_time\_first, walk\_time\_second, na.rm = TRUE) >= 1, 1, 0),  
 walk\_test\_aid %in% c(2, 3, 4, 97) ~ 0,  
 TRUE ~ NA\_real\_  
 )  
 )  
 )  
  
# 统计最终步行时间分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_walk\_time, useNA = "always")

0 1 <NA>   
6252 1714 2016

# 计算平衡分数  
  
# 计算双脚半前后站立时间分数（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_semi\_tandem\_time = case\_when(  
 stand\_test\_semi\_tandem %in% c(993) ~ 0, # 因为存在这个变量为993但是原因没有4和6的情况，所以这种直接赋值为0了，按这个变量优先  
 stand\_test\_semi\_tandem %in% c(999, NA) ~ case\_when(  
 rowSums(select(., starts\_with("stand\_test\_semi\_tandem\_reason\_")) == 4, na.rm = TRUE) > 0 ~ 0,  
 rowSums(select(., starts\_with("stand\_test\_semi\_tandem\_reason\_")) == 6, na.rm = TRUE) > 0 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 stand\_test\_semi\_tandem == 1 ~ 1,  
 stand\_test\_semi\_tandem == 5 ~ 0,  
 TRUE ~ NA\_real\_  
 )  
 )  
  
  
# 统计双脚半前后站立时间分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_semi\_tandem\_time, useNA = "always")

0 1 <NA>   
 393 7719 1870

# 计算双脚前后一条线站立时间分数（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_tandem\_time = case\_when(  
 stand\_test\_tandem %in% c(993) ~ 0, # 因为存在这个变量为993但是原因没有4和6的情况，所以这种直接赋值为0了，按这个变量优先  
 stand\_test\_tandem %in% c(999, NA) ~ case\_when(  
 rowSums(select(., starts\_with("stand\_test\_tandem\_reason\_")) == 4, na.rm = TRUE) > 0 ~ 0,  
 rowSums(select(., starts\_with("stand\_test\_tandem\_reason\_")) == 6, na.rm = TRUE) > 0 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 stand\_test\_tandem == 1 ~ 1,  
 stand\_test\_tandem == 5 ~ if\_else(stand\_test\_tandem\_time >= 10, 1, 0),  
 TRUE ~ NA\_real\_  
 )  
 )  
# 统计双脚前后一条线站立分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_tandem\_time, useNA = "always")

0 1 <NA>   
 774 6913 2295

# 计算双脚并拢站立时间分数（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_feet\_together\_time = case\_when(  
 stand\_test\_feet\_together %in% c(993) ~ 0,  
 stand\_test\_feet\_together %in% c(999, NA) ~ case\_when(  
 rowSums(select(., starts\_with("stand\_test\_feet\_together\_reason\_")) == 4, na.rm = TRUE) > 0 ~ 0,  
 rowSums(select(., starts\_with("stand\_test\_feet\_together\_reason\_")) == 6, na.rm = TRUE) > 0 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 stand\_test\_feet\_together == 1 ~ 1,  
 stand\_test\_feet\_together == 5 ~ 0,  
 TRUE ~ NA\_real\_  
 )  
 )  
  
# 统计双脚并拢站立分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_feet\_together\_time, useNA = "always")

0 1 <NA>   
 112 154 9716

# 计算最终平衡得分（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_balance\_score = case\_when(  
 rowSums(select(., final\_semi\_tandem\_time, final\_tandem\_time, final\_feet\_together\_time) == 1, na.rm = TRUE) >= 2 ~ 1,  
 rowSums(is.na(select(., final\_semi\_tandem\_time, final\_tandem\_time, final\_feet\_together\_time))) >= 2 ~ NA\_real\_,  
 rowSums(is.na(select(., final\_semi\_tandem\_time, final\_tandem\_time, final\_feet\_together\_time))) == 1 &  
 rowSums(select(., final\_semi\_tandem\_time, final\_tandem\_time, final\_feet\_together\_time) == 1, na.rm = TRUE) == 1 &  
 rowSums(select(., final\_semi\_tandem\_time, final\_tandem\_time, final\_feet\_together\_time) == 0, na.rm = TRUE) == 1 ~ NA\_real\_,  
 TRUE ~ 0  
 )  
 )  
  
# 根据平衡是否受影响调整NA值  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_balance\_score = case\_when(  
 is.na(final\_balance\_score) & balance\_test\_affected == 1 ~ 0,  
 TRUE ~ final\_balance\_score  
 )  
 )  
  
# 计算坐起测试时间(无误)  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_sit\_stand\_time = case\_when(  
 sit\_stand\_test\_completed %in% c(993) ~ 0,  
 sit\_stand\_test\_completed %in% c(999, NA) ~ case\_when(  
 rowSums(select(., starts\_with("sit\_stand\_test\_reason\_")) == 4, na.rm = TRUE) > 0 ~ 0,  
 rowSums(select(., starts\_with("sit\_stand\_test\_reason\_")) == 6, na.rm = TRUE) > 0 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 sit\_stand\_test\_completed == 1 ~ case\_when(  
 sit\_stand\_test\_arm\_use == 1 ~ 0,  
 sit\_stand\_test\_arm\_use == 5 ~ case\_when( # 这个地方数据和代码本否是5，但questionare是1，弄了半天  
 sit\_stand\_test\_time <= 12 ~ 1,  
 sit\_stand\_test\_time > 12 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 TRUE ~ NA\_real\_   
 ),  
 sit\_stand\_test\_completed == 5 ~ 0,  
 TRUE ~ NA\_real\_  
 )  
 )  
  
# 查看起坐测试时间分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_sit\_stand\_time, useNA = "always")

0 1 <NA>   
2589 5473 1920

# 计算运动维度总分（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_physical\_score = rowSums(select(., final\_walk\_time, final\_balance\_score, final\_sit\_stand\_time), na.rm = FALSE)  
 )  
  
# 查看运动维度总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_physical\_score, useNA = "always")

0 1 2 3 <NA>   
 272 1603 3763 1399 2945

# 导出运动维度所有变量和ID，去除其他所有变量，导出一个csv  
required\_columns <- c(  
 "ID", "final\_walk\_time", "final\_semi\_tandem\_time", "final\_tandem\_time", "final\_feet\_together\_time", "final\_balance\_score", "final\_sit\_stand\_time", "final\_physical\_score"  
)  
my\_data\_physical <- my\_data\_wash %>%  
 select(all\_of(required\_columns))  
  
write.csv(my\_data\_physical, "physical\_dimension\_data.csv", row.names = FALSE)

### 认知维度计算（需要考虑别人代问时也是NA）

# 计算最终回忆分数（无误，代填的也会有分，这种怎么处理，暂时不考虑这个条目）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_recall\_score = case\_when(  
 !is.na(recall\_refused) ~ NA\_real\_,  
 !is.na(recall\_none) ~ 0,  
 TRUE ~ rowSums(select(., starts\_with("recall\_word\_")) %>% mutate\_all(~ if\_else(!is.na(.), 1, 0)), na.rm = TRUE)  
 )  
 )  
  
# 统计最终回忆分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_recall\_score, useNA = "always")

0 1 2 3 4 5 6 7 8 9 10 <NA>   
1932 468 1156 1960 1934 1374 660 264 68 21 9 136

# 计算减法测试总分  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 total\_subtraction\_score = case\_when(  
 is.na(subtraction\_tool\_use) ~ NA\_real\_,# 辅助工具这项NA还挺多的，要考虑这个吗  
 subtraction\_tool\_use == 1 ~ 0,  
 rowSums(is.na(select(., starts\_with("subtraction\_test\_")))) == 5 ~ 0,  
 TRUE ~ rowSums(  
 cbind(  
 if\_else(subtraction\_test\_1 == 93, 1, 0),  
 if\_else(subtraction\_test\_2 == 86, 1, 0),  
 if\_else(subtraction\_test\_3 == 79, 1, 0),  
 if\_else(subtraction\_test\_4 == 72, 1, 0),  
 if\_else(subtraction\_test\_5 == 65, 1, 0)  
 ),  
 na.rm = TRUE  
 )  
 )  
 )  
  
# 统计减法测试总分不同取值的数量，包括NA  
table(my\_data\_wash$total\_subtraction\_score, useNA = "always")

0 1 2 3 4 5 <NA>   
 386 1520 878 467 439 2937 3355

# 计算日期感知总分(NA处理成0)  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_date\_perception\_score = rowSums(  
 cbind(  
 if\_else(!is.na(recall\_year), 1, 0),  
 if\_else(!is.na(recall\_month), 1, 0),  
 if\_else(!is.na(recall\_day), 1, 0)  
 ),  
 na.rm = TRUE  
 )  
 )  
  
# 统计日期感知总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_date\_perception\_score, useNA = "always")

0 1 2 3 <NA>   
2414 778 2162 4628 0

# 计算星期和季节感知总分（NA处理成NA，这样确实值还挺多）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_week\_season\_perception\_score = rowSums(  
 cbind(  
 if\_else(recall\_weekday == 1, 1, if\_else(recall\_weekday == 2, 0, NA\_real\_)),  
 if\_else(recall\_season == 1, 1, if\_else(recall\_season == 2, 0, NA\_real\_))  
 ),  
 na.rm = FALSE  
 )  
 )  
  
# 统计星期和季节感知总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_week\_season\_perception\_score, useNA = "always")

0 1 2 <NA>   
 474 1887 4581 3040

# 计算时间感知总分  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_time\_perception\_score = final\_date\_perception\_score + final\_week\_season\_perception\_score  
 )  
  
# 统计时间感知总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_time\_perception\_score, useNA = "always")

0 1 2 3 4 5 <NA>   
 175 340 651 860 1745 3171 3040

# 计算最终绘画得分（NA算0还是NA）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_drawing\_score = case\_when(  
 drawing\_test == 1 ~ 1,  
 drawing\_test == 2 ~ 0,  
 TRUE ~ NA\_real\_  
 )  
 )  
  
# 统计最终绘画得分不同取值的数量，包括NA  
table(my\_data\_wash$final\_drawing\_score, useNA = "always")

0 1 <NA>   
4143 4881 958

# 认知维度总分计算  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_cognitive\_score = case\_when(  
 final\_recall\_score + total\_subtraction\_score + final\_time\_perception\_score + final\_drawing\_score > 21 ~ 21,  
 TRUE ~ final\_recall\_score + total\_subtraction\_score + final\_time\_perception\_score + final\_drawing\_score  
 )  
 )  
  
# 统计认知维度总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_cognitive\_score, useNA = "always")

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15   
 4 9 12 32 63 83 143 179 305 377 506 561 582 529 635 610   
 16 17 18 19 20 21 <NA>   
 418 246 111 30 10 2 4535

# 计算认知维度层次  
quantiles <- quantile(my\_data\_wash$final\_cognitive\_score, probs = c(0.25, 0.5, 0.75), na.rm = TRUE)  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 cognitive\_level = case\_when(  
 is.na(final\_cognitive\_score) ~ NA\_real\_,  
 final\_cognitive\_score <= quantiles[1] ~ 0,  
 final\_cognitive\_score > quantiles[1] & final\_cognitive\_score <= quantiles[2] ~ 1,  
 final\_cognitive\_score > quantiles[2] & final\_cognitive\_score <= quantiles[3] ~ 2,  
 final\_cognitive\_score > quantiles[3] ~ 3  
 )  
 )  
  
# 查看认知维度层次不同取值的数量，包括NA  
table(my\_data\_wash$cognitive\_level, useNA = "always")

0 1 2 3 <NA>   
1713 1143 1774 817 4535

# 导出认知维度所有变量和ID，去除其他所有变量，导出一个csv  
required\_columns <- c(  
 "ID", "final\_recall\_score", "total\_subtraction\_score", "final\_time\_perception\_score", "final\_drawing\_score", "final\_cognitive\_score", "cognitive\_level"  
)  
my\_data\_cognitive <- my\_data\_wash %>%  
 select(all\_of(required\_columns))  
  
write.csv(my\_data\_cognitive, "cognitive\_dimension\_data.csv", row.names = FALSE)

### 心理维度计算

# 计算总睡眠时间（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 total\_sleep\_time = case\_when(  
 is.na(night\_sleep\_time) ~ NA\_real\_,  
 TRUE ~ night\_sleep\_time + coalesce(nap\_time / 60, 0)  
 )  
 )  
  
# 计算睡眠时间分数（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_sleep\_time\_score = case\_when(  
 is.na(total\_sleep\_time) ~ NA\_real\_,  
 total\_sleep\_time >= 5 & total\_sleep\_time <= 10.5 ~ 1,  
 TRUE ~ 0  
 )  
 )  
  
# 查看睡眠时间分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_sleep\_time\_score, useNA = "always")

0 1 <NA>   
1900 7184 898

# 计算不良睡眠频率分数（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_poor\_sleep\_score = case\_when(  
 poor\_sleep\_frequency %in% c(1, 2) ~ 1,  
 poor\_sleep\_frequency %in% c(3, 4) ~ 0,  
 TRUE ~ NA\_real\_  
 )  
 )  
  
# 查看不良睡眠频率分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_poor\_sleep\_score, useNA = "always")

0 1 <NA>   
3275 5904 803

# 计算抑郁总分（目前全部NA才算NA）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 total\_depression\_score = if\_else(  
 rowSums(is.na(select(., starts\_with("depression\_scale\_")))) == 9,  
 NA\_real\_,  
 rowSums(  
 select(., starts\_with("depression\_scale\_")) %>%  
 mutate\_all(~ case\_when(  
 . == 1 ~ 0,  
 . == 2 ~ 1,  
 . == 3 ~ 2,  
 . == 4 ~ 3,  
 TRUE ~ 0  
 )) %>%  
 mutate(  
 depression\_scale\_5 = case\_when(  
 depression\_scale\_5 == 1 ~ 3,  
 depression\_scale\_5 == 2 ~ 2,  
 depression\_scale\_5 == 3 ~ 1,  
 depression\_scale\_5 == 4 ~ 0,  
 TRUE ~ 0  
 ),  
 depression\_scale\_8 = case\_when(  
 depression\_scale\_8 == 1 ~ 3,  
 depression\_scale\_8 == 2 ~ 2,  
 depression\_scale\_8 == 3 ~ 1,  
 depression\_scale\_8 == 4 ~ 0,  
 TRUE ~ 0  
 )  
 ),  
 na.rm = TRUE  
 )  
 )  
 ) %>%  
 mutate(  
 final\_depression\_score = case\_when(  
 is.na(total\_depression\_score) ~ NA\_real\_,  
 total\_depression\_score <= 9 ~ 1,  
 TRUE ~ 0  
 )  
 )  
  
# 统计抑郁总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_depression\_score, useNA = "always")

0 1 <NA>   
2812 6415 755

# 心理维度总分计算  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_psychological\_score = rowSums(select(., final\_sleep\_time\_score, final\_poor\_sleep\_score, final\_depression\_score), na.rm = FALSE)  
 )  
  
# 统计心理维度总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_psychological\_score, useNA = "always")

0 1 2 3 <NA>   
 588 1692 2658 4060 984

# 导出心理维度所有变量和ID，去除其他所有变量，导出一个csv  
required\_columns <- c(  
 "ID", "total\_sleep\_time", "final\_sleep\_time\_score", "final\_poor\_sleep\_score", "final\_depression\_score", "final\_psychological\_score",  
 "depression\_scale\_1", "depression\_scale\_2", "depression\_scale\_3", "depression\_scale\_4", "depression\_scale\_5", "depression\_scale\_6",  
 "depression\_scale\_7", "depression\_scale\_8", "depression\_scale\_9"  
)  
my\_data\_psychological <- my\_data\_wash %>%  
 select(all\_of(required\_columns))  
  
write.csv(my\_data\_psychological, "psychological\_dimension\_data.csv", row.names = FALSE)

### 感官维度计算(无误)

# 计算视力总分（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_far\_vision = case\_when(  
 wearing\_glasses %in% c(1, 2) ~ 0,  
 wearing\_glasses %in% c(3, 4) ~ case\_when(  
 far\_vision %in% c(1, 2, 3, 4) ~ 1,  
 far\_vision == 5 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 TRUE ~ NA\_real\_  
 ),  
 final\_near\_vision = case\_when(  
 wearing\_glasses %in% c(1, 2) ~ 0,  
 wearing\_glasses %in% c(3, 4) ~ case\_when(  
 near\_vision %in% c(1, 2, 3, 4) ~ 1,  
 near\_vision == 5 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 TRUE ~ NA\_real\_  
 )  
 )  
  
# 统计视力总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_far\_vision, useNA = "always")

0 1 <NA>   
2996 6337 649

table(my\_data\_wash$final\_near\_vision, useNA = "always")

0 1 <NA>   
2595 6725 662

# 计算听力总分（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_hearing\_score = case\_when(  
 hearing\_aid == 1 ~ 0,  
 hearing\_aid == 2 ~ case\_when(  
 hearing\_status %in% c(1, 2, 3, 4) ~ 1,  
 hearing\_status == 5 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 TRUE ~ NA\_real\_  
 )  
 )  
# 统计听力总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_hearing\_score, useNA = "always")

0 1 <NA>   
1553 7650 779

# 计算感官维度总分  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_sensory\_score = rowSums(select(., final\_far\_vision, final\_near\_vision, final\_hearing\_score), na.rm = FALSE)  
 )  
# 统计感官维度总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_sensory\_score, useNA = "always")

0 1 2 3 <NA>   
 600 1592 1955 5017 818

# 导出感官维度所有变量和ID，去除其他所有变量，导出一个csv  
required\_columns <- c(  
 "ID", "final\_far\_vision", "final\_near\_vision", "final\_hearing\_score", "final\_sensory\_score"  
)  
my\_data\_sensory <- my\_data\_wash %>%  
 select(all\_of(required\_columns))  
  
write.csv(my\_data\_sensory, "sensory\_dimension\_data.csv", row.names = FALSE)

### 活力维度计算（无误，性别没有的也都没法衡量，为NA）

# 计算左手握力分数（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_left\_hand\_grip = case\_when(  
 left\_hand\_grip\_1 == 993 ~ 0,  
 left\_hand\_grip\_1 %in% c(999, NA) ~ case\_when(  
 rowSums(select(., starts\_with("grip\_test\_reason\_")) == 4, na.rm = TRUE) > 0 ~ 0,  
 rowSums(select(., starts\_with("grip\_test\_reason\_")) == 6, na.rm = TRUE) > 0 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 TRUE ~ pmax(left\_hand\_grip\_1, left\_hand\_grip\_2, na.rm = TRUE)  
 )  
 )  
  
# 计算右手握力分数（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_right\_hand\_grip = case\_when(  
 right\_hand\_grip\_1 == 993 ~ 0,  
 right\_hand\_grip\_1 %in% c(999, NA) ~ case\_when(  
 rowSums(select(., starts\_with("grip\_test\_reason\_")) == 4, na.rm = TRUE) > 0 ~ 0,  
 rowSums(select(., starts\_with("grip\_test\_reason\_")) == 6, na.rm = TRUE) > 0 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 TRUE ~ pmax(right\_hand\_grip\_1, right\_hand\_grip\_2, na.rm = TRUE)  
 )  
 )  
  
# 计算最终握力分数（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_grip\_score = case\_when(  
 gender == 1 ~ if\_else(pmax(final\_left\_hand\_grip, final\_right\_hand\_grip, na.rm = TRUE) >= 35, 1, 0),  
 gender == 2 ~ if\_else(pmax(final\_left\_hand\_grip, final\_right\_hand\_grip, na.rm = TRUE) >= 25, 1, 0),  
 TRUE ~ NA\_real\_ # 性别没法获取的时候这个也没法赋值  
 )  
 )  
  
# 统计握力分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_grip\_score, useNA = "always")

0 1 <NA>   
4682 3471 1829

# 计算呼吸功能总分（无误）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_breath\_test = case\_when(  
 breath\_test\_1 == 993 ~ 0,  
 breath\_test\_1 %in% c(999, NA) ~ case\_when(  
 rowSums(select(., starts\_with("breath\_test\_reason\_")) == 4, na.rm = TRUE) > 0 ~ 0,  
 rowSums(select(., starts\_with("breath\_test\_reason\_")) == 6, na.rm = TRUE) > 0 ~ 0,  
 TRUE ~ NA\_real\_  
 ),  
 TRUE ~ case\_when(  
 gender == 1 ~ if\_else(pmax(breath\_test\_1, breath\_test\_2, breath\_test\_3, na.rm = TRUE) >= 400, 1, 0),  
 gender == 2 ~ if\_else(pmax(breath\_test\_1, breath\_test\_2, breath\_test\_3, na.rm = TRUE) >= 290, 1, 0),  
 TRUE ~ NA\_real\_  
 )  
 )  
 )  
  
# 统计呼吸功能分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_breath\_test, useNA = "always")

0 1 <NA>   
5630 2480 1872

# 计算血红蛋白分数（算法无误，但不知道上界和下界算不算正常值）  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_hemoglobin\_score = case\_when(  
 is.na(hemoglobin) ~ NA\_real\_,  
 gender == 1 & hemoglobin \* 10 >= 120 & hemoglobin \* 10 <= 160 ~ 1,  
 gender == 1 & (hemoglobin \* 10 < 120 | hemoglobin \* 10 > 160) ~ 0,  
 gender == 2 & hemoglobin \* 10 >= 110 & hemoglobin \* 10 <= 150 ~ 1,  
 gender == 2 & (hemoglobin \* 10 < 110 | hemoglobin \* 10 > 150) ~ 0,  
 TRUE ~ NA\_real\_  
 )  
 )  
  
# 统计血红蛋白分数不同取值的数量，包括NA  
table(my\_data\_wash$final\_hemoglobin\_score, useNA = "always")

0 1 <NA>   
1150 5552 3280

# 计算活力维度总分  
my\_data\_wash <- my\_data\_wash %>%  
 mutate(  
 final\_vitality\_score = rowSums(select(., final\_grip\_score, final\_breath\_test, final\_hemoglobin\_score), na.rm = FALSE)  
 )  
  
# 统计活力维度总分不同取值的数量，包括NA  
table(my\_data\_wash$final\_vitality\_score, useNA = "always")

0 1 2 3 <NA>   
 519 2724 2275 1054 3410

# 导出活力维度所有变量和ID，去除其他所有变量，导出一个csv  
required\_columns <- c(  
 "ID", "final\_left\_hand\_grip", "final\_right\_hand\_grip", "final\_grip\_score", "final\_breath\_test", "final\_hemoglobin\_score", "final\_vitality\_score"  
)  
my\_data\_vitality <- my\_data\_wash %>%select(all\_of(required\_columns))  
write.csv(my\_data\_vitality, "vitality\_dimension\_data.csv", row.names = FALSE)

## 内在能力清洗

# 挑出只包含ID和五个内在能力维度总分的变量  
my\_data\_final <- my\_data\_wash %>%  
 select(ID, final\_physical\_score, cognitive\_level, final\_psychological\_score, final\_sensory\_score, final\_vitality\_score)  
  
# 计算覆盖百分之百内在能力变量的个体数量  
covered\_individuals <- my\_data\_final %>%  
 filter(!is.na(final\_physical\_score) & !is.na(cognitive\_level) & !is.na(final\_psychological\_score) & !is.na(final\_sensory\_score) & !is.na(final\_vitality\_score))  
  
# 输出覆盖百分之百内在能力变量的个体数量  
covered\_count <- nrow(covered\_individuals)  
print(paste("覆盖百分之百内在能力变量的个体数量:", covered\_count))

[1] "覆盖百分之百内在能力变量的个体数量: 3384"

# 导出成一个新的数据框  
write.csv(my\_data\_final, "final\_data.csv", row.names = FALSE)  
covered\_individuals

# A tibble: 3,384 × 6  
 ID final\_physical\_score cognitive\_level final\_psychological\_score  
 <chr> <dbl> <dbl> <dbl>  
 1 094004103001 2 0 2  
 2 094004112001 1 3 3  
 3 094004317002 1 3 2  
 4 094004303001 2 3 3  
 5 094004303002 2 3 3  
 6 094004307001 3 3 3  
 7 094004307002 2 3 2  
 8 194037213001 2 2 2  
 9 194037223002 1 3 3  
10 194037219002 3 1 3  
# ℹ 3,374 more rows  
# ℹ 2 more variables: final\_sensory\_score <dbl>, final\_vitality\_score <dbl>

## 自变量筛选

## 数据分割

## 模型构建

## 模型验证