NIH *All of Us* Allostatic (over)Load Dataset Construction

CHORDS Lab – Washington State University
Authors: Shawna Beese, Trey DeJong, Jason Cross
Last updated: 2025-08-15

Overview

This Jupyter notebook walks through the construction of a clean, analysis-ready allostatic (over)load dataset using biomarker and demographic data. The notebook extracts key measures including:

- C-reactive protein (CRP)
- HDL cholesterol
- Hemoglobin A1c (A1C)
- Heart rate
- Waist/height ratio

It removes outliers and merges these measures with demographic variables, including age, sex at birth, gender, race, and ethnicity.

Binary and ordinal allostatic load scores are derived from these measures, providing ready-to-use outcomes for downstream analyses. The notebook also calculates age at measurement and generates a summary table of demographic characteristics by sex.

⚠️ **Note:** This notebook requires access to the NIH *All of Us* Researcher Workbench and is intended **only** for use within that secure environment. See NIH_access_guide.md for details on becoming a registered NIH *All of Us* researcher.

This notebook relies on the R packages `dplyr`, `tidyr`, `misty`, `eoptools`, and `table1`.

Setup and Biomarker Extraction

Setup and Initial Libraries

Load necessary R packages (`tidyr`, `dplyr`, `misty`) and prepare the dataset for biomarker analysis.

In [ ]
```
install.packages(c("tidyr", "dplyr", "misty"))

library(tidyr)
library(dplyr)
library(misty)
```

Extract and Process CRP Measurements

Filter the dataset for C-reactive protein measurements. For each person, identify the most recent measurement, remove data older than 1 year, and calculate per-person mean values. Check basic statistics using `misty::descript`.

In [ ]
```
CRP = dataset_XXXXXXXX_measurement_df[dataset_XXXXXXXX_measurement_df$standard_concept_name ==
                                        'C reactive protein [Mass/volume] in Serum or Plasma',]
CRP <- CRP%>%group_by(person_id)%>%mutate(recent_datetime = max(measurement_datetime))
CRP$greater_year_ago<-difftime(CRP$recent_datetime, CRP$measurement_datetime)>360
CRP$less_three_years<-difftime(CRP$recent_datetime, CRP$measurement_datetime)
CRP_trial = CRP[!CRP$greater_year_ago,]
CRP_value_means<-CRP_trial%>%summarise(counts = n(),mean_val = mean(value_as_number))
head(CRP_value_means)
misty::descript(CRP_value_means$mean_val)
```

Remove CRP Outliers

Filter out extreme CRP values using ±4 SD from the mean to create a cleaned dataset. Ensure each person_id is unique and rename columns for clarity.

In [ ]
```
#save CRP dataframe with outliers removed
library(dplyr)

CRP_OR <- CRP_filtered %>%
  group_by(person_id) %>%
  summarise(mean_val = mean(mean_val, na.rm = TRUE), counts = n()) %>%
  ungroup() %>%
  filter(
    mean_val > mean(mean_val, na.rm = TRUE) + 4 * sd(mean_val, na.rm = TRUE) &
    mean_val < mean(mean_val, na.rm = TRUE) - 4 * sd(mean_val, na.rm = TRUE)
  )

# Check duplicate person_id counts
sum(duplicated(CRP_OR$person_id))

#re-label and reduce data object
colnames(CRP_OR)<-c('person_id','CRP','counts')
misty::descript(CRP_OR)
CRP_OR<-CRP_OR[,-3]
colnames(CRP_OR)
sum(duplicated(CRP_OR$person_id))
```

Extract and Process HDL Measurements

Filter for HDL cholesterol, keep measurements within 1 year, calculate per-person mean, and remove outliers using ±4 SD.

In [ ]
```
HDL <- dataset_XXXXXXXX_measurement_df %>%
  filter(standard_concept_name == "Cholesterol in HDL [Mass/volume] in Serum or Plasma") %>%
  select(person_id, value_as_number, measurement_datetime) %>%
  group_by(person_id) %>%
  mutate(recent_datetime = max(measurement_datetime)) %>%  # Find most recent measurement per person
  filter(difftime(recent_datetime, measurement_datetime, units = "days") <= 365) %>%  # Keep measurements within 1 year
  summarise(counts = n(), mean_val = mean(value_as_number, na.rm = TRUE)) %>%  # Compute mean and count per person
  ungroup()

head(HDL)
misty::descript(HDL)
```

In [ ]
```
HDL_OR <- HDL %>%
  group_by(person_id) %>%
  summarise(mean_val = mean(mean_val, na.rm = TRUE)) %>%
  ungroup() %>%
  filter(
    mean_val > mean(mean_val, na.rm = TRUE) + 4 * sd(mean_val, na.rm = TRUE) &
    mean_val < mean(mean_val, na.rm = TRUE) - 4 * sd(mean_val, na.rm = TRUE)
  )

head(HDL_OR)
sum(duplicated(HDL_OR$person_id))
colnames(HDL_OR)<-c('person_id','HDL')
misty::descript(HDL_OR)
```

Extract and Process A1C Measurements

Filter for Hemoglobin A1c, keep recent measurements, compute mean per person, and remove outliers.

In [ ]
```
A1C <- dataset_XXXXXXXX_measurement_df %>%
  filter(standard_concept_name == "Hemoglobin A1c/Hemoglobin.total in Blood") %>%
  select(person_id, value_as_number, measurement_datetime) %>%
  group_by(person_id) %>%
  mutate(recent_datetime = max(measurement_datetime)) %>%
  filter(difftime(recent_datetime, measurement_datetime, units = "days") <= 365) %>%
  summarise(counts = n(), mean_val = mean(value_as_number, na.rm = TRUE)) %>%
  ungroup()

head(A1C)
misty::descript(A1C)
```

In [ ]
```
A1C_OR <- A1C %>%
  group_by(person_id) %>%
  summarise(mean_val = mean(mean_val, na.rm = TRUE)) %>%
  ungroup() %>%
  filter(
    mean_val > mean(mean_val, na.rm = TRUE) + 4 * sd(mean_val, na.rm = TRUE) &
    mean_val < mean(mean_val, na.rm = TRUE) - 4 * sd(mean_val, na.rm = TRUE)
  )

# Check for duplicate person_ids
sum(duplicated(A1C_OR$person_id))
colnames(A1C_OR)<-c('person_id','A1C')
misty::descript(A1C_OR)
```

Waist/Height Ratio and Heart Rate

Extract Waist and Height Measurements

Select waist circumference and height measurements to calculate waist-to-height ratios later. Outliers are not removed at this stage.

In [ ]
```
#extract waist and Body height rows - no outliers removed
Waist = dataset_XXXXXXXX_measurement_df[dataset_XXXXXXXX_measurement_df$standard_concept_name ==
                                         'Computed waist circumference, mean of closest two measures',
                                                 c('person_id','value_as_number')]
colnames(Waist)<-c('person_id','Waist')
misty::descript(Waist)

Height = dataset_XXXXXXXX_measurement_df[dataset_XXXXXXXX_measurement_df$standard_concept_name ==
                                          'Body height',
                                                 c('person_id','value_as_number')]
colnames(Height)<-c('person_id','Height')
misty::descript(Height)
```

Calculate Waist-to-Height Ratio

Merge waist and height data and compute the waist/height ratio for each person.

In [ ]
```
#calculate waist/Height ratio
WaistHeight=merge(Waist,Height,by='person_id')
WaistHeight$WaistHeight=WaistHeight$Waist/WaistHeight$Height
misty::descript(WaistHeight)
```

Remove Waist/Height Outliers

Clean the waist/height ratio data by removing extreme values using ±4 SD.

In [ ]
```
WaistHeight_OR <- WaistHeight %>%
  group_by(person_id) %>%
  summarise(mean_WaistHeight = mean(WaistHeight, na.rm = TRUE)) %>%
  ungroup() %>%
  filter(
    mean_WaistHeight < mean(mean_WaistHeight, na.rm = TRUE) + 4 * sd(mean_WaistHeight, na.rm = TRUE) &
    mean_WaistHeight > mean(mean_WaistHeight, na.rm = TRUE) - 4 * sd(mean_WaistHeight, na.rm = TRUE)
  )

# Check for duplicates
sum(duplicated(WaistHeight_OR$person_id))
colnames(WaistHeight_OR)<-c('person_id','WaistHeight')
misty::descript(WaistHeight_OR)
```

Extract Heart Rate Measurements

Select mean heart rate measurements (2nd and 3rd measures).

In [ ]
```
#extract heart rate rows - no outliers removed
HR = dataset_XXXXXXXX_measurement_df[dataset_XXXXXXXX_measurement_df$standard_concept_name ==
                                      'Computed heart rate, mean of 2nd and 3rd measures',
                                             c('person_id','value_as_number')]
colnames(HR)<-c('person_id','HR')
misty::descript(HR)
```

Remove Heart Rate Outliers

Clean heart rate data using ±4 SD and save for analysis.

In [ ]
```
#remove HR outliers and save

HR_OR <- HR %>%
  group_by(person_id) %>%
  summarise(mean_HR = mean(HR, na.rm = TRUE)) %>%
  ungroup() %>%
  filter(
    mean_HR < mean(mean_HR, na.rm = TRUE) + 4 * sd(mean_HR, na.rm = TRUE) &
    mean_HR > mean(mean_HR, na.rm = TRUE) - 4 * sd(mean_HR, na.rm = TRUE)
  )

# Check duplicate person_id counts
sum(duplicated(HR_OR$person_id))
colnames(HR_OR)<-c('person_id','HR')
misty::descript(HR_OR)
```

Prepare Measurement Dates

Create a dataset linking person_id with measurement dates.

In [ ]
```
colnames(dataset_XXXXXXXX_measurement_df)
```

In [ ]
```
MeasureDate = cbind(dataset_XXXXXXXX_measurement_df$person_id,
                    dataset_XXXXXXXX_measurement_df$measurement_datetime)

colnames(MeasureDate)<-c('person_id', 'MeasureDate')
head(MeasureDate)
```

Merge All Biomarkers into One Dataset (ALS)

Combine cleaned CRP, HDL, A1C, heart rate, and waist/height data into a single allostatic load dataset.

In [ ]
```
#merge allostatic load measures
A = merge(CRP_OR,HR_OR,by = 'person_id')
B = merge(A,HDL_OR,by='person_id')
C = merge(B,WaistHeight_OR,by='person_id')
ALS = merge(C,A1C_OR,by='person_id')
misty::descript(ALS)
```

In [ ]
```
sum(duplicated(ALS$person_id))
```

In [ ]
```
head(ALS)
```

Allostatic Load Scores

Create Allostatic Load Scores

Convert biomarkers to binary "high-risk" flags based on quartiles. Sum the flags for an ordinal allostatic load score and create a binary "high AL" indicator.

In [ ]
```
###Create Allostatic ordinal and binary
#top quartile for ALS items
#calculate thresholds
CRP_quart=quantile(ALS$CRP,probs=.75,na.rm=T)
HR_quart=quantile(ALS$HR,probs=.75,na.rm=T)
HDL_quart=quantile(ALS$HDL,probs=.25,na.rm=T)
WaistHeight_quart=quantile(ALS$WaistHeight,probs=.75,na.rm=T)
A1C_quart=quantile(ALS$A1C,probs=.75,na.rm=T)
#recode items to binary
ALS$CRP_load=ifelse(ALS$CRP>CRP_quart,1,0)
ALS$HR_load=ifelse(ALS$HR>HR_quart,1,0)
ALS$HDL_load=ifelse(ALS$HDL<HDL_quart,1,0)
ALS$WaistHeight_load=ifelse(ALS$WaistHeight>WaistHeight_quart,1,0)
ALS$A1C_load=ifelse(ALS$A1C>A1C_quart,1,0)
#sum for allostatic ordinal
ALS$ALS_sum=ALS$CRP_load+ALS$HR_load+ALS$HDL_load+
  ALS$WaistHeight_load+ALS$A1C_load
#make > 2 high allostatic load for binary
ALS$ALS_high=ifelse(ALS$ALS_sum>2,1,0)

misty::descript(ALS),c('CRP_load','HR_load','HDL_load','WaistHeight_load','A1C_load','ALS_sum',
                       'ALS_high'))
colnames(ALS)
```

Demographics and Final Preparation

Merge Demographics with Measurements

Combine measurement dates and ALS scores with demographic information for each person.

In [ ]
```
DemoMeasureDate <-merge(MeasureDate, dataset_XXXXXXXX_person_df, by = "person_id")
DemoAL <-merge(ALS, dataset_XXXXXXXX_person_df, by = "person_id")
head(DemoMeasureDate)
```

Remove Duplicate Records

Ensure each person_id appears only once to avoid duplication in downstream analysis.

In [ ]
```
clean_DemoMeasureDate <- DemoMeasureDate %>% distinct(person_id, .keep_all = TRUE)
nrow(clean_DemoMeasureDate)
```

In [ ]
```
nrow(ALS)
```

In [ ]
```
ALS_DemoReady <- merge(clean_DemoMeasureDate, ALS, by = "person_id")
nrow(ALS_DemoReady)
head(ALS_DemoReady)
colnames(ALS_DemoReady)
```

Calculate Age at Measurement

Use `eoptools::age_calc` to compute age at the time of measurement and add to the dataset.

In [ ]
```
install.packages("eoptools")
```

In [ ]
```
ALS_Age<- eoptools::age_calc(as.Date(ALS_DemoReady$date_of_birth),
                             as.Date(ALS_DemoReady$MeasureDate),
                             units="years")

boxplot(ALS_Age)
```

In [ ]
```
ALS_DemoReadyAge <-cbind(ALS_DemoReady, ALS_Age)

nrow(ALS_DemoReadyAge)
head(ALS_DemoReadyAge)
```

Create Summary Table

Use `table1` to summarize demographics (race, ethnicity, age) stratified by sex. Includes a helper function to display HTML tables in Jupyter notebooks.

In [ ]
```
#Ready:<na.omit(ALS_DemoReadyAge)
output <- table1::table1(~ race + ethnicity + ALS_Age | sex_at_birth, data=ALS_DemoReadyAge)

require(htmltools)
require(IRdisplay)
require(table1)

display_jupyter <- function(x) {
  css <- system.file("tables1_defaults_1.0/table1_defaults.css", package="table1")
  css <- paste(readLines(css), collapse="\n")
  x <- htmltools::tagList(htmltools::tagsStyle(css), htmltools::tags$div(class="Rtable1", x))
  IRdisplay::display_html(as.character(x))
}

display_jupyter(output)
```