Supplementary material to

Drugs - Real World Outcomes

Using real-world data to predict clinical and economic benefits of a future drug based on its target product profile

Christoph Gerlinger1,2*, Thomas Evers3, Jeremy Rassen4 and Richard Wyss5

* Correspondence: Christoph Gerlinger, Statistics and Data Insights, Bayer AG, Müllerstraße 178, 13353 Berlin, Germany. Email: christoph.gerlinger@bayer.com

1 Statistics and Data Insights, Bayer AG, Müllerstraße 178, 13353 Berlin, Germany.

2 Gynecology, Obstetrics and Reproductive Medicine, University Medical School of Saarland, Homburg/Saar, Germany.

3 Global Market Access, Bayer AG, Wuppertal, Germany.

4 Aetion, Inc., New York, NY, USA.

5 Division of Pharmacoepidemiology and Pharmacoeconomics, Harvard Medical School, Boston, MA, USA.
Documentation of R code

Table of Contents
DESCRIPTION OF INPUT: ................................................................. 3
sim_main.R ....................................................................................... 9
check_input.R ........................................................................... 29
Sampling.R .................................................................................. 40
cell_counts.R ............................................................................ 46
analysis.R ..................................................................................... 47
utils.R ......................................................................................... 49
DESCRIPTION OF INPUT:

1) Dataset
2) Outcomes:
   o Up to 3 binary outcomes can be specified
3) Categorical subgroups:
   o name of subgroups for categorical variables
   o no restrictions on number of subgroups
4) Continuous subgroups:
   o name of subgroups for continuous variables
   o no restrictions on number of subgroups
5) Subgroup combination:
   o Vector of names for binary subgroup variables that the user would like to combine through some logical operation. Currently, tool only supports binary variables for subgroup combinations.
   o Input for this variable only includes the names of the binary variables that the user would like to perform some logical operation on. It does not include the logical operators. The logical operators that determine how these variables will be combined is specified in the ‘subgroups_combination_operators’ input variables (see below).
6) Subgroups combination operators
   o Vector of logical operators specified as ‘AND’ or ‘OR’
   o These logical operators determine how the variables specified in ‘subgroup_combination’ (see above) are combined to form a new variable that is a logical combination of the variables specified in ‘subgroup_combination’.
   o The result is a single variable that is some logical combination of the variables specified in ‘subgroup_combination’.
   o Logical operators are performed in order from left to right. For example, if (var1, var2, var3) is specified as input for ‘subgroups_combination’, then the user could specify (AND, AND) as input for the ‘subgroups_combination_operators’ to create a new variable that is equal to 1 when var1=1 AND var2=1 AND var3=1. If the user specified (AND, OR) as input for ‘subgroups_combination_operators’, then the new variable would be equal to 1 when (var1=1 AND var2=1) or when var3=1.
7) Time variable for cost (time_var_cost):
   o Name of variable in dataset for follow-up time for the cost variable
   o The time variable should correspond with the time frame where cost contributes to the total cost (i.e., correspond with the cost outcome).
   o Variable is rounded up to the nearest day
   o This variable is used to calculate cost/day
8) Time variable(s) for outcome(s):
   o name of variable(s) in dataset for follow-up time for each outcome
   o Each time variable corresponding with a specific outcome represents the time frame where the individual contributes person time for that outcome.
   o Time variables are rounded up to the nearest day
These times are used to calculate rates for each outcome (e.g., rate differences and rate ratios)

**9) Name of Cost Variable:**
- Cost variable is divided by ‘time_var_cost’ to obtain cost/day
- Individuals with negative values for cost are deleted prior to performing analysis

**10) Treatment cost:**
- Value specified by user for hypothetical cost of Bayer drug (numeric scalar).
- Used to evaluate hypothetical cost comparisons

**11) Sample type:**
- Specified as either ‘marginal’ or ‘conditional’
- When ‘marginal’ is specified, then the user must specify input for either ‘target risk ratios’ or ‘cell proportions’ (see below).
- ‘marginal’ sampling creates a pseudo-treatment group so that the marginal risk ratios for each outcome between the original population and pseudo-treatment (Bayer) group are equal to the values specified for ‘target risk ratios’, or values according to the sampled ‘cell proportions’ if values for ‘cell proportions are specified instead. See Gerlinger et al. (2019 working manuscript) in references below for details on the sampling performed for ‘marginal’ sampling.
- ‘conditional’ uses inverse odds of sampling weights to create the pseudo-treatment group and is a more flexible sampling approach that creates a pseudo-treatment group so that the conditional odds ratios for each outcome between the original population and pseudo-treatment (Bayer) group are equal to the values specified for ‘conditional odds ratios’ (see description on ‘conditional odds ratios’ below for more detail). See Westreich et al. (2017) in references below for details on inverse odds of sampling weights
- IMPORTANT NOTE: Ideally, the user would want to create a pseudo-treatment group that satisfies the target product profile which is usually specified in terms of marginal effects. However, ‘marginal’ sampling as implemented in this tool is less flexible than ‘conditional’ sampling as it requires at least some individuals in all possible cell combinations. For example, if we have two outcomes, marginal sampling requires at least one individual in each of the cells:

| Cell | Outcome 1 | Outcome 2 |
|------|-----------|-----------|
| 1    | 0         | 0         |
| 2    | 1         | 0         |
| 3    | 0         | 1         |
| 4    | 1         | 1         |

When this is not satisfied, conditional sampling can be used to try to approximate the desired marginal effects. Of course, specified values for the ‘conditional odds ratios’ may not result in similar values for the ‘marginal risk ratios’. The user will have to play around with specified input values for ‘conditional odds ratios’ to try to get values for the marginal risk ratio that are close to the target product profile.
- Note that if conditional sampling is used, the conditional odds ratios between the original population and sampled pseudo-population (Bayer treatment group) will be equal to the specified input for ‘conditional odds ratios’ on average. Therefore, it may be beneficial to run several simulations and take the average as there is more variation in the values of the
resulting conditional odds ratios and marginal risk ratios between the original and sampled pseudo-population from sample to sample when conditional sampling is used versus marginal sampling.

12) Target risk ratios
- To obtain desired risk ratios for ‘marginal’ sampling, the user can specify the target risk ratio for each outcome.
- If values for the target risk ratios are specified, then no values for either ‘cell proportions’ or ‘conditional odds ratios’ can be entered.
- Number of values for target risk ratios must correspond with number of outcomes
- IMPORTANT NOTE: If there are two or more outcomes, then there are many ways to sample individuals into the pseudo-population (representing Bayer drug) to obtain the specified risk ratios (i.e., sampling proportions for each possible outcome subgroup combination, or cells, are not unique). Therefore, to obtain unique sampling proportions we need to add an additional constraint. There are many different constraints that we could consider. In the module, we adopt the suggestion of William A. Huber who suggested using odds ratios. In other words, for two outcomes we add the constraint that the odds ratio between outcome 1 and outcome 2 in the newly sampled pseudo-population is equal to the odds ratio between outcome 1 and outcome 2 in the original population. For three outcomes, we use the same method for 2 outcomes within strata of the third. See References for details.

13) Cell proportions
- Instead of specifying a target risk ratio for each outcome, the user can manually specify the size for each cell (in terms of proportion of total population).
- If the user wants to manually specify cell proportions, then no values for the target risk ratios can be specified.
- The number of cells are \(2^{\text{number of outcomes}}\).
- **Example for 1 outcome:**
  - for 1 outcome, there are only 2 cells that need to be specified:
    | Cell | Outcome 1 |
    |------|-----------|
    | 1    | 0         |
    | 2    | 1         |
  - The user would need to specify a proportion for cells 1 and 2. Each proportion determines the size of that cell in the new population (pseudo-population) and is equal to the proportion specified times the total size of the original population. For example, if 0.9 is specified for cell 1 and 0.1 for cell 2, then the size of cell 1 in the new population would be \(0.9\times N\) where \(N\) is the size of the original population, and the size of cell 2 would be \(0.1\times N\).
  - Proportions need to be specified in order. For example, if PC1 and PC2 represent the specified proportions for cells 1 and 2, respectively, the input would be:
    - PC1, PC2
- **Example for 2 outcomes:**
  - For 2 outcomes there are 4 cell proportions that need to be specified: The table below illustrates each of the possible cells for 2 outcomes.
The user specifies a vector of size 4, where each entry contains a number between 0 and 1 representing the proportion of the total population for that cell.

Proportions need to be specified in the order:
- PC1, PC2, PC3, PC4

**Example for 3 outcomes:**

For 3 outcomes there are 8 cell proportions. The table below illustrates each of the possible cells.

| Cell | Outcome 1 | Outcome 2 | Outcome 3 |
|------|-----------|-----------|-----------|
| 1    | 0         | 0         | 0         |
| 2    | 1         | 0         | 0         |
| 3    | 0         | 1         | 0         |
| 4    | 1         | 1         | 0         |
| 5    | 0         | 0         | 1         |
| 6    | 1         | 0         | 1         |
| 7    | 0         | 1         | 1         |
| 8    | 1         | 1         | 1         |

The user specifies a vector of size 8, where each entry contains a number between 0 and 1 representing the proportion of the total population for that cell. Proportions need to be specified in order.
- PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8

14) Conditional odds ratios (conditional\_OR):

- If marginal sampling fails, user can try ‘conditional’ sampling to approximate the desired marginal risk ratios. This type of sampling uses inverse odds of sampling weights as described in Westreich et al. (2017).

- When ‘conditional’ is specified for ‘sampling type’, the user must specify the desired effect for each outcome between the original population and pseudo-treatment (Bayer) group in terms of conditional odds ratios. The number of values specified as input for ‘conditional odds ratios’ must correspond to the number of outcomes specified.

- The values specified for ‘conditional odds ratios’ will not correspond to the desired effects in terms of marginal risk ratios. The user will have to try different values for conditional effects to get marginal effects that are suitable.

15) Percentage trimmed on cost variable:

- Number between 0 and 100 specifying what percent of the population to trim (trimming is done on the right tail of the distribution for cost)

- There are some outlying observations after calculating the cost outcome (cost per day). This parameter allows user to trim those extreme values prior to analyses.

16) Percentage of population for analysis:
1) Number between 0 and 100 specifying how much of the population to randomly sample for analysis.
2) Input variable allowing user to take random % of population for analysis
3) Speeds up computation if smaller sample is selected

17) Number of simulations

Example Input:

Outcomes: “PO1”, “PO2”, “PO3”
Time Variable Cost: "time_cost"
Time Variable Outcomes: “time1”, “time2”, “time3”
Subgroups (Categorical): "PC2", "PC3", "PC5"
Subgroups (Continuous): "PC1"
Subgroups (Combination): “PC2”, “PC3”
Subgroups Combination Operators: “AND”
Cost Variable: "PO3"
Sample Type: “marginal”
Target Risk Ratio(s): 0.8, 0.9, 1.2
Cell Proportions:
Conditional Odds Ratios:
Treatment Cost: 100
Percent Trimmed on Cost outcome (value between 0 and 1): 0.01
Percent Sampled for Analysis (value between 0 and 1): 1
Number of Simulations: 1

REFERENCES

https://stats.stackexchange.com/questions/241384/how-to-derive-2x2-cell-counts-from-contingency-table-margins-and-the-odds-ratio

Gerlinger C, Evers T, et al. Using real world evidence to estimate the costs of future drugs based on its target product profile. Working paper.
Westreich D, Edwards JK, Lesko CR, Stuart E, Cole SR. Transportability of trial results using inverse odds of sampling weights. *American Journal of Epidemiology*. 2017; 186(8):1010-1014.
sim_main.R

rm(list=ls())

## loading libraries
library(tidyverse)
library(dplyr)
library(ggplot2)
library(ggrepel)
library(jsonlite)
library(xtable)
library(base64enc)
library(epitools)
library(knitr)
library(purrr)

## loading helper functions
basedir = "//Cifs2/homedir$/Aetion_project/github_6_24_19/value-simulator-master/value-simulator-master/src/R/revised/Jeremy_email/sim_module"
source(file.path(basedir, "check_input.R"))
source(file.path(basedir, "sampling.R"))
source(file.path(basedir, "cell_counts.R"))
source(file.path(basedir, "analysis.R"))
source(file.path(basedir, "utils.R"))

# Simulation Module

sim_module<- function(in_dataset, in_Y, in_subgroups_categorical, in_subgroups_continuous, in_subgroups_combination, in_subgroups_combination_operators, in_time_var_cost, in_time_var_outcomes, in_cost_var, in_tmt_cost, in_sample_type, in_target_RR=NULL, in_conditional_OR=NULL, in_cell_prop=NULL, in_p_cost_trimmed=0, in_p_sample=1,
in_nsim, ...)
#
# \#
# checking that input is valid & setting up data
#
# #
# check_input(dataset=dataset,
#    Y=in_Y,
#    subgroups_categorical=in_subgroups_categorical,
#    subgroups_continuous=in_subgroups_continuous,
#    subgroups_combination=in_subgroups_combination,
#    subgroups_combination_operators=in_subgroups_combination_operators,
#    time_var_cost=in_time_var_cost,
#    time_var_outcomes=in_time_var_outcomes,
#    cost_var=in_cost_var,
#    tmt_cost=in_tmt_cost,
#    sample_type=in_sample_type,
#    target_RR=in_target_RR,
#    conditional_OR=in_conditional_OR,
#    cell_prop=in_cell_prop,
#    p_cost_trimmed=in_p_cost_trimmed,
#    p_sample=in_p_sample,
#    nsim=in_nsim)
#
# #
# renaming input variables to avoid overwrite of input variables
#
# Y <- in_Y
# subgroups_categorical <- in_subgroups_categorical
# subgroups_continuous <- in_subgroups_continuous
# subgroups_combination <- in_subgroups_combination
# subgroups_combination_operators <- in_subgroups_combination_operators
# time_var_cost <- in_time_var_cost
# time_var_outcomes <- in_time_var_outcomes
# cost_var <- in_cost_var
# tmt_cost <- in_tmt_cost
# sample_type <- in_sample_type
# target_RR <- in_target_RR
# conditional_OR <- in_conditional_OR
cell_prop <- in_cell_prop
p_cost_trimmed <- in_p_cost_trimmed
p_sample <- in_p_sample
nsim <- in_nsim

##########################################################################
### Setting up data, cleaning data, and generating new subgroups
variables and strata
###
##########################################################################

### setting up dataset of subgroup variables
subgroups_all <- NULL
subgroups_all <- c(subgroups_categorical, subgroups_continuous, subgroups_combination)
Xcovs <- select(dataset, subgroups_all, subgroups_combination)

### setting up dataset of follow-up time for cost (round up to nearest
day)
time_data_cost <- NULL
time_data_cost <- mutate(select(dataset, time_cost = time_var_cost),
time_cost = ceiling(as.numeric(time_cost)))

### setting up dataset of follow-up time for outcomes (round up to
nearest day)
time_data_outcomes <- dataset %>%
  select(time_var_outcomes) %>%
  mutate_if(is.character, as.numeric)
time_data_outcomes <- as.data.frame(apply(time_data_outcomes, 2,
  ceiling))
names_time_outcomes <- paste0('time_outcome',
1:ncol(time_data_outcomes))
names(time_data_outcomes) <- names_time_outcomes

### setting up dataset of outcome variables
outcome_data <- NULL
outcome_data <- dataset %>%
  select(Y) %>%
  mutate_if(is.integer, as.logical) %>%
  mutate_if(is.character, as.logical)
names_outcomes <- paste0('outcome', 1:dim(outcome_data)[2])
names(outcome_data) <- names_outcomes

### setting up dataset of cost variable (rounding up to nearest dollar)
total_cost <- NULL
total_cost <- mutate(select(dataset, total_cost = cost_var),
total_cost = ceiling(as.numeric(total_cost)))
## combining datasets

dt0 <- NULL
dt0 <- bind_cols(outcome_data, total_cost, time_data_cost, time_data_outcomes, Xcovs)

## removing observations from data with missing values

dt1 <- NULL
dt1 <- na.omit(dt0)

## taking random sample of data (specified by p_sample). If p_sample=1, then full dataset is used.

data_sample <- NULL
data_sample <- sample_frac(dt1, size=p_sample, replace=FALSE)

## dividing total cost by time to get cost/day (rounding up to nearest dollar)

data <- NULL
data <- data_sample %>%
  mutate(newID = row_number()) %>%
  mutate(outcome_cost = ceiling((total_cost / time_cost)))

## cleaning data (excluding individuals with negative cost)

data_clean <- NULL
data_clean <- data %>%
  filter(total_cost >= 0)

## cleaning data (excluding individuals with negative or 0 follow-up time for outcomes)

for (v in names(time_data_outcomes)) {
  v.f = sym(v)
data_clean <- data_clean %>%
    filter(!!v.f > 0)
}

## trimming population based on cost distribution

t1 <- NULL
t1 <- quantile(data_clean$outcome_cost, 1-p_cost_trimmed)
data1 <- NULL
data1 <- data_clean %>%
  filter(outcome_cost <= t1) %>%
  arrange(newID)

## creating new subgroup variables based on combinations of binary covs specified by user

helper_fun_comb <- function(x, y, operator){
  newvar_AND <- ifelse(x==1 & y==1, 1, 0)
  newvar_OR <- ifelse(x==1 | y==1, 1, 0)
  if(operator=='AND') return(newvar_AND)
  if(operator=='OR') return(newvar_OR)
}
temp1 <- temp2 <- temp3 <- name_temp <- NULL
if(length(subgroups_combination)>1){
  for(i in 1:(length(subgroups_combination)-1)){
    if(i==1){

temp1 <- as.numeric(data1[,subgroups_combination[i]])
name_temp <- paste0(subgroups_combination[i], " ",
subgroups_combination_operators[i], ", ", subgroups_combination[i+1])
}
if(i>1){
temp1 <- temp3
name_temp <- paste0(name_temp, " ",
subgroups_combination_operators[i], ", ", subgroups_combination[i+1])
temp2 <- as.numeric(data1[,subgroups_combination[i+1]])
temp3 <- helper_fun_comb(temp1, temp2,
toupper(subgroups_combination_operators[i]))
}
temp3 <- as.data.frame(temp3)
names(temp3) <- name_temp
data1 <- as.data.frame(cbind(data1, temp3))
}
subgroups_categorical <- unique(c(subgroups_categorical,
subgroups_combination, name_temp))

## creating strata for continuous variables
data2 = data1
strata_names1 <- strata_names2 <- NULL
if(length(subgroups_continuous)>0) {
  for(i in 1:length(subgroups_continuous)){
    var_name = subgroups_continuous[i]
data2 <- data2 %>%
      # assign ntile to stratum variable
      mutate(!!(paste0(var_name, "_strata")) := ntile(!!(sym(var_name),
n=5)) %>%
        mutate(newID = row_number()) %>%
        group_by_(!!(paste0(var_name, "_strata"))) %>%
        mutate(!!(paste0(var_name, "_strata_mean")) :=
          mean(!!(sym(var_name))) %>%
          ungroup())
strata_names1 <- c(strata_names1, paste(subgroups_continuous[i],
"_strata", sep=""))
strata_names2 <- c(strata_names2, paste(subgroups_continuous[i],
"_strata", "_mean", sep=""))
  }
}
all_subgroups <- c(subgroups_categorical, strata_names2)

########################################################################
############################
## simulate pseudo treatment & perform risk and cost analyses
### #
risk_results <- NULL
rate_results <- NULL
cost_results <- NULL
cell_sizes <- NULL

for(jj in 1:nsim){
  message(sprintf("simulation %d %s", jj, date()))

  cell_sizes <- bind_rows(cell_sizes,
                         get_cell_sizes(data_new, names(outcome_data), group="treatment", iter=jj),
                         get_cell_sizes(data2, names(outcome_data), group="control", iter=jj))
}

## Sampling from the population to create a pseudo-treatment group & assigning cost for hypothetical treatment

## Sampling approach 1: stratified sampling within subgroups (risk ratio) when "target_RR" is specified
if (toupper(sample_type) == 'MARGINAL' & is.numeric(target_RR)){
  data_new <- NULL
  data_new <- sample_by_rr(data2, target_RR)
  cell_sizes <- bind_rows(cell_sizes,
                          get_cell_sizes(data_new, names(outcome_data), group="treatment", iter=jj),
                          get_cell_sizes(data2, names(outcome_data), group="control", iter=jj))
}

## Sampling approach 2: stratified sampling within subgroups when using cell proportions specified by user
if(toupper(sample_type) == 'MARGINAL' & !is.numeric(target_RR)){
  data_new <- NULL
  data_new = sample_by_proportion(data=data2, outcomes=names(outcome_data), cell_prop=cell_prop)
}

## Sampling approach 3: using inverse odds sampling (see Westreich, et al. "Transporting trial results using inverse odds of sampling weights." Am J Epid 2017.)
if(toupper(sample_type) == 'CONDITIONAL' & is.numeric(conditional_OR)){
  data_new <- NULL
  data_new = sample_by_inverseodds(data=data2, outcomes=names(outcome_data), effects=conditional_OR)
}

## assigning outcome cost to pseudo-treatment group
data_new$outcome_cost <- data_new$outcome_cost + tmt_cost
## pseudo data

data_pseudo <- bind_rows(data2, data_new)

## assigning pseudo treatment
data_pseudo$zz <- c(rep(0, nrow(data2)), rep(1, nrow(data_new)))

# Performing risk and cost analysis

data_pseudo_temp <- NULL
for(outcome_num in 1:length(Y)) {
  outcome_temp <- paste0('outcome', outcome_num)
  time_outcome_temp <- paste0('time_outcome', outcome_num)
  data_pseudo_temp <- data_pseudo %>%
    mutate(outcome = !!sym(outcome_temp)) %>%
    mutate(time_outcome = !!sym(time_outcome_temp))

  # analysis in full population
  risk_results <- bind_rows(risk_results,
    do_risk_measurement(data_pseudo_temp, iter=jj, outcome=Y[outcome_num],
    subgroup="full population")
  rate_results <- bind_rows(rate_results,
    do_rate_measurement(data_pseudo_temp, iter=jj, outcome=Y[outcome_num],
    subgroup="full population")
  cost_results <- bind_rows(cost_results,
    do_cost_measurement(data_pseudo_temp, iter=jj, outcome="cost",
    subgroup="full population")

  if(length(all_subgroups)>0)
    for(i in 1:length(all_subgroups)) {
      var_name <- all_subgroups[i]
      # get strata of this variable
      strata <- data_pseudo_temp %>%
        group_by_at(.vars=var_name) %>%
        summarise()
      # calculate strata-specific values
      for(j in 1:nrow(strata)) {
        stratum_value <- as.character(strata[j,])
        d <- data_pseudo_temp %>%
          filter(!sym(var_name) == stratum_value)
        if (var_name %in% subgroups_categorical) {
          subgroup <- sprintf("%s_level_%s", var_name, stratum_value)
        } else {
          # !!! what if two strata have the same mean???
          stratum_mean <- round(as.numeric(d[, var_name][1,1]))
          subgroup <- sprintf("%s_%d", var_name, stratum_mean)
        }
        risk_results <- bind_rows(risk_results, do_risk_measurement(d,
          iter=jj, outcome=Y[outcome_num], subgroup=subgroup))
        rate_results <- bind_rows(rate_results, do_rate_measurement(d,
          iter=jj, outcome=Y[outcome_num], subgroup=subgroup))
Gerlinger et al. Using real-world data ... supplementary material

```r
cost_results <- bind_rows(cost_results, do_cost_measurement(d, iter=jj, outcome="cost", subgroup=subgroup))
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}

########################################################
##
##            Organizing results into tables
##
##
########################################################

## tables for risk results
raw_tables <- list()
summary_risk_results <- risk_results %>%
  select(-iter) %>%
  group_by(outcome, subgroup) %>%
  summarise_if(is.numeric, funs(mean(.), na.rm=TRUE)) %>%
  mutate_at(c("rd"), funs(sprintf("%0.5f", .))) %>%
  mutate_at(c("rr", "rl", "r0"), funs(sprintf("%0.3f", .))) %>%
  mutate_at(c("Yd", "Y1", "Y0", "N1", "N0", "N"), funs(sprintf("%0.0f", .)))

## reordering subgroups so that subgroup combinations are at end
n_subgroups <- length(unique(risk_results$subgroup))
temp_order1 <- match(unique(risk_results$subgroup), summary_risk_results$subgroup)
temp_order2 <- NULL
for(i in 1:length(Y)){
  temp_vector <- rep((i-1)*n_subgroups, n_subgroups)
  temp_order2 <- c(temp_order2, temp_vector)
}
temp_order3 <- temp_order1 + temp_order2
summary_risk_results <- summary_risk_results[temp_order3,]
for (i in 1:length(Y)) {
  outcome_name = Y[i]
  key = sprintf("%s_risk_results", outcome_name)
  raw_tables[[key]] = summary_risk_results %>%
    filter(outcome == outcome_name)
}

## tables for rate results
summary_rate_results <- rate_results %>%
  select(-iter) %>%
```

---

The provided text is a snippet of R code. It seems to be part of a script for processing and organizing data related to risk and rate results. The code involves functions such as `bind_rows`, `do_cost_measurement`, `group_by`, `summarise_if`, and others typically used in data analysis and reporting. The blocks of code are organized for readability and clarity, indicating steps for processing data, organizing results into tables for risk and rate results, and reordering subgroups to facilitate the display of results.
group_by(outcome, subgroup) %>%
  summarise_if(is.numeric, funs(mean(., na.rm=TRUE))) %>%
  mutate_at(c("rd"), funs(sprintf("%0.5f", .))) %>%
  mutate_at(c("rr", "r1", "r0"), funs(sprintf("%0.3f", .))) %>%
  mutate_at(c("Yd", "Y1", "Y0", "T1", "T0", "T"), funs(sprintf("%0.0f", .)))

## reordering subgroups so that subgroup combinations are at end
n_subgroups<- length(unique(rate_results$subgroup))
temp_order1<- match(unique(rate_results$subgroup),
  summary_rate_results$subgroup)
temp_order2<- NULL
for(i in 1:length(Y)){
  temp_vector<- rep((i-1)*n_subgroups, n_subgroups)
  temp_order2<- c(temp_order2, temp_vector)
}
temp_order3<- temp_order1 + temp_order2
summary_rate_results<- summary_rate_results[temp_order3,]
for (i in 1:length(Y)) {
  outcome_name = Y[i]
  key = sprintf("%s_rate_results", outcome_name)
  raw_tables[[key]] = summary_rate_results %>%
    filter(outcome == outcome_name)
}

##########################
## Tables for cost results
summary_cost_results <- cost_results %>%
  select(-iter) %>%
  group_by(outcome, subgroup) %>%
  summarise_if(is.numeric, funs(mean(., na.rm=TRUE))) %>%
  mutate_at(vars(starts_with("T")), funs(sprintf("%0.0f", .))) %>%
  mutate_at(vars(starts_with("C")), funs(sprintf("%0.2f", .))) %>%
  mutate_at(c("N1", "N0", "N"), funs(sprintf("%0.0f", .)))

## reordering subgroups so that subgroup combinations are at end
n_subgroups<- length(unique(cost_results$subgroup))
temp_order1<- match(unique(cost_results$subgroup),
  summary_cost_results$subgroup)[1:n_subgroups]
summary_cost_results<- summary_cost_results[temp_order1,]
raw_tables$mean_cost_results <- summary_cost_results %>%
  select_at( vars("subgroup", contains("avg"), contains("N")))
raw_tables$median_cost_results <- summary_cost_results %>%
  select_at( vars("subgroup", contains("med"), contains("N")))
raw_tables$total_cost_results <- summary_cost_results %>%
  select_at( vars("subgroup", contains("tot"), contains("N")))
if (!is.null(cell_sizes)) {
  summary_cell_size_results <- cell_sizes %>%
```
# Using real-world data ...

# group_by(group, cell_num) %>%
# select(-iter, -timestamp, -total_count) %>%
# summarise_all(mean)

raw_tables$treatment_cell_sizes <- summary_cell_size_results %>%
  filter(group == "treatment")

raw_tables$control_cell_sizes <- summary_cell_size_results %>%
  filter(group == "control")

} return(raw_tables)

#########################################################################
####
##                  End of simulation module code
##
#########################################################################

#########################################################################
#########################################################################
###############################
##        Input parameters specified by user
##
##        parameter definitions:
##        nsim: number of simulation runs
##        Y: vector of names specifying up to 3 outcome variables
##        time_var_cost: follow-up time in days (used for calculating cost per day)
##        time_var_outcomes: follow-up time for each outcome (used to calculate rates)
##        subgroups_categorical: vector of names specifying categorical variables for subgroup analyses
##        subgroups_continuous: vector of names specifying continuous variables for subgroup analyses
##        subgroups_combination: vector of names specifying which binary covariates to use when creating new subgroup variable
##        subgroups_combination_operators: vector of "AND" "OR" that determines how variables specified in "subgroups_combination" should be combined
##        cost_var: name of cost variable in dataset
##        tmt_cost: scalar specifying the desired incremental cost per day of the pseudo treatment
##        sample_type: specified as "marginal" or "conditional" to determine how pseudo-treatment group is sampled
```
target_RR: vector specifying the desired effect 
on the relative scale of the pseudo treatment on each outcome (used for "marginal sampling")

conditional_OR: vector specifying desired conditional OR effects of outcomes for pseudo_population (used for "conditional" sampling)

cell_prop: vector specifying the cell proportions for each outcome subgroup (used for 'marginal' sampling) (see input instruction document for details)

p_cost_trimmed: scalar specifying percentage trimmed in right tale of cost variable (# > than 0 but less than or equal to 1)

p_sample: scalar specifying percentage of random subset of population to perform analysis on to speed up computation (# > 0 but less than or equal to 1)

loading data

reading in actual dataset used for analysis

dataset<-
read.csv("//Cifs2/homedir$/Aetion_project/data/data_5_13_19/summary.csv", sep=',' , header=TRUE, as.is=TRUE)

# Simulating Fake Data for Testing (note: simulated data is arbitrary and only used for testing errors in module)
nstudy<- 10000
PO1<- rbinom(nstudy, 1, 0.5)
PO5<- rbinom(nstudy, 1, 0.5)
PO6<- rbinom(nstudy, 1, 0.5)
PO4<- runif(nstudy, 1, 10000)
PO1_censor_time<- runif(nstudy, 1, 1000)
PO5_censor_time<- runif(nstudy, 1, 1000)
PO6_censor_time<- runif(nstudy, 1, 1000)
PO4_censor_time<- runif(nstudy, 1, 1000)
SG1<- rbinom(nstudy, 1, 0.5)
SG2<- rbinom(nstudy, 1, 0.5)
SG6<- rbinom(nstudy, 1, 0.5)
SG16<- rbinom(nstudy, 1, 0.5)
SG5<- runif(nstudy, 1, 100)

dataset<- as.data.frame(cbind(PO1, PO5, PO6, PO4, PO1_censor_time, PO5_censor_time, PO6_censor_time, PO4_censor_time, SG1, SG2, SG6, SG16, SG5))
assign names from dataset for input to sim module

```r
Y = c("PO1", "PO5", "PO6")
time_var_cost = c("PO4\_censor\_time")
time_var_outcomes = c("PO1\_censor\_time", "PO5\_censor\_time", "PO6\_censor\_time")
subgroups_categorical = c("SG1", "SG2", "SG6", "SG16")
subgroups_continuous = c()
subgroups_combination = c("SG1", "SG2")
subgroups_combination_operators = c("AND")
cost_var = c("PO4")
tmt_cost = 0
sample_type = c("marginal")
target_RR = c(0.8, 0.9, 1.2)
conditional_OR = c()
cell_prop = c()
p_cost_trimmed = 0
p_sample = 1
nsim = 1
```

# Running sim module

```r
raw_tables <- sim_module(in_dataset=dataset,
in_Y=Y,
in_subgroups_categorical=subgroups_categorical,
in_subgroups_continuous=subgroups_continuous,
in_subgroups_combination=subgroups_combination,
in_subgroups_combination_operators=subgroups_combination_operators,
in_time_var_cost=time_var_cost,
in_time_var_outcomes=time_var_outcomes,
in_cost_var=cost_var,
in_tmt_cost=tmt_cost,
in_sample_type=sample_type,
in_target_RR=target_RR,
in_conditional_OR=conditional_OR,
in_cell_prop=cell_prop,
in_p_cost_trimmed=p_cost_trimmed,
in_p_sample=p_sample,
in_nsim=nsim)

# view results from raw_tables
raw_tables
```
### Making tables and figures from output in raw_tables (only run the code below if you want to output results to JSON)

### Risk difference plots (need to change name of title and plot_var to specific outcome you want plotted)

```r
# risk_table <- risk_table %>%
# mutate_at(c("rd"), as.numeric) %>%
# mutate(subgroup = factor(subgroup),
#        subgroup = factor(subgroup, levels = rev(levels(subgroup))))

# reordering smallest to largest and creating labels
risk_table <- risk_table[order(risk_table$rd),]
risk_table$subgroup <- factor(risk_table$subgroup, levels=risk_table$subgroup[order(risk_table$rd)])
perc_population <- round(100 * (as.numeric(risk_table$N) / max(as.numeric(risk_table$N))))
less_than_one <- perc_population < 1
labs1 <- risk_table$subgroup
labs2 <- paste(perc_population, '%', sep='')
labs2[less_than_one] <- '<1%'

ggplot(risk_table, aes(x=rd, y=as.numeric(subgroup))) +
  plain_theme +
  ggtitle(title) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Subgroup") +
  geom_point(col="black", size=2) +
  geom_line() +
  scale_x_continuous(name="Risk Difference Across Treatment Groups") +
  scale_y_continuous(breaks=1:length(labs1),
                    labels=labs1,
                    sec.axis=sec_axis(~.,
                    breaks=1:length(labs2),
                    labels=labs2,
                    name='subgroup size (% of full population)'))
```
 Gerlinger et al. Using real-world data ...

 supplementary material

## rate difference plots (need to change name of title and plot_var to specific outcome you want plotted)

title<- 'Rate Results for P01'
plot_var<- 'P01_rate_results'
rate_table<- raw_tables[[plot_var]]

rate_table <- rate_table %>%
  mutate_at(c("rd"), as.numeric) %>%
  mutate(subgroup = factor(subgroup),
         subgroup = factor(subgroup, levels = rev(levels(subgroup))))

# reordering smallest to largest and creating labels
rate_table<- rate_table[order(rate_table$rd),]
rate_table$subgroup<- factor(rate_table$subgroup,
                            levels=rate_table$subgroup[order(rate_table$rd)])
perc_population<- round(100 * (as.numeric(rate_table$T) /
                              max(as.numeric(rate_table$T))))
less_than_one<- perc_population < 1
labs1<- rate_table$subgroup
labs2<- paste(perc_population, '%', sep='')
labs2[less_than_one]<- '<1%

ggplot(rate_table, aes(x=rd, y=as.numeric(subgroup))) +
  plain_theme +
  ggtitle(title) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Subgroup") +
  geom_point(col="black", size=2) +
  geom_line() +
  scale_x_continuous(name="Rate Difference Across Treatment Groups") +
  scale_y_continuous(breaks=1:length(labs1),
                   labels=labs1,
                   sec.axis=sec_axis(~.,
                         breaks=1:length(labs2),
                         labels=labs2,
                         name='subgroup size (% of total follow-up time)'))

## event plots (need to change name of title and plot_var to specific outcome you want plotted)

# reordering smallest to largest and creating labels
rate_table<- rate_table[order(rate_table$rd),]
rate_table$subgroup<- factor(rate_table$subgroup,
                            levels=rate_table$subgroup[order(rate_table$rd)])
perc_population<- round(100 * (as.numeric(rate_table$T) /
                              max(as.numeric(rate_table$T))))
less_than_one<- perc_population < 1
labs1<- rate_table$subgroup
labs2<- paste(perc_population, '%', sep='')
labs2[less_than_one]<- '<1%

ggplot(rate_table, aes(x=rd, y=as.numeric(subgroup))) +
  plain_theme +
  ggtitle(title) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Subgroup") +
  geom_point(col="black", size=2) +
  geom_line() +
  scale_x_continuous(name="Rate Difference Across Treatment Groups") +
  scale_y_continuous(breaks=1:length(labs1),
                   labels=labs1,
                   sec.axis=sec_axis(~.,
                         breaks=1:length(labs2),
                         labels=labs2,
                         name='subgroup size (% of total follow-up time)'))
title<- 'Differences in Outcome Events for PO1'
plot_var<- 'PO1_risk_results'
rate_table<- raw_tables[[plot_var]]

risk_table <- risk_table %>%
  mutate_at(c("Yd"), as.numeric) %>%
  mutate(subgroup = factor(subgroup),
         subgroup = factor(subgroup, levels = rev(levels(subgroup))))

## reordering smallest to largest and creating labels
risk_table <- risk_table[order(risk_table$Yd),]
risk_table$subgroup <- factor(risk_table$subgroup, levels=risk_table$subgroup[order(risk_table$Yd)])
perc_population <- round(100 * (as.numeric(risk_table$N) / max(as.numeric(risk_table$N))))
less_than_one <- perc_population < 1
labs1 <- risk_table$subgroup
labs2 <- paste(perc_population, '%', sep='')
labs2[less_than_one] <- '<1%'

ggplot(risk_table, aes(x=Yd, y=as.numeric(subgroup))) +
  plain_theme +
  ggtitle(title) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Subgroup") +
  geom_point(col="black", size=2) +
  geom_line() +
  scale_x_continuous(name="Difference in Total Outcome Events Across Treatment Groups") +
  scale_y_continuous(breaks=1:length(labs1),
                   labels=labs1,
                   sec.axis=sec_axis(~.,
                                    breaks=1:length(labs2),
                                    labels=labs2,
                                    name='subgroup size (% of full population)'))

## plots for average cost difference, median cost difference, and total cost difference

## average cost difference
title<- 'Average Difference in Cost'
plot_var<- 'mean_cost_results'
cost_table<- raw_tables[[plot_var]]
cost_outcome = 'cd_avg'
cost_table <- cost_table %>%
  mutate(cd = !!sym(cost_outcome),
         subgroup = factor(subgroup),
         subgroup = factor(subgroup, levels = rev(levels(subgroup)))) %>%
mutate_at(c("cd"), as.numeric)

cost_table<- cost_table[order(cost_table$cd),]
cost_table$subgroup<- factor(cost_table$subgroup,
levels=cost_table$subgroup[order(cost_table$cd)])
perc_population<- round(100 * (as.numeric(cost_table$N) / 
max(as.numeric(cost_table$N))))
less_than_one<- perc_population < 1
labs1<- cost_table$subgroup
labs2<- paste(perc_population, '\%', sep='')
labs2[less_than_one]<- '<1%'

ggplot(cost_table, aes(x=cd, y=as.numeric(subgroup))) +
plain_theme +
ggtitle(title) +
theme(plot.title = element_text(hjust = 0.5)) +
ylab("Subgroup") +
geom_line() +
scale_x_continuous(name="Cost Difference ($) Across Treatment Groups") +
scale_y_continuous(breaks=1:length(labs1),
labels=labs1,
sec.axis=sec_axis(~.,
breaks=1:length(labs2),
labels=labs2,
name='size of subgroup (% of full population)'))

## median cost difference

title<- 'Median Difference in Cost'
plot_var<- 'median_cost_results'
cost_table<- raw_tables[[plot_var]]

cost_outcome = 'cd_med'
cost_table <- cost_table %>%
mutate(cd = !!sym(cost_outcome),
        subgroup = factor(subgroup),
        subgroup = factor(subgroup, levels = rev(levels(subgroup)))) %>%
mutate_at(c("cd"), as.numeric)

cost_table<- cost_table[order(cost_table$cd),]
cost_table$subgroup<- factor(cost_table$subgroup,
levels=cost_table$subgroup[order(cost_table$cd)])
perc_population<- round(100 * (as.numeric(cost_table$N) / 
max(as.numeric(cost_table$N))))
less_than_one<- perc_population < 1
labs1<- cost_table$subgroup
labs2<- paste(perc_population, '\%', sep='')
labs2[less_than_one]<- '<1%'

ggplot(cost_table, aes(x=cd, y=as.numeric(subgroup))) +
plain_theme +
ggtitle(title) +
theme(plot.title = element_text(hjust = 0.5)) +
ylab("Subgroup") +
geom_point(col="black", size=2) +
geom_line() +
scale_x_continuous(name="Cost Difference ($) Across Treatment Groups") +
scale_y_continuous(breaks=1:length(labs1),
  labels=labs1,
  sec.axis=sec_axis(~.,
    breaks=1:length(labs2),
    labels=labs2,
    name='size of subgroup (% of full population)'))

## total cost difference
title<- 'Total Difference in Cost'
plot_var<- 'total_cost_results'

cost_table<- raw_tables[[plot_var]]

cost_outcome = 'cd_tot'
cost_table <- cost_table %>%
  mutate(cd = !!sym(cost_outcome),
    subgroup = factor(subgroup),
    subgroup = factor(subgroup, levels = rev(levels(subgroup))))
%>%
  mutate_at(c("cd"), as.numeric)

cost_table<- cost_table[order(cost_table$cd),]
cost_table$subgroup<- factor(cost_table$subgroup,
  levels=cost_table$subgroup[order(cost_table$cd)])
perc_population<- round(100 * (as.numeric(cost_table$N) /
  max(as.numeric(cost_table$N))))
less_than_one<- perc_population < 1
labs1<- cost_table$subgroup
labs2<- paste(perc_population, '%', sep='')
labs2[less_than_one]<- '<1%'

ggplot(cost_table, aes(x=cd, y=as.numeric(subgroup))) +
  plain_theme +
  ggtitle(title) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ylab("Subgroup") +
  geom_point(col="black", size=2) +
  geom_line() +
  scale_x_continuous(name="Cost Difference ($) Across Treatment Groups") +
  scale_y_continuous(breaks=1:length(labs1),
    labels=labs1,
    sec.axis=sec_axis(~.,
      breaks=1:length(labs2),
      labels=labs2,
      name='size of subgroup (% of full population)'))

##########################################################################
##########################
functions for making tables for all results

```r
make_risk_table <- function(risk_table, title) {
  risk_table <- risk_table %>%
    rename(`Subgroup` = subgroup,
          `Outcome` = outcome,
          `Risk Difference` = rd,
          `Risk Ratio` = rr,
          `Risk Exposed` = r1,
          `Risk Unexposed` = r0,
          `Exposed Events` = Y1,
          `Unexposed Events` = Y0,
          `Event Difference` = Yd,
          `N Exposed` = N1,
          `N Unexposed` = N0,
          `N Total` = N)
  return(finalize_table(risk_table, title))
}

make_rate_table <- function(rate_table, title) {
  rate_table <- rate_table %>%
    rename(`Subgroup` = subgroup,
          `Outcome` = outcome,
          `rate Difference` = rd,
          `rate Ratio` = rr,
          `rate Exposed` = r1,
          `rate Unexposed` = r0,
          `Exposed Events` = Y1,
          `Unexposed Events` = Y0,
          `Event Difference` = Yd,
          `Time Exposed` = T1,
          `Time Unexposed` = T0,
          `Time Total` = T)
  return(finalize_table(rate_table, title))
}

make_cost_table <- function(cost_table, title) {
  if ("cd_avg" %in% colnames(cost_table)) {
    cost_table <- cost_table %>%
      rename(`Subgroup` = subgroup,
             `mean cost difference ($)` = cd_avg,
             `mean exposed cost/day ($)` = C1_avg,
             `mean unexposed cost/day ($)` = C0_avg,
             `mean exposed time (days)` = T1_avg,
             `mean unexposed time (days)` = T0_avg)
  }
  if ("cd_med" %in% colnames(cost_table)) {
    cost_table <- cost_table %>%
      rename(`Subgroup` = subgroup,
```
`median cost difference ($)` = cd_med,  
`median exposed cost/day ($)` = C1_med,  
`median unexposed cost/day ($)` = C0_med,  
`median exposed time (days)` = T1_med,  
`median unexposed time (days)` = T0_med)

if ("cd_tot" %in% colnames(cost_table)) {
  cost_table <- cost_table %>%
    rename('Subgroup' = subgroup,  
          'total cost difference ($)' = cd_tot,  
          'total exposed cost/day ($)' = C1_tot,  
          'total unexposed cost/day ($)' = C0_tot,  
          'total exposed time (days)' = T1_tot,  
          'total unexposed time (days)' = T0_tot)
}

return(finalize_table(cost_table, title))

make_cell_size_table <- function(cell_size_table, title) {
  cell_size_table <- cell_size_table %>%
    rename('Cell Number' = cell_num,  
           'Frequency' = frequency,  
           'Proportion' = proportion)
  return(finalize_table(cell_size_table, title))
}

# make the tables
risk_tables <- map(Y, function(x) {
  table_key = sprintf("%s_risk_results", x)
  title = sprintf("Risk Analysis for Outcome %s", x)
  table = make_risk_table(raw_tables[[table_key]], title)
  return(table)
})

mean_cost_table = make_cost_table(raw_tables$mean_cost_results, "Average Cost Differences Across Treatment Groups")
med_cost_table = make_cost_table(raw_tables$median_cost_results, "Median Cost Differences Across Treatment Groups")
tot_cost_table = make_cost_table(raw_tables$total_cost_results, "Total Cost Differences Across Treatment Groups")
all_tables <- risk_tables %>%
  append(list(mean_cost_table)) %>%
  append(list(med_cost_table)) %>%
  append(list(tot_cost_table))

if (!is.null(raw_tables$treatment_cell_sizes)) {
  treatment_cell_table =
  make_cell_size_table(raw_tables$treatment_cell_sizes, "Cell Sizes (treatment Drug)")
control_cell_table = make_cell_size_table(raw_tables$control_cell_sizes, "Cell Sizes (Control Drug)"
all_tables <- all_tables %>%
  append(list(treatment_cell_table)) %>%
  append(list(control_cell_table))

# add captions
all_tables <- map2(all_tables, 1:length(all_tables), function(table, i) {
  table$caption = sprintf("Table %0.0f", i)
  return(table)
})

all_tables
check_input.R

check_input<- function(dataset,
                      Y,
                      time_var_cost,
                      time_var_outcomes,
                      subgroups_categorical,
                      subgroups_continuous,
                      subgroups_combination,
                      subgroups_combination_operators,
                      cost_var,
                      tmt_cost,
                      sample_type,
                      target_RR,
                      conditional_OR,
                      cell_prop,
                      p_cost_trimmed,
                      p_sample,
                      nsim, ...) {

############################################
##    check input for Y
############################################

if (length(Y) == 0) {
  stop(paste("Need to specify variable name(s) for outcome(s)", sep=""))
}

if (length(Y) > 3) {
  stop(paste("Simulation module currently only supports up to 3 outcome
variables", sep=""))
}

has_Y<- Y %in% names(dataset)
for(i in 1:length(Y)) {
  if (has_Y[i] == FALSE) {
    stop(paste("Variable ", Y[i], ", only has 1
level. Cannot perform analyses for this outcome", sep=""))
  }
}

for(i in 1:length(Y)) {
  outcome <- dataset[, Y[i]]

  if (length(unique(outcome)) < 2) {
    stop(paste("Outcome variable ", Y[i], ", only has 1
level. Cannot perform analyses for this outcome", sep=""))
  }
  if (length(unique(outcome)) > 2) {
    stop(paste("Outcome variable ", Y[i], ", has more
than 1 level. Outcome variables need to be a numeric or logical vector
with 2 levels (e.g., vector of 1's and 0's or TRUE/FALSE)", sep=""))
  }
}
outcome <- as.numeric(as.logical(outcome[!is.na(outcome)]))  # excluding missing values and setting as numeric (if not numeric or logical we get NA)

if( !(is.numeric(outcome)) | sum(!is.na(outcome))==0){
  stop(paste("Outcome variable ", "Y[", i, "], " needs to be a numeric or logical vector (e.g., vector of 1's and 0's or TRUE/FALSE)", sep=""))
}

###########################################
##    check input for time_var_cost
###########################################

if(length(time_var_cost)==0){
  stop(paste("Need to specify variable name for follow-up time", sep=""))
}

if(length(time_var_cost)>1){
  stop(paste("Simulation module currently only supports one time variable", sep=""))
}

time_name<- time_var_cost %in% names(dataset)
if (time_name==FALSE) {
  stop(paste("Variable ", time_var_cost, " for time is not a variable name in the dataset", sep=""))
}

if( !is.numeric(dataset[,time_var_cost]) ){
  warning(paste("Input for follow-up time is not numeric. Variable ", time_var_cost, " will be coerced to numeric. This may produce missing values", sep=""))
}

time_numeric<- as.numeric(dataset[,time_var_cost])
if(sum(!is.na(time_numeric))==0){
  stop(paste("Variable ", time_var_cost, " is not valid. All values set as missing after being coerced to numeric", sep=""))
}

if(!is.numeric(dataset[,time_var_cost]) & sum(is.na(time_numeric)>0)){
  warning(paste("Some values for time variable ", time_var_cost, " are not valid (set as missing after being coerced to numeric", sep=""))
}
## check input for time_var_outcomes

```r
if(length(time_var_outcomes)==0 | length(time_var_outcomes)!=length(Y)){
  stop(paste("Need to specify variable name for follow-up time for each outcome. Number of outcome follow-up time variables needs to equal number of outcomes", sep=""))
}
```

```r
time_name<- NULL
time_name<- time_var_outcomes %in% names(dataset)
for(i in 1:length(time_name)){
  if (time_name[i]==FALSE) {
    stop(paste("Variable ", "\", time_var_outcomes[i], "\", " ", "for outcome follow-up time is not a variable name in the dataset", sep=""))
  }
  if( !is.numeric(dataset[,time_var_outcomes[i]]) ){
    warning(paste("Input for follow-up time is not numeric. Variable ", "\", time_var_outcomes[i], "\", " ", "will be coerced to numeric. This may produce missing values", sep=""))
  }
  time_numeric<- as.numeric(dataset[,time_var_outcomes[i]])
  if(sum(!is.na(time_numeric))==0){
    stop(paste("Variable ", "\", time_var_outcomes[i], "\", " ", "is not valid. All values set as missing after being coerced to numeric", sep=""))
  }
  if(!is.numeric(dataset[,time_var_outcomes[i]]) & sum(is.na(time_numeric)>0)){
    warning(paste("Some values for time variable ", "\", time_var_outcomes[i], "\", " ", "are not valid (set as missing) after being coerced to numeric", sep=""))
  }
}
```

## check input for cost_var

```r
if(length(cost_var)==0){
  stop(paste("Need to specify variable name for treatment cost", sep=""))
}
```

```r
if(length(cost_var)>1){
  stop(paste("Simulation module currently only supports one outcome cost variable", sep=""))
}
```
```r
cost_name <- cost_var %in% names(dataset)
if (cost_name==FALSE) {
  stop(paste("Variable ", "\"", cost_var, "\"", " ", "for cost is not a variable name in the dataset", sep=""))
}

if( !is.numeric(dataset[,cost_var]) ){
  warning(paste("Input for cost is not numeric. Variable ", "\"", cost_var, "\"", " ", "will be coerced to numeric. This may produce missing values", sep=""))
}

cost_numeric <- as.numeric(dataset[,cost_var])
if(sum(!is.na(cost_numeric))==0){
  stop(paste("Variable ", "\"", cost_var, "\"", " ", "is not valid. All values set as missing after being coerced to numeric", sep=""))
}

if(!is.numeric(dataset[,cost_var]) & sum(is.na(cost_numeric)>0)){
  warning(paste("Some values for cost variable ", "\"", cost_var, "\"", " ", "are not valid (set as missing) after being coerced to numeric", sep=""))
}

#########################################################################
##      check input for subgroups_categorical & subgroups_continuous
#########################################################################

has_subcat <- subgroups_categorical %in% names(dataset)
if(length(subgroups_categorical)>0){
  for(i in 1:length(subgroups_categorical)){
    if (has_subcat[i]==FALSE) {
      stop(paste("Variable ", "\"", subgroups_categorical[i], "\"", " ", "for categorical subgroup is not a variable name in the dataset", sep=""))
    }
  }
}

has_subcont <- subgroups_continuous %in% names(dataset)
if(length(subgroups_continuous)>0){
  for(i in 1:length(subgroups_continuous)){
    if (has_subcont[i]==FALSE) {
      stop(paste("Variable ", "\"", subgroups_continuous[i], "\"", " ", "for continuous subgroup is not a variable name in the dataset", sep=""))
    }
  }

#########################################################################
## check input for subgroups_categorical & subgroups_continuous
#########################################################################
```

```
## check input for subgroups_combination & subgroups_combination_operators

has_subcatcomb1 <- subgroups_combination %in% names(dataset)
size_sub <- (length(subgroups_combination) - 1)
test_operators <- (toupper(subgroups_combination_operators) %in% "AND") | (toupper(subgroups_combination_operators) %in% "OR")

if(length(subgroups_combination) == 1){
  stop(paste("Invalid input for 'subgroups_combination' field. Need to leave field empty or specify >1 variable", sep=""))
}

if(length(subgroups_combination) > 1){
  for(i in 1:length(subgroups_combination)){
    if (has_subcatcomb1[i] == FALSE) {
      stop(paste("Variable ", ", "", subgroups_combination[i], ", "", " ", " for subgroups combination is not a variable name in the dataset", sep=""))
    }
    if (length(has_subcatcomb1[i]) > 2) {
      stop(paste("Variable ", ", "", subgroups_combination[i], ", "", " ", " for subgroups combination has more than 2 levels. Simulation module currently supports only binary variables for subgroup combinations", sep=""))
    }
    if(is.numeric(dataset[,subgroups_combination[i]])==FALSE) & (is.logical(dataset[,subgroups_combination[i]])==FALSE) {
      stop(paste("Variable ", ", "", subgroups_combination[i], ", "", " ", " for subgroups combination needs to be a numeric vector of 1's and 0's or a logical vector of 'TRUE' and 'FALSE'")
    }
  }
  if((length(subgroups_combination_operators) != (length(subgroups_combination) - 1)) | any(test_operators=='FALSE')) {
    stop(paste("Invalid input for 'subgroups_combination_operators'. Need to specify a vector containing a series of 'AND' or 'OR' statements to determine how variables listed in 'subgroups_combination' field will be combined. For example, if 2 variables are listed in 'subgroups_combination' then one 'AND' or 'OR' statement must be specified. If 3 variables are listed, then 2 logical operators must be specified", sep=""))
  }
}

## check input for sample_type
```
has_sample_type = (length(sample_type) > 0)
if(has_sample_type){
    stop(paste("Please specify input for 'sample type' as 'conditional' or 'marginal'"))
}

if(toupper(sample_type) != 'CONDITIONAL' & toupper(sample_type) != 'MARGINAL'){
    stop(paste("Invalid input for 'sample type'. Please specify either 'conditional' or 'marginal' for input for 'sample type'"))
}

has_target_RR      = (length(target_RR) > 0)
has_cell_prop      = (length(cell_prop) > 0)
has_conditional_OR = (length(conditional_OR) > 0)

if(toupper(sample_type)=='MARGINAL' & has_conditional_OR){
    stop(paste("Input for 'conditional OR' cannot be specified when 'sample type' is specified as 'marginal'"))
}

if(toupper(sample_type)=='MARGINAL' & (!has_target_RR & !has_cell_prop)){
    stop(paste("Input for either 'target RR' or 'cell proportions' is needed when 'sample type' is specified as 'marginal'"))
}

if(toupper(sample_type)=='CONDITIONAL' & !has_conditional_OR){
    stop(paste("Input for 'conditional odds ratios' must be specified when 'sample type' is specified as 'conditional'"))
}

if(toupper(sample_type)=='CONDITIONAL' & has_target_RR){
    stop(paste("Input for target risk ratios cannot be specified when 'sample type' is specified as 'conditional'. Either change sample_type to 'marginal' or specify 'conditional OR' instead of 'target RR'"))
}

if(toupper(sample_type)=='CONDITIONAL' & has_cell_prop){
    stop(paste("Input for cell proportions cannot be specified when 'sample type' is specified as 'conditional'. Either change sample_type to 'marginal' or specify 'conditional OR' instead of 'cell proportions'"))
}

if(has_conditional_OR & has_target_RR){
stop(paste("Input for 'target risk ratios' and input for 'conditional odds ratios' cannot both be specified. Please input values for either 'risk ratios' or 'conditional odds ratios'"))

if(has_conditional_OR & has_cell_prop) {
  stop(paste("Input for 'cell proportions' and input for 'conditional odds ratios' cannot both be specified. Please input values for either 'cell proportions' or 'conditional odds ratios'"))
}

if(has_target_RR & has_cell_prop) {
  stop(paste("Input for target risk ratios and input for cell proportions cannot both be specified. Please input values for either risk ratios or cell proportions"))
}

if(!(has_target_RR | has_cell_prop | has_conditional_OR)) {
  stop(paste("Please specify input for either 'target risk ratios' or for 'cell proportions' or for 'conditional odds ratios'"))
}

if((has_target_RR) & length(target_RR) != length(Y)) {
  stop(paste("The number of specified values for target risk ratios must match the number of outcomes"))
}

if(has_target_RR)
  for(i in 1:length(target_RR)) {
    if(target_RR[i] <= 0 | !is.numeric(target_RR[i]) ){
      stop(paste("Input for 'target risk ratios' is not valid. Please specify a numeric scalar with values greater than or equal to 0", sep=""))
    }
  }

if((has_conditional_OR) & length(conditional_OR) != length(Y)) {
  stop(paste("The number of specified values for conditional odds ratios must match the number of outcomes"))
}

if(has_conditional_OR)
  for(i in 1:length(conditional_OR)) {
    if(conditional_OR[i] <= 0 | !is.numeric(conditional_OR[i]) ){
      stop(paste("Input for 'conditional odds ratios' is not valid. Please specify a numeric scalar with values greater than or equal to 0", sep=""))
    }
  }

if(has_cell_prop & length(cell_prop) != 2^(length(Y))) {

Gerlinger et al. Using real-world data ... supplementary material

stop(paste("Invalid number of cell proportions. Please specify a
proportion for each cell (subgroup for every possible outcome
combination)" , sep=" "))

if(has_cell_prop){
    for(i in 1:length(cell_prop)){
        if(cell_prop[i] < 0 | cell_prop[i] > 1 | !is.numeric(cell_prop[i])){
            stop(paste("Input for at least one cell size is not valid. Please
specify a numeric scalar between 0 & 1", sep=""))
        }
    }
}

##########################################################################
#######################################################
##  check correlation between outcomes for marginal sampling or if
events for one outcome is subset of events for another outcome
#######################################################
##########################################################################

if(length(Y)==2 & length(target_RR)==2){
    test_cor1<- NULL
    test_cor1<- cor(dataset[,Y[1]], dataset[,Y[2]])
    if(test_cor1==1){
        stop(paste("Correlation between outcome", Y[1], "and outcome", Y[2],
"is 1. Cannot perform the marginal sampling with 'target risk ratio' field
specified. Either change the outcomes or try 'marginal' sampling with
'cell proportions' specified or 'conditional' sampling type with
conditional odds ratios specified", sep=" "))
    }

test_sub1<- test_sub2<- NULL
    test_sub1<- sum(dataset[,Y[1]]==1 & dataset[,Y[2]]==0) == 0
    test_sub2<- sum(dataset[,Y[1]]==0 & dataset[,Y[2]]==1) == 0
    if(test_sub1==1){
        stop(paste("Cannot perform the marginal sampling with 'target risk
ratio' field specified because there are no individuals for which
outcome", Y[1], "= 1 and outcome", Y[2], "= 0. Either change the outcomes
or try 'marginal' sampling with 'cell proportions' specified or
'conditional' sampling type with conditional odds ratios specified", sep=" "))
    }else{
        stop(paste("Cannot perform the marginal sampling with 'target risk
ratio' field specified because there are no individuals for which
outcome", Y[1], "= 0 and outcome", Y[2], "= 1. Either change the outcomes
or try 'marginal' sampling with 'cell proportions' specified or
'conditional' sampling type with conditional odds ratios specified", sep=" "))
    }
}
if(length(Y)==3 & length(target_RR)==3){
    test_cor1<- test_cor2<- test_cor3<- NULL
    test_cor1<- cor(dataset[,Y[1]], dataset[,Y[2]])
    test_cor2<- cor(dataset[,Y[1]], dataset[,Y[3]])
    test_cor3<- cor(dataset[,Y[2]], dataset[,Y[3]])
    if(test_cor1==1){
        stop(paste("Correlation between outcome", Y[1], "and outcome", Y[2], "is 1. Cannot perform the marginal sampling with 'target risk ratio' field specified. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
    }
    if(test_cor2==1){
        stop(paste("Correlation between outcome", Y[1], "and outcome", Y[3], "is 1. Cannot perform the marginal sampling with 'target risk ratio' field specified. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
    }
    if(test_cor3==1){
        stop(paste("Correlation between outcome", Y[2], "and outcome", Y[3], "is 1. Cannot perform the marginal sampling with 'target risk ratio' field specified. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
    }
    test_sub1<- test_sub2<- test_sub3<- test_sub4<- test_sub5<- test_sub6<- NULL
    test_sub1<- sum(dataset[,Y[1]]==1 & dataset[,Y[2]]==0) == 0
    test_sub2<- sum(dataset[,Y[1]]==0 & dataset[,Y[2]]==1) == 0
    test_sub3<- sum(dataset[,Y[1]]==1 & dataset[,Y[3]]==0) == 0
    test_sub4<- sum(dataset[,Y[1]]==0 & dataset[,Y[3]]==1) == 0
    test_sub5<- sum(dataset[,Y[2]]==1 & dataset[,Y[3]]==0) == 0
    test_sub6<- sum(dataset[,Y[2]]==0 & dataset[,Y[3]]==1) == 0
    if(test_sub1==1){
        stop(paste("Cannot perform the marginal sampling with 'target risk ratio' field specified because there are no individuals for which outcome", Y[1], ": = 1 and outcome", Y[2], ": = 0. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
    }
    if(test_sub2==1){
        stop(paste("Cannot perform the marginal sampling with 'target risk ratio' field specified because there are no individuals for which outcome", Y[1], ": = 0 and outcome", Y[2], ": = 1. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
    }
}
if(test_sub3==1){
  stop(paste("Cannot perform the marginal sampling with 'target risk ratio' field specified because there are no individuals for which outcome", Y[1], "= 1 and outcome", Y[3], "= 0. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
}

if(test_sub4==1){
  stop(paste("Cannot perform the marginal sampling with 'target risk ratio' field specified because there are no individuals for which outcome", Y[1], "= 0 and outcome", Y[3], "= 1. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
}

if(test_sub5==1){
  stop(paste("Cannot perform the marginal sampling with 'target risk ratio' field specified because there are no individuals for which outcome", Y[2], "= 1 and outcome", Y[3], "= 0. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
}

if(test_sub6==1){
  stop(paste("Cannot perform the marginal sampling with 'target risk ratio' field specified because there are no individuals for which outcome", Y[2], "= 0 and outcome", Y[3], "= 1. Either change the outcomes or try 'marginal' sampling with 'cell proportions' specified or 'conditional' sampling type with conditional odds ratios specified", sep=" "))
}

##################################################################
##  check input for tmt_cost, p_cost_trimmed, p_sample, & nsim
##################################################################

if(length(tmt_cost) != 1 | !is.numeric(tmt_cost) ){
  stop(paste("Input for treatment cost is not valid. Please specify a numeric scalar", sep=" "))
}

if(length(p_cost_trimmed) != 1 | p_cost_trimmed<0 | p_cost_trimmed >= 1 | !is.numeric(p_cost_trimmed)){
  stop(paste("Input for percent trimmed on cost not valid. Enter value >=0 or <1", sep=" "))
}

if(length(p_sample) != 1 | p_sample<0 | p_sample > 1 | !is.numeric(p_sample)){

stop(paste("Input for percentage to randomly sample from data for analyses not valid. Enter value >0 or \leq 1", sep=""))

if(length(nsim) != 1 | nsim==0 | !is.numeric(nsim) ){
  stop(paste("Input for nsim is not valid. Please specify an integer > 0", sep=""))
}
Sampling.R

```r
f <- function(beta, gamma, rho, eps = 1e-15) {
  a <- rho - 1
  b <- (beta + gamma) * (1 - rho) - 1
  c_ <- rho * beta * gamma
  if (abs(a) < eps) {
    z <- -c_ / b
  } else {
    d <- b^2 - 4 * a * c_
    if (d < eps * eps) s <- 0 else s <- c(-1, 1)
    z <- (-b + s * sqrt(max(0, d))) / (2 * a)
  }
  y <- vapply(z, function(a) zapsmall(matrix(c(a, gamma - a, beta - a, 1 + a - beta - gamma), 2, 2)),
              matrix(0.0, 2, 2))
  i <- apply(y, 3, function(u) all(u >= 0))
  return(y[, , i])
}
```

```r
helper_f <- function(y1, y2, RR1, RR2, N_num = FALSE, N_total) {
  table1 <- table(y1, y2)
  table2 <- table1 / N_total
  beta <- N_num / N_total
  if (N_num == FALSE) {
    beta <- rowSums(table2)[2]
  }
}
gamma <- colSums(table2)[2]
rho <- (table2[1,1]*table2[2,2]) / (table2[1,2]*table2[2,1])

## calling function f (see above)
table_new<- f(beta*RR1, gamma*RR2, rho)

n4<- round( table_new[1,1] * N_total )
n3<- round( table_new[2,1] * N_total )
n2<- round( table_new[1,2] * N_total )
n1<- round( table_new[2,2] * N_total )

return(c(n1, n2, n3, n4))
}

sample_by_rr <- function(data, target_RR) {
  num_outcomes = length(target_RR)
  data_new <- NULL
  if (num_outcomes == 1) {
    a <- nrow(filter(data, outcome1 == 1))
    b <- nrow(filter(data, outcome1 == 0))
    data1 <- sample_n(filter(data, outcome1 == 1), (target_RR * a),
                      replace=TRUE)
    data0 <- sample_n(filter(data, outcome1 == 0), ((1-target_RR)*a + b),
                      replace=TRUE)
    data_new <- bind_rows(data0, data1)
  } else if (num_outcomes == 2) {
    target_RR1<- target_RR[1]
    target_RR2<- target_RR[2]

    ## epitab produces NAs for one of the test dataset???
    # t = data.frame(epitab(data$outcome1, data$outcome2, method="oddsratio")$tab)
    # get the row for outcome = TRUE
    # rho <- t[which(rownames(t) == "TRUE"), ]$oddsratio

    ## calculating rho manually instead of using epitab
    table1<- table(data$outcome1, data$outcome2)
    table2<- table1/sum(table1)
    rho <- (table2[1,1]*table2[2,2]) / (table2[1,2]*table2[2,1])

    beta <- sum(data$outcome1) / nrow(data)
    gamma <- sum(data$outcome2) / nrow(data)

    ## calling function f (see above)
    table_new<- f(beta*target_RR1, gamma*target_RR2, rho)
    # or_new<-
    (table_new[1,1]*table_new[2,2])/(table_new[1,2]*table_new[2,1])

    cell1_new<- filter(data, outcome1 == 0 & outcome2 == 0) %>%
                   sample_n(round(table_new[2,2] * nrow(data)),
                             replace=TRUE)
cell2_new <- filter(data, outcome1 == 1 & outcome2 == 0) %>%
  sample_n(round(table_new[1,2] * nrow(data)),
  replace=TRUE)
cell3_new <- filter(data, outcome1 == 0 & outcome2 == 1) %>%
  sample_n(round(table_new[2,1] * nrow(data)),
  replace=TRUE)
cell4_new <- filter(data, outcome1 == 1 & outcome2 == 1) %>%
  sample_n(round(table_new[1,1] * nrow(data)),
  replace=TRUE)
data_new = bind_rows(cell1_new, cell2_new, cell3_new, cell4_new)
} else if (num_outcomes == 3) {
  target_RR1 <- target_RR[1]
target_RR2 <- target_RR[2]
target_RR3 <- target_RR[3]

  ## apply function f for first two outcomes to get cell numbers for
  margins of third outcome
  new_cell_sizes1 <- helper_f(y1=data$outcome1,
    y2=data$outcome2,
    RR1=target_RR1,
    RR2=target_RR2,
    N_num=FALSE,
    N_total=dim(data)[1])

  n_cell1_new <- new_cell_sizes1[1]
n_cell2_new <- new_cell_sizes1[2]
n_cell3_new <- new_cell_sizes1[3]
n_cell4_new <- new_cell_sizes1[4]

  ## apply function f for third and second outcomes within strata of
  first outcome (outcome1=1)
  data_strata1 <- data[data$outcome1==1, ]
  strata1_N_new <- target_RR1 * dim(data_strata1)[1]        ## specifying
  total number to sample from strata1
  new_cell_sizes2 <- helper_f(y1=data_strata1$outcome2,
    y2=data_strata1$outcome3,
    RR1=1,
    RR2=target_RR3,
    N_num=n_cell4_new,
    N_total=strata1_N_new)

  stratal1_cell1_original <- data_stratal[data_stratal$outcome2==0 &
    data_stratal$outcome3==0,]
  stratal1_cell2_original <- data_stratal[data_stratal$outcome2==1 &
    data_stratal$outcome3==0,]
  stratal1_cell3_original <- data_stratal[data_stratal$outcome2==0 &
    data_stratal$outcome3==1,]
  stratal1_cell4_original <- data_stratal[data_stratal$outcome2==1 &
    data_stratal$outcome3==1,]

  stratal_index_cell11_new <- sample(1:nrow(stratal1_cell1_original),
    new_cell_sizes2[1], replace=TRUE)
strata1_index_cell2_new<- sample(1:nrow(strata1_cell2_original),
new_cell_sizes2[2], replace=TRUE)
strata1_index_cell3_new<- sample(1:nrow(strata1_cell3_original),
new_cell_sizes2[3], replace=TRUE)
strata1_index_cell4_new<- sample(1:nrow(strata1_cell4_original),
new_cell_sizes2[4], replace=TRUE)

strata1_cell1_new<- strata1_cell1_original[strata1_index_cell1_new,]
strata1_cell2_new<- strata1_cell2_original[strata1_index_cell2_new,]
strata1_cell3_new<- strata1_cell3_original[strata1_index_cell3_new,]
strata1_cell4_new<- strata1_cell4_original[strata1_index_cell4_new,]

## cell sizes for pseudo_data for strata1 (subset of data with
## outcome1=1)
data_strata1_new<- as.data.frame(rbind(strata1_cell1_new,
strata1_cell2_new,
strata1_cell3_new,
strata1_cell4_new))

## apply function f for third and second outcomes within strata of
## first outcome (outcome1=0)
data_strata2<- data[data$outcome1==0, ]
strata2_N_new<- dim(data)[1] - strata1_N_new
new_cell_sizes2<- helper_f(y1=data_strata2$outcome2,
y2=data_strata2$outcome3,
RR1=1,
RR2=target_RR3,
N_num=n_cell3_new,
N_total=strata2_N_new)

strata2_cell1_original<- data_strata2[data_strata2$outcome2==0 &
data_strata2$outcome3==0,]
strata2_cell2_original<- data_strata2[data_strata2$outcome2==1 &
data_strata2$outcome3==0,]
strata2_cell3_original<- data_strata2[data_strata2$outcome2==0 &
data_strata2$outcome3==1,]
strata2_cell4_original<- data_strata2[data_strata2$outcome2==1 &
data_strata2$outcome3==1,]

strata2_index_cell1_new<- sample(1:nrow(strata2_cell1_original),
new_cell_sizes2[1], replace=TRUE)
strata2_index_cell2_new<- sample(1:nrow(strata2_cell2_original),
new_cell_sizes2[2], replace=TRUE)
strata2_index_cell3_new<- sample(1:nrow(strata2_cell3_original),
new_cell_sizes2[3], replace=TRUE)
strata2_index_cell4_new<- sample(1:nrow(strata2_cell4_original),
new_cell_sizes2[4], replace=TRUE)

strata2_cell1_new<- strata2_cell1_original[strata2_index_cell1_new,]
strata2_cell2_new<- strata2_cell2_original[strata2_index_cell2_new,]
strata2_cell3_new<- strata2_cell3_original[strata2_index_cell3_new,]
strata2_cell4_new<- strata2_cell4_original[strata2_index_cell4_new,]
data_strata2_new<- as.data.frame(rbind(strata2_cell1_new,
                                          strata2_cell2_new,
                                          strata2_cell3_new,
                                          strata2_cell4_new))

## combining to pseudo data
data_new<- as.data.frame(rbind(data_strata1_new,
                                data_strata2_new))

} else {
  stop("Invalid number of outcomes specified.")
}

return(data_new)

sample_by_proportion <- function(data, outcomes, cell_prop) {
  cells = enumerate_cells(data, outcomes) #%>% arrange_all(.vars = outcomes)
  if (nrow(cells) != length(cell_prop)) {
    stop("Zero cells prevent calculation")
  }

  n_total = sum(cells$frequency)
  cells$new_frequency = round(n_total * cell_prop)  # or !!! total coutn by frequency?

  filter_by_outcomes <- function(data, outcomes, values) {
    d = data
    for (i in 1:length(outcomes)) {
      d <- d %>% filter(!sym(outcomes[i]) == values[i])
    }
    return(d)
  }

  pieces <- list()
  for (i in 1:nrow(cells)) {
    values = as.numeric(select(cells, starts_with("outcome"))[i,])
    piece <- sample_n(filter_by_outcomes(data, outcomes, values),
                      cells$new_frequency[i], replace=TRUE)
    pieces[[i]] = piece
  }

  data_new = bind_rows(pieces)
  return(data_new)
}

sample_by_inverseodds <- function(data, outcomes, effects){
  linear_pred<- as.matrix(data[,outcomes]) %*% as.matrix(log(effects))
  prob<- (1+exp(-(linear_pred)))^-1
inv_odds <- prob/(1-prob)
treated <- sample(1:nrow(data), nrow(data), prob=inv_odds, replace=TRUE)
data_new <- data[treated,]
return(data_new)
}
cell_counts.R

# enumerate all possibilities of outcomes, with outcome1 varying fastest
# (0, 1, 0, 1...)
enumerate_cells <- function(data, outcomes) {
  if (length(outcomes) == 1) {
    cells = expand.grid(c(0,1))
  } else if (length(outcomes) == 2) {
    cells = expand.grid(c(0,1), c(0, 1))
  } else if (length(outcomes == 3)) {
    cells = expand.grid(c(0,1), c(0, 1), c(0, 1))
  }
  colnames(cells) = outcomes

  cells$frequency = 0
  for (i in 1:nrow(cells)) {
    d = data
    # filter d down by each outcome and value (0 or 1)
    for (j in 1:length(outcomes)) {
      d = d %>% filter(!sym(outcomes[j]) == cells[i,j])
    }
    cells$frequency[i] = nrow(d)
  }
  return(cells)
}

# basis for cell size table
get_cell_sizes <- function(data, outcomes, ...) {
  n_cells <- 2^length(outcomes)
  cells = data.frame(cell_num = 1:n_cells, timestamp=date(), ..., stringsAsFactors=FALSE)

  cells <- bind_cols(cells, enumerate_cells(data, outcomes))
  cells$total_count = sum(cells$frequency)
  cells$proportion = cells$frequency / cells$total_count
  return(cells)
}
do_risk_measurement <- function(data, ...) {
  results = data.frame(timestamp=date(), ..., stringsAsFactors=FALSE)
  results$r1<- mean(filter(data, zz==1)$outcome, na.rm=TRUE)
  results$r0<- mean(filter(data, zz==0)$outcome, na.rm=TRUE)
  results$rd<- results$r1 - results$r0
  results$rr<- ifelse(results$r0>0, (results$r1 / results$r0), NA)
  results$N<- dim(data)[1]
  results$N1<- sum(data$zz)
  results$N0<- results$N - results$N1
  results$Y1<- sum(filter(data, zz==1)$outcome, na.rm=TRUE)
  results$Y0<- sum(filter(data, zz==0)$outcome, na.rm=TRUE)
  results$Yd<- results$Y1 - results$Y0
  return(results)
}

do_rate_measurement <- function(data, ...) {
  results = data.frame(timestamp=date(), ..., stringsAsFactors=FALSE)
  results$r1<- sum(filter(data, zz==1)$outcome, na.rm=TRUE) / sum(filter(data, zz==1)$time_outcome, na.rm=TRUE)
  results$r0<- sum(filter(data, zz==0)$outcome, na.rm=TRUE) / sum(filter(data, zz==0)$time_outcome, na.rm=TRUE)
  results$rd<- results$r1 - results$r0
  results$rr<- ifelse(results$r0>0, (results$r1 / results$r0), NA)
  results$T1<- sum(filter(data, zz==1)$time_outcome, na.rm=TRUE)
  results$T0<- sum(filter(data, zz==0)$time_outcome, na.rm=TRUE)
  results$Y1<- sum(filter(data, zz==1)$outcome, na.rm=TRUE)
  results$Y0<- sum(filter(data, zz==0)$outcome, na.rm=TRUE)
  results$Yd<- results$Y1 - results$Y0
  return(results)
}
do_cost_measurement <- function(data, ...) {
  results = data.frame(timestamp=date(), ..., stringsAsFactors=FALSE)
  results$C1_avg<- mean(subset(data, zz==1)$outcome_cost, na.rm=TRUE)
  #average cost per day in exposed
  results$C0_avg<- mean(subset(data, zz==0)$outcome_cost, na.rm=TRUE)
  #average cost per day in unexposed
  results$cd_avg<- results$C1_avg - results$C0_avg
  #average cost difference (per day)
  results$T1_avg<- mean(subset(data, zz==1)$time_cost, na.rm=TRUE)
  #average follow-up time in exposed
  results$T0_avg<- mean(subset(data, zz==0)$time_cost, na.rm=TRUE)
  #average follow-up time in unexposed
  results$C1_med<- median(subset(data, zz==1)$outcome_cost, na.rm=TRUE)
  #median cost per day in exposed
  results$C0_med<- median(subset(data, zz==0)$outcome_cost, na.rm=TRUE)
  #median cost per day in unexposed
  results$cd_med<- results$C1_med - results$C0_med
  #median cost difference (per day)
  results$T1_med<- median(subset(data, zz==1)$time_cost, na.rm=TRUE)
  #median follow-up time in exposed
  results$T0_med<- median(subset(data, zz==0)$time_cost, na.rm=TRUE)
  #median follow-up time in unexposed
  results$C1_tot<- sum(subset(data, zz==1)$outcome_cost, na.rm=TRUE)
  #total cost per day in exposed
  results$C0_tot<- sum(subset(data, zz==0)$outcome_cost, na.rm=TRUE)
  #total cost per day in unexposed
  results$cd_tot<- results$C1_tot - results$C0_tot
  #total cost difference (per day)
  results$N<- dim(data)[1]
  results$N1<- sum(data$zz)
  results$N0<- results$N - results$N1
  return(results)
}
```r
utils.R

# Plot setup

plain_theme <- theme(
  panel.background = element_rect(color="black", fill="white", size=0),
  panel.border = element_blank(),
  panel.grid.major.y = element_line(color="#eeeeee", size=0.5),
  panel.grid.major.x = element_blank(),
  panel.grid.minor = element_blank(),
  axis.line = element_line(color="black", size=0.75),
  axis.ticks = element_line(color="black", size=0.75),
  # golden ratio -- ~1.61:1 (x:y)
  aspect.ratio = 1 / ((1 + sqrt(5)) / 2)
)

finalize_plot <- function(plot, title) {
  plot_file = tempfile(pattern="plot_", fileext=".png")
  ggsave(plot_file, plot, device=png())
  l = list("title" = title,
            "plotFile" = plot_file,
            "plotData" = paste0("data:image/png;base64,",
            base64encode(plot_file)))
  return(l)
}

finalize_table <- function(table, title) {
  html = "<html></html>" #print(kable(table, "html"))
  l = list("title" = title,
            "data" = table,
            "html" = html)
  return(l)
}

empty_to_null <- function(x) {
  if (length(x) == 0) { return(NULL) } # check this??!
  if (length(x) > 1) { return(x) }
  if (is.null(x)) { return(NULL); }
  if (is.na(x)) { return(NA); }
  if (x == "") { return(NULL); }
  return(x)
}
```