Algorithms for calculating MF from hierarchical data

Center for Veterinary Biologics - Statistics Section

August 2019

Contents

INTRODUCTION

This document is intended to provide additional details regarding the algorithms used for calculating mitigated fraction values for hierarchical data as implemented in package `MF` via the functions `MFClusHier`, `MFnest`, `MFClusBootHier`, and `MFnestBoot`. A passing familiarity is helpful in understanding the examples and underlying data input in this guide. A quick start may be found at Quick Start when calculating MF from Hierarchical Data.

Data assumptions. Nested Hierarchical structure.

Examples in this document expect data to be structured in a *nested hierarchical tree*, as shown in Table 1. If there are only two variables (columns) or the experimental design is not nested, then the methods described here may not apply. A nested design assumes that the factor of one variable co-occur with another variable. For example, Pen D exists only within

Room Z. If the experimental design of the data set is crossed (all factors of a variable co-occur with all factors of another variable), it is not appropriate to use this guide as written.

DETERMINING MITIGATED FRACTION

Rank tables.

Deterimining the rank table is the first step of the algorithm. For each *cluster*, the following values are determined:

- `con_n` & `vac_n`: Counts of observations for each treatment (control or vaccinate) for a particular instance of a unique factor level of a variable.
- `n1n2`: Product of counts, `con_n` * `vac_n`.
- `w`: Wilcoxon statistic.
- `u`: Mann-Whitney statistic.
- `con_medResp` & `vac_medResp`: Median observation for each treatment (control or vaccinate) in a particular instance of a unique factor level of a variable.

```
## # A tibble: 12 x 10
##    room  pen   litter con_medResp con_n     w vac_medResp vac_n  n1n2     u
##    <chr> <chr> <chr>        <dbl> <dbl> <dbl>       <dbl> <dbl> <dbl> <dbl>
##  1 Room~ Pen A Litte~        8.24     2     7        5.12     2     4     4
##  2 Room~ Pen A Litte~        4.92     2     5        3.81     2     4     2
##  3 Room~ Pen B Litte~        8.10     2     7        5.23     2     4     4
##  4 Room~ Pen B Litte~        8.12     2     7        5.60     2     4     4
##  5 Room~ Pen C Litte~        8.09     2     7        5.26     2     4     4
##  6 Room~ Pen C Litte~        6.77     2     7        4.50     2     4     4
##  7 Room~ Pen D Litte~        5.57     2     7        4.26     2     4     4
##  8 Room~ Pen D Litte~        7.44     2     6        6.33     2     4     3
##  9 Room~ Pen E Litte~        7.98     2     7        4.58     2     4     4
## 10 Room~ Pen E Litte~        6.78     2     7        4.86     2     4     4
## 11 Room~ Pen F Litte~        6.82     2     7        5.36     2     4     4
## 12 Room~ Pen F Litte~        6.77     1     3        5.14     2     2     2
```

A *cluster* refers to the full grouping designation of each observation. For example, in Table 1 the cluster of the first observation is Room W/Pen A/Litter 11, and there are four observations within that cluster.

The values in the rank table is the same regardless of the variable of interest. That is, whether a user is interested in the value of mitigated fraction for variable “pen” or “litter” does not affect the content of rows or columns, assuming the same input data.

Using package MF.

Rank tables can be output with the following usage in package `MF`:

- `MFClusHier(...)$MFh`: Rank table field from `MFClusHier()` output, calculated from input data.
- `MFh(...)`: Calculate just the rank table from input data.
- `MFClusBootHier(...)$MFhBoot`: List which includes rank table using bootstrapped input data as `...$bootmfh` and calculated directly from inputdata as `...$mfh`.
- `MFhBoot(...)`: List which includes rank table using bootstrapped input data as `...$bootmfh` and calculated directly from inputdata as `...$mfh` without the additional `...$MFnestBoot` output.

Summarizing the rank table.

With a detailed rank table, it is possible to calculate the mitigated fraction for a variety of variable choices. This is the second step of the algorithm. For each variable of interest the following summations are calculated for each level of that variable.

- `N1N2`: Sum of the `n1n2` values from the rank table for the factor level designated in the row.
- `U`: Sum of the `u` values from the rank table for the factor level designated in the row.
- `con_N` & `vac_N`: Sum of the counts from the rank table matching the particular factor level designated in the row.
- `con_medResp` & `vac_medResp`: Median of responses for each comparison group matching the particular factor level of the row.
- `MF`: $2 * U/N1N2 - 1$.

The following example looks at the mitigated fraction of factors in variable "pen". We can see that the values for "N1N2" have been summed from the previous rank table for all cases of "Pen A", "Pen B", etc.

```
## # A tibble: 6 x 9
##   variable level    MF  N1N2     U con_N vac_N con_medResp vac_medResp
##   <fct>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl>       <dbl>       <dbl>
## 1 pen      Pen A  0.5      8     6     4     4        6.80        4.24
## 2 pen      Pen B  1        8     8     4     4        8.12        5.60
## 3 pen      Pen C  1        8     8     4     4        7.69        4.85
## 4 pen      Pen D  0.75     8     7     4     4        6.1         4.98
## 5 pen      Pen E  1        8     8     4     4        6.86        4.86
## 6 pen      Pen F  1        6     6     3     4        6.77        5.14
```

It is possible to perform this summary without regard for variables to get an overall mitigated fraction value.

```
## # A tibble: 1 x 9
##   variable level    MF  N1N2     U con_N vac_N con_medResp vac_medResp
```

```
##   <fct>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl>       <dbl>       <dbl>
## 1 All      All   0.870    46    43    23    24        7.21        4.91
```

Using package MF.

- `MFClusHier(...)`: calculates rank table and MF table as a single step.
- `MFnest(...)`: calculates MF table from rank table.
- `MFClusBootHier(...)`: calculates rank table and MF table as a single step, with bootstrapping.
- `MFnestBoot(...)`: calculates MF table including from bootstrapped rank table.

BOOTSTRAPPING

Bootstrapping can occur in one of two points of consideration: at *clusters* or at *units*.

Resampling clusters.

In Table 1 the cluster of the first observation is Room W/Pen A/Litter 11, and there are four observations within that cluster. Bootstrapping clusters samples from the unique clusters with replacement, so every instance will have the same number of clusters as the original data, but they may not be unique. The example below shows one possiblility of a bootstrap instance from the example data.

```
## # A tibble: 12 x 4
##    bootID room   pen   litter
##     <int> <chr>  <chr> <chr>
##  1      1 Room W Pen A Litter 11
##  2      1 Room W Pen B Litter 13
##  3      1 Room W Pen B Litter 14
##  4      1 Room W Pen C Litter 15
##  5      1 Room W Pen C Litter 15
##  6      1 Room W Pen C Litter 16
##  7      1 Room Z Pen E Litter 20
##  8      1 Room Z Pen E Litter 20
##  9      1 Room Z Pen E Litter 20
## 10      1 Room Z Pen F Litter 21
## 11      1 Room Z Pen F Litter 22
## 12      1 Room Z Pen F Litter 22
```

During bootstrapping of clusters, the observations within that cluster do not change composition. That is, Room W/Pen A/Litter 11 continues to have the same two vaccinates (5.63, 4.62) and the same two controls (9.20, 7.28).

Resampling units.

Bootstrapping the unit involves sampling with replacement the observed values within a particular comparison group for that cluster. The mix of control and vaccinate groups remains constant for a unique cluster and determines the number of sampling events. For example, the cluster Room W/Pen A/Litter 11 has 2 vaccinates and 2 controls. Bootstrapping the unit involves sampling with replacement twice from the observed response of samples in the selection Room W/Pen A/Litter 11/vac and twice from the observed response of samples in the selection Room W/Pen A/Litter 11/con for each instance of the cluster Room W/Pen A/Litter 11. The number of times that a unique cluster exists in the bootstrapped data is determined by sampling of the clusters (see above discussion).

After bootstrapping at the unit level, a particular instance of the cluster Room W/Pen A/Litter 11 will continue to contain two vaccinates but the values may both be 5.63 while the two controls may have the values (7.28, 9.20).

APPENDIX

Code used in this example

```
a <- data.frame(
 room = paste('Room', rep(c('W', 'Z'), each = 24)),
 pen = paste('Pen', rep(LETTERS[1:6], each = 8)),
 litter = paste('Litter', rep(11:22, each = 4)),
 tx = rep(rep(c('vac', 'con'), each = 2), 12),
 stringsAsFactors = FALSE
 )
set.seed(76153)
a$lung[a$tx == 'vac'] <- rnorm(24, 5, 1.3)
a$lung[a$tx == 'con'] <- rnorm(24, 7, 1.3)
a <- a[-48,]

# DETERMINING MITIGATED FRACTION
thismfh <- MFh(formula = lung ~ tx + room/pen/litter, data = a)
MFnest(thismfh, which.factor = "pen")
MFnest(thismfh, which.factor = "All")

# BOOTSTRAPPING
MFClusBootHier(formula = lung ~ tx + room/pen/litter,
                                  data = a, boot.unit = FALSE,
                                  boot.cluster = TRUE,  alpha = 0.1,
                                  which.factor = c('room', 'litter', 'All'))
mfboot_multiple$MFhBoot$bootmfh %>%
```

```
  filter(bootID == 1) %>%
  select(c('bootID', 'room', 'pen', 'litter'))
```

Session details for this manual.

```
sessionInfo()
```

```
## R version 3.5.3 (2019-03-11)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 14393)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] MF_4.3.6         forcats_0.4.0    stringr_1.4.0    dplyr_0.8.3
##  [5] purrr_0.3.2      readr_1.3.1      tidyr_0.8.3      tibble_2.1.3
##  [9] ggplot2_3.2.0    tidyverse_1.2.1  kableExtra_1.1.0 knitr_1.23
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.5  xfun_0.8          haven_2.1.1
##  [4] lattice_0.20-38   colorspace_1.4-1  vctrs_0.2.0
##  [7] generics_0.0.2    htmltools_0.3.6   viridisLite_0.3.0
## [10] yaml_2.2.0        utf8_1.1.4        rlang_0.4.0
## [13] pillar_1.4.2      withr_2.1.2       glue_1.3.1
## [16] modelr_0.1.4      readxl_1.3.1      plyr_1.8.4
## [19] munsell_0.5.0     gtable_0.3.0      cellranger_1.1.0
## [22] rvest_0.3.4       evaluate_0.14     fansi_0.4.0
## [25] broom_0.5.2       Rcpp_1.0.1        scales_1.0.0
## [28] backports_1.1.4   webshot_0.5.1     jsonlite_1.6
## [31] hms_0.5.0         digest_0.6.20     stringi_1.4.3
## [34] grid_3.5.3        cli_1.1.0         tools_3.5.3
## [37] magrittr_1.5      lazyeval_0.2.2    crayon_1.3.4
## [40] pkgconfig_2.0.2   zeallot_0.1.0     xml2_1.2.0
```

```
## [43] lubridate_1.7.4   assertthat_0.2.1  rmarkdown_1.14
## [46] httr_1.4.0        rstudioapi_0.10   R6_2.4.0
## [49] nlme_3.1-137      compiler_3.5.3
```

Table 1: Nested hierarchical data structure.

room	pen	litter	tx	lung
Room W	Pen A	Litter 11	vac	5.63
				4.62
			con	9.20
				7.28
		Litter 12	vac	3.76
				3.86
			con	6.31
				3.52
	Pen B	Litter 13	vac	4.36
				6.10
			con	7.87
				8.34
		Litter 14	vac	5.39
				5.80
			con	8.28
				7.95
	Pen C	Litter 15	vac	4.86
				5.66
			con	7.66
				8.52
		Litter 16	vac	4.17
				4.84
			con	5.82
				7.72
Room Z	Pen D	Litter 17	vac	3.60
				4.92
			con	5.22
				5.93
		Litter 18	vac	7.62
				5.04
			con	8.62
				6.27
	Pen E	Litter 19	vac	3.79
				5.38
			con	9.45
				6.51
		Litter 20	vac	5.30
				4.42
			con	7.21
				6.35
	Pen F	Litter 21	vac	4.69
				6.04
			con	6.64
				7.00
		Litter 22	vac	4.90
				5.37
			con	6.77