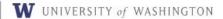
HMS 520 Final Project

Lauryn Stafford & Jessica Bishai

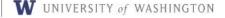




Overview: Three IHME-specific functions

Outliers data using the Median **MAD Outliering Absolute Deviation** All packaged Produces weighted means of IHME **Weighted Means** ()into estimates by specified parameters bestTeam.true **Bundle Shape** Converts bundle data in DisMod shape Converter to ST-GPR shape and vice versa





MAD Outliering

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

Motivation **Process** Outputs A MAD outliering script Runs checks to make sure input The inputted was floating around data is valid dataframe, with an Pulls in and merges specified age updated "is_outlier" among YLDs team members and causes column and MAD weights Making a generalizable Calculates age-standardized outliering specified mean for each data series -> in corresponding MAD outliering function can improve outliers data if mean is higher "note modeler" workflow and code than/lower than the specified number of MADs from median sharing





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 {
    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 ]</pre>
```

Weighted Means

Produces weighted means of IHME estimates by specified parameters

Motivation	Process	Outputs
 IHME data are not always grouped by the same parameters Standardizing the data can enhance comparability 	 Runs checks to make sure input data is valid Pulls in and merges corresponding population data Calculates and returns weighted means 	 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

Process Motivation Outputs A bundle that Requested by NOTE: two separate functions modelers on the NCH Takes in an ST-GPR or DisMod bundle contains the Four actions for conversion team to automate a necessary columns frequently manual Renaming columns for the output Creating null columns bundle-shape process Prompts user to create or modify columns Converting 'type' columns to 'id' columns and vice versa

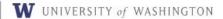




Converting id to type [1]

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]

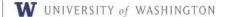
```
sampling type converter <- function(val) {</pre>
  # https://hub.ihme.washington.edu/pages/viewpage.action?pageId=18575829
  sampling_type <- c("Not set", "Cluster", "Multistage", "Nonprobability",
                     "Probability", "Simple random")
  sampling_map <- data.frame(id = c(-1, 1:5), type = as.character(sampling_type))</pre>
  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 (<u>laurynks@uw.edu</u>) or Jessica Bishai (<u>jbishai@uw.edu</u>) 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
S-11	VE.	A 11 110-110		82326	ture.	200





Appendix 2: Weighted Means calculation example

Loc	Sex	Value	Pop	pop_share	weighted_mean_share	weighted_mean	
200	Too Sox Value		ГОР	= 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		



