HMS 520 Final Project

Lauryn Stafford & Jessica Bishai

Overview: Three IHME-specific functions

All packaged into bestTeam.true

01	**MAD Outliering**	Outliers data using the Median Absolute Deviation
02	**Weighted Means**	Produces weighted means of IHME estimates by specified parameters
03	**Bundle Shape Converter**	Converts bundle data in DisMod shape to ST-GPR shape and vice versa

MAD Outliering

Produces a dataframe with updated outliers, identified through the Median Absolute Deviation method

Motivation

- A MAD outliering script was floating around among YLDs team members and causes
- Making a generalizable MAD outliering function can improve workflow and code sharing

Process

- Runs checks to make sure input data is valid
- Pulls in and merges specified age weights
- Calculates age-standardized mean for each data series -> outliers data if mean is higher than/lower than the specified number of MADs from median

Outputs

- The inputted dataframe, with an updated “is_outlier” column and MAD outliering specified in corresponding “note_modeler”

Calculating MAD outliers

```
## calculate median absolute deviation
dt_inp[as_mean == 0, as_mean := NA] ## don't count zeros in median calculations
dt_inp[,mad:=mad(as_mean,na.rm = T),by=c("sex")]
dt_inp[,median:=median(as_mean,na.rm = T),by=c("sex")]

dt_inp[as_mean>((outlier_val*mad)+median), is_outlier := 1]
dt_inp[as_mean>((outlier_val*mad)+median), note_modeler := paste0(note_modeler, " | outliered because a
dt_inp[as_mean<(median-(outlier_val*mad)), is_outlier := 1]
dt_inp[as_mean<(median-(outlier_val*mad)), note_modeler := paste0(note_modeler, " | outliered because l
```

Weighted Means

Produces weighted means of IHME estimates by specified parameters

Motivation

- IHME data are not always grouped by the same parameters
- Standardizing the data can enhance comparability

Process

- Runs checks to make sure input data is valid
- Pulls in and merges corresponding population data
- Calculates and returns weighted means

Outputs

- A standardized dataframe with corresponding location metadata, if applicable

How are the weighted means calculated?

```
weighted_df <- df_pop %>%
  # group by the desired variables
  group_by(!!!syms(weight_by)) %>%
  # calculate the share of total population in each group
  mutate(pop_share = population/sum(population)) %>%
  # calculate the weighted_mean share for each group
  mutate(weighted_mean_share = pop_share * mean) %>%
  # sum the weighted mean for each group
  summarise(weighted_mean = sum(weighted_mean_share))
```

Bundle Converter

Converts bundle data in DisMod shape to ST-GPR shape and vice versa

Motivation

- Requested by modelers on the NCH team to automate a frequently manual process

Process

- NOTE: two separate functions
- Takes in an ST-GPR or DisMod bundle
- Four actions for conversion
  - Renaming columns
  - Creating null columns
  - Prompts user to create or modify columns
  - Converting ‘type’ columns to ‘id’ columns and vice versa

Outputs

- A bundle that contains the necessary columns for the output bundle-shape

Converting id to type [1]

- ST-GPR shape bundles have type IDs, DisMod shape bundles have types

sampling_type_id	sampling_type
1	Cluster
2	Multistage
3	Nonprobability
4	Probability
5	Simple random

Converting id to type [2]

```r
sampling_type_converter <- function(val) {
  # https://hub.ihme.washington.edu/pages/viewpage.action?pageId=18575829
  sampling_type <- c("Not set", "Cluster", "Multistage", "Nonprobability",
                     "Probability", "Simple random")
  sampling_map <- data.frame(id = c(-1, 1:5), type = as.character(sampling_type))
  if (val %in% sampling_map$id) {
    return(sampling_map$type[sampling_map$id == val])
  }
  else {
    return("Not set")
  }
}
```

Converting id to type [3]

```
if (!(paste0(var_name, "_type") %in% columns)) {
  message(paste0("creating column '", var_name, "_type' from '",
                 var_name, "_type_id'"))
  # create a vector to store the mapped variable types
  store <- sapply(input_stgpr_bundle[,(paste0(var_name, "_type_id"))],
                  get(paste0(var_name, "_type_converter")))
  # put the newly created column in the data
  input_stgpr_bundle <- input_stgpr_bundle %>%
    mutate(!!paste0(var_name, "_type") := store)
}
else {
  # neither variable type or variable type_id exist inthe data.
  stop(paste0("input bundle must either have column '", var_name,
              "_type' or '", var_name, "_type_id'"))
```

bestTeam.true Package

- Includes documentation for function inputs, outputs, and examples
- Includes testing scripts
- Hoping to publish internally to IHME once the functions are finalized

Thank you!

Find the code here:
https://github.com/laurynks/HMS520_Final_Project_bestteam-true

Feel free to contact Lauryn Stafford (laurynks@uw.edu) or Jessica Bishai (jbishai@uw.edu) with any questions/suggestions!

Appendix 1: Weighted Means output

```
weighted_mean(test_df, weight_by = c("location_id", "age_group_id"))
```

location_id	age_group_id	weighted_mean	location_set_version_id	location_set_id	parent_id	path_to_top_
1	7	0.05245681	793	22	1	1
1	8	0.09080311	793	22	1	1
1	9	0.13913927	793	22	1	1
1	10	0.16849763	793	22	1	1
1	11	0.17110569	793	22	1	1
1	12	0.13607296	793	22	1	1
1	13	0.10785376	793	22	1	1
1	14	0.09847023	793	22	1	1
1	15	0.10391483	793	22	1	1
4	7	0.04146013	793	22	1	1,4
4	8	0.06906947	793	22	1	1,4
4	9	0.10977485	793	22	1	1,4
4	10	0.14088826	793	22	1	1,4
4	11	0.15154201	793	22	1	1,4

Appendix 2: Weighted Means calculation example

Loc	Sex	Value	Pop	pop_share	weighted_mean_share	weighted_mean
				= pop / sum(pop)	= value * pop_share	= sum(weighted_mean_share)
Atlantis	m	0.3	100	1/3	0.1	0.5
Atlantis	f	0.6	200	2/3	0.4	
Arendelle	m	0.25	30	3/5	0.15	0.45
Arendelle	f	0.75	20	2/5	00.3	
Agrabah	m	0.1	15	1/2	0.05	0.05
Agrabah	f	0	15	1/2	0	