`cd_new()`: Statistics and p-values

2022-04-22

Contents

1 Overview

- This .Rmd document is intended to be run on your PC and describes the commands in the `cd_new.R` script that defines an R function `cd_new()`. `cd_new()` is a revision of Christina's `compute_distributions()` that is more computationally efficient for running simulations with multiple assumed carrier probabilities of the cRV.
- `cd_new()` is a long function, and the description of its components is split across several code chunks in this document. Knitting the document will fail because if a code chunk starts a function but doesn't finish it, RStudio will exit with an "unexpected end of input" error.
- To avoid this error, the code chunk at the top of this RMarkdown document sets `eval=FALSE`.

- The end of the document has a `purl()` command that you must manually execute on your PC (i.e., cut-and-paste into the R console) to generate the R script `cd_new.R`.
- After generating `cd_new.R` on your PC, you can source it into your R session with `source("cd_new.R")` to get a copy of the `cd_new()` function into your PC workspace if you like.
- You need to port `cd_new.R` to the Compute Canada cluster so that scripts such as `simalt.R` and `simnull.R` can find it when running the simulations.

2 Introduction

- Christina's project was motivated by a genetic study of lymphoid cancer families at the BC Cancer Agency. In these types of studies, investigators collect families with multiple affected members (so-called **multiplex** families) to increase the chances that a disease-predisposing variant is being transmitted

(or segregating) in them. They assume that any disease-predisposing variants are rare and look for rare variants (RVs) shared amongst the affected family members after sequencing their exomes. The assumption that disease-predisposing variants are rare is justified by the rareness of multiplex families in the population.

- The investigators showed Christina a family containing (at the time) five members affected with multiple subtypes of lymphoid cancer (one of these family members was subsequently found to be in error and was removed). They believed some of these subtypes were more genetically determined than others. Subtypes that are less genetically determined are more likely to include "sporadic" or non-genetic cases. Christina started working on methods to downweight the statistical evidence from the more sporadic subtypes when prioritizing RVs for further investigation.

- We looked to the work of Alexandre Bureau who subsequently joined the project as Christina's co-supervisor. He and colleagues had developed new methods for *testing* rare sequencing variants in multiplex families which we thought might guide our thinking about our related problem of *prioritizing* or ranking variants in some statistically rigourous way. However, their testing methods do not accommodate multiple subtypes and do not model transmission probabilities. Christina and I were interested in extending *transmission* methods, such as the classic transmission-disequilibrium test in parent-child trios, to *model* transmission probabilities in extended families. Modelling transmission probabilities allows for multiple subtypes and other factors of the offspring or variant to be incorporated into the prioritization. Unfortunately, family studies of RVs have limited statistical precision because genetic transmissions from parent to offspring (i.e. the meioses or segregations) are relatively few in number. Often only a single family carries a given RV, and might provide a maximum of, say, 10 transmissions in total from which to estimate its different transmission probabilities. To retain a reasonable number of transmissions per subtype, Christina modelled two subtypes only.

- Two approaches may be taken to model transmissions. The most general is to consider all families in the study, even those which do not segregate the RV of interest. We refer to this as the *global* approach. The global approach requires specifying the probability that a pedigree founder carries the RV, a quantity that may not be known well. The second approach considers only the families which segregate the RV of interest. By conditioning on the families that carry the RV, we eliminate the need to specify the RV carrier probability for pedigree founders. We refer to this second approach as a *local* approach. We refer to an RV configuration that includes all families in the study as a *global* configuration and an RV configuration that includes only the families carrying the RV of interest as a *local* configuration.

- Christina's function `compute_distributions()` takes as input a set of pedigrees in the study, and a given value of the carrier probability p_c assumed to be known (and approximately twice the cumulative cRV frequency in the population). The function enumerates all possible RV configurations of affected individuals in the study and for each such *global configuration* returns some test statistics, assuming that the causal RV (cRV) enters a pedigree through at most a single founder. The five test statistics returned by the function are:

  (1) a *global likelihood ratio* or LR (CN thesis eqn 4.7, page 37) based on the conditional probability of the global configuration (i.e. the configuration for all the families in the study) given at least one affected individual in the study carries the RV,
  (2) a transmission statistic (CN thesis eqn 4.11, page 38) which is approximately proportional to the global LR statistic,
  (3) a local LR (CN thesis eqn 4.15, page 40) based on the conditional probability of the global configuration given the families that carry the RV,
  (4) the RVS statistic, based on the inverse of the conditional probability of the global configuration, given the families that carry the RV, and on the total number of affected individuals sharing the RV (see the displayed eqn at bottom of page 40 of CN's thesis), and
  (5) the modified RVS statistic, similar to the RVS statistic but based on the number of affected individuals who carry the genetically compelling subtype (see the displayed eqn at top of page 41 of the thesis).

- For each global configuration, the function also returns the p-values for each test statistic; i.e. the

chance of a value of the test statistic as or more extreme than that for the given configuration.

- The output of `compute_distributions()` is a lookup table of test-statistic values and p-values for a global configuration of interest.
  - Looking up test-statistic values is useful when ranking variants and looking up p-values is useful when estimating type 1 error rates and power.
- However, for simulations `compute_distributions()` is wasteful.
  - It returns statistics and p-values for all five methods given a *single* value of the carrier probability. However, in our simulations we want to compare methods for *multiple* carrier probabilities (the true value used to simulate the data and several misspecified values)
  - The local approaches don't depend on the carrier probability, so multiple calls to `compute_distributions()` will wastefully return the same local statistics multiple times.
- In the sections that follow we develop an alternative function, `cd_new()`, that takes a vector of carrier probabilities as input and returns global statistics and p-values under each carrier probability. All code in `cd_new()` is Christina's except where otherwise noted.
- For development and testing we use a single study of 2 pedigrees that are included in Christina's `RVMethods` package, available on GitHub.

3 Workflow of `cd_new()`

- The key steps in `cd_new()` are
  1. Define the function and its arguments
  2. Compute conditional *familial* configuration probabilities, given that the RV was introduced to the family through one founder,
  3. Using the conditional familial configuration probabilities and the vector of carrier probabilities as input, calculate the test statistics. These test statistics are: (i) global LR statistics for each value of the carrier probability, (ii) the transmission statistic, (iii) the local LR statistic, (iv) the RVS statistic and (v) the modified RVS statistic. Notes:
     - Statistic values are calculated for each *global* configuration.
     - In addition to the statistics, we calculate the probability of each global configuration under the null hypothesis that the two transmission probabilities are both 1/2, because these probabilities are needed for computing p-values.
  4. Calculate p-values for all statistics.
  5. Return look-up tables of statistics and p-values.

3.1 Define the function and its arguments

```
cd_new <-  function(peds, subtypes,carrier_probs,
                    tau_increment=0.05,subtype_weights=NULL,useK=FALSE){
```

The function arguments are as follows:

- `peds`: is a `ped` object containing the $m \geq 2$ pedigrees that make up the study.
- `subtypes`: is a character vector of length two that contains labels for the two subtypes found in the pedigrees of `peds`. The label of the more genetically-compelling subtype must be listed first.
- `carrier_probs`: is a vector of carrier probabilities. Assuming the probability of *any* cRV is rare, each carrier prob is essentially twice the cumulative probability of all cRVs.
- `tau_increment`: is the spacing of the grid of transmission probabilities (taus) to search for MLEs.
- `subtype_weights`: is a numeric vector of length two for the prior weights on founders. This doesn't appear to be used. Christina writes that "When assuming an informative prior for the founders, these are the weights of the individuals with [disease] subtypes A and B, respectively. By default, subtype_weights = NULL so that no founder weights are applied (i.e. flat founder prior)."

- `useK`: indicates whether or not to consider the number of carriers in a configuration (K) when computing p-values. In general, p-values are sums of probabilities over configurations with statistic values as or more extreme than the observed configuration. If `useK=TRUE` we restrict the sum to configurations with at least as many carriers as in the observed configuration. Default is `FALSE` (do not restrict).

3.2 Compute conditional probabilities of familial configurations

- For a given carrier probability, p_c, all of the likelihoods and RV sharing probabilities we consider can be expressed in terms of familial configuration probabilities $P(\overrightarrow{C}_i; \tau, p_c)$ for a configuration $\overrightarrow{C}_i$ of affecteds in pedigree i and value $\tau = (\tau_a, \tau_b)$ of the transmission parameters.
  - See, for example, equation (4.2) on page 32 of CN's thesis
- The familial configuration probabilities, in turn, are a function of conditional familial configuration probabilities $P(\overrightarrow{C}_i; \tau | H_i = 1)$. The conditioning event $\{H_i = 1\}$ is that the RV was introduced into the ith pedigree through one founder.
- Details of the relationship between unconditional and conditional configuration probabilities are in Appendix A.1 of this document.

3.2.1 Compute configuration probabilities over a grid of values

- The first major step is to set up a grid of τ's and then loop over the pedigrees, i, in the study, calculating $P(\overrightarrow{C}_i; \tau | H_i = 1)$ for each possible configuration, C_i, and τ on the grid. The probabilities are stored in an object called `fam_likeGrids`, which is a list with elements for each family in the study. The list elements are themselves lists with the following components:
  - `like_mat`, the matrix of probabilities with rows for different τ's and columns for different familial configurations,
  - `configs`, a matrix of logicals with rows for familial configurations and columns for the affecteds in the pedigree, and
  - `binIDs`, a vector of length equal to the number of rows of `configs` containing the base-10 representations of each configuration. For example, a configuration with carrier status `TRUE FALSE TRUE`, encoded as `1 0 1`, would be represented by $1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 5$ in the vector.
- Notes:
  - Unconditional *versus* conditional probabilities: Christina's worker function `test_allcombos_withZero()` calculates unconditional configuration probabilities for a given pedigree. Using a working value of p_c, this function returns unconditional familial configuration probabilities in a matrix with rows corresponding to τ's and columns corresponding to configurations.
  - Christina's `compute_distributions()` function stores unconditional familial probabilities which depend on p_c. By contrast, `cd_new()` stores conditional familial probabilities (given that the RV was introduced into the family by one founder) that do not depend on p_c. - To support `cd_new()` we have written a new function called `uncond2cond()`, defined in the Appendix of this document, to convert the unconditional probabilities of the familial configurations (that depend on p_c) to conditional probabilities given that the RV was introduced into the family by one founder (that don't depend on p_c).
  - Connection with the `gRain` package: `test_allcombos_withZero()` calls a worker function `compute_sharingProb()` to calculate unconditional probabilities for specific configurations and τ's. `compute_sharingProb()` uses functions from the `gRain` package to represent the pedigree as a Bayesian network and calculate sharing probabilities on this network.
  - `withZero`: The "withZero" part of Christina's function name `test_allcombos_withZero` refers to the unconditional probability of the all-zero (i.e. all non-carrier) configuration being returned, as required for calculating the probability of the conditioning event in the global probabilities (i.e. the event that at least one of the sequenced affected individuals in the study carries the RV).
  - Coding note: We access internal functions from `RVMethods` with the `:::` operator.

```
tau_grid <- RVMethods:::make_tauGrid(increment_width = tau_increment, constrained = TRUE)
study_FamIDs <- unique(peds$FamID)
```

```
famIndex <- c(1:length(study_FamIDs)) # indices of family IDs
fam_likeGrids <- list() # empty list to hold grids of config probs for each family
for (i in 1:length(study_FamIDs)){ # loop over families
  ped <- peds[peds$FamID == study_FamIDs[[i]], ]
  # Unlike compute_distributions which has one carrier_prob, we have a vector.
  # Use the first carrier_probs value to compute unconditional familial probs.
  fam_likeGrids[[i]] <- RVMethods:::test_allcombos_withZero(ped, subtypes, tau_grid,
    carrier_probs[1],subtype_weights)
  # Now use our new uncond2cond() function to convert the unconditional
  # familial probs to conditional ones that don't depend on the carrier prob.
  fam_likeGrids[[i]] <- uncond2cond(fam_likeGrids[[i]],ped,carrier_probs[1])
}
```

3.2.2 Remove invalid configurations

- Configurations that can only happen if the RV is introduced through multiple founders are considered invalid and are removed to save space.
- The function `remove_invalidConfigs()` does this for a single pedigree. We use `lapply()` to call it on all pedigrees in `fam_likeGrids`.

```
fam_likeGrids <- lapply(fam_likeGrids, function(x){
  RVMethods:::remove_invalidConfigs(x)
})
```

3.3 Compute statistics and null probabilities for global configurations

- At this stage of her workflow, Christina uses the familial configuration probabilities calculated in the previous section to construct the global and local test statistics as well as the null probabilities for each global configuration.
- Christina's functions `condition_globalDist_zeroConfig()`, `approx_globalDist()` and `conditioned_semiGlobalDist()` calculate, respectively, the global likelihood ratio (LR), transmission and local LR statistics for each global configuration. Their output is a matrix with rows for global configurations and columns that include the base-10 representation of the binary familial configuration or `binID`, the LR statistics and the null probabilities for each global configuration.
- The global LR statistic (CN thesis eqn 4.7, page 37) is based on the conditional probability of the global configuration (for all families in the study) given that at least one affected individual in the study carries the cRV.
- The transmission statistic (CN thesis eqn 4.11, page 38) is a LR based on an approximate unconditional probability of the global configuration. Notes:
  - On pages 38-39 of her thesis, Christina shows that the transmission statistic is approximately proportional to the global LR statistic. We should therefore expect similar inference from the two approaches.
  - In equation (4.10) on page 37 of her thesis, Christina derives the transmission statistic as a likelihood ratio from (an approximation to) the unconditional global probability.
  - Though the transmission statistic does not depend on the carrier probability, the null probabilities of each global configuration do. As a result, the **p-values** of the transmission statistic will depend on the assumed value of the carrier probability.
- The local LR (CN thesis eqn 4.15, page 40) is a LR based on the conditional probability of the global configuration given the families that carry the RV.

3.3.1 Dealing with several carrier probabilities

- As previously mentioned, Christina's `compute_distributions()` assumed only one value of the carrier probability.

- In `cd_new()` we instead loop over a supplied vector of carrier probabilities.
- For each carrier probability, we create an object `flg` listing the unconditional probabilities of familial configurations based on the current value of the carrier probability.
- The unconditional probabilities of familial configurations in `flg` are created from Christina's `fam_likeGrid`, a list that contains conditional probabilities of the familial configurations given that the RV was introduced by one founder. The conditional probabilities in `fam_likeGrid` do not depend on the carrier probability.
- We use our new utility function `cond2uncond()`, defined in the Appendix of this document, to convert the conditional probabilities in `fam_likeGrid` to the unconditional probabilities for the current value of the carrier probability in `flg`.
- We then pass `flg` to Christina's functions `condition_globalDist_zeroConfig` and `approx_globalDist` to calculate the global statistics and the global configuration probabilities under the null hypothesis.
- The calculation of the local statistics and probabilities under the null hypothesis is the same as Christina's. These calculations are implemented in her function `conditioned_semiGlobalDist()` and are called at the end of the following code chunk.

```
D1_global_sharing_byBinID <- list() # list for global LR, will be one list item per p_c
D2_global_sharing_byBinID <- list() # list for global transm,  ditto
flg <- list()
for(i in 1:length(carrier_probs)){
  for(j in 1:length(fam_likeGrids)){
    ped <- peds[peds$FamID == study_FamIDs[[j]], ]
    # Need to convert conditional probs in fam_likeGrids to
    # unconditional using current value of carrier_probs before
    # passing to functions that calculate global statistics.
    flg[[j]] <- cond2uncond(fam_likeGrids[[j]],ped,carrier_probs[i])
  }
  D1_global_sharing_byBinID[[i]] <-
    RVMethods:::condition_globalDist_zeroConfig(likeGrids_byFam = flg,
                                                famID_index = famIndex,
                                                tau_grid)
  D2_global_sharing_byBinID[[i]] <-
    RVMethods:::approx_globalDist(likeGrids_byFam = flg,
                                  famID_index = famIndex,
                                  tau_grid)

}
# local approaches don't depend on p_c, so it doesn't matter
# what value of the carrier probability was used to get the
# unconditional familial probs. Can use last val of flg from
# the for loop above.
D3_global_sharing_byBinID <-
  RVMethods:::conditioned_semiGlobalDist(likeGrids_byFam = flg,
                                         famID_index = famIndex,
                                         tau_grid)
```

3.4 Calculate p-values

- In this section, we calculate p-values and combine them with the global configurations and associated information, such as the MLE of the τ's, the number K of affecteds who are carriers in the study, the statistic values (LR, RVS or modified RVS) and the null probabilities.
- Christina's matrix `fam_configs` holds the global configurations that she later combines with the p-values, statistics, etc. The rows of `fam_configs` are global configurations and the columns are the affected individuals in the study. The affected study members are labelled by family and subject ID as

familyID:subjectID.

```
fam_configs <- list()
# Find configurations for each family:
for(i in famIndex){
  fam_configs[[i]] <- 1*fam_likeGrids[[i]]$configs[match(D3_global_sharing_byBinID[, i],
                                                         fam_likeGrids[[i]]$binIDs), ]
  # label columns familyID:subjectID
  colnames(fam_configs[[i]]) <- paste0(study_FamIDs[[i]], ":", colnames(fam_configs[[i]]))
}
# fam_configs is now a list of familial configs for each pedigree in the study.
# Combine them (i.e., cbind() them) into one matrix. The rows of this matrix
# are the global configs and the columns are the affecteds in the study.
fam_configs <- do.call(cbind, fam_configs)
```

- We now calculate the p-values.
- The following code is essentially Christina's *except* that for the global approaches we loop over multiple values of the carrier probability.
  - For a given matrix of configuration-specific statistics and null probabilities, Christina uses `sapply()` to calculate the p-values for every configuration.
  - For a given target configuration, the p-value is the sum of configuration-specific probabilities over configurations with statistic values as or more extreme than the target configuration. If `useK=TRUE` we restrict the sum to configurations with at least as many carriers as in the target configuration.

```
# Global LR: collect LR stats, null probs and calculate p-values.
# First initialize w/ fam configs, taus and K.
global_dist <- cbind(fam_configs,
                     D1_global_sharing_byBinID[[1]][, c("tau_A","tau_B")],
                     K=apply(fam_configs, 1, sum))
for(i in 1:length(carrier_probs)) {
  # add LR and null_configProb cols from ith D1_global_sharing_byBinID matrix
  global_dist <- cbind(global_dist,
                       D1_global_sharing_byBinID[[i]][,c("LR","null_configProb")])
  LR <- paste0("LR",i); nullconf <- paste0("null_configProb",i)
  names(global_dist)[ncol(global_dist) - (1:0)] <- c(LR,nullconf)
  # Calculate p-values and add to global_dist
  if(useK){
    global_dist <- cbind(global_dist,sapply(1:nrow(global_dist), function(x){
      sum(global_dist[global_dist[,LR] >= global_dist[x,LR]
                      & global_dist$K >= global_dist$K[x],nullconf],
          na.rm = TRUE)}))
  } else {
    global_dist <- cbind(global_dist,sapply(1:nrow(global_dist), function(x){
      sum(global_dist[global_dist[,LR] >= global_dist[x,LR],nullconf],
          na.rm = TRUE)}))

  }
  # add index i to LR_pvalue column name
  names(global_dist)[ncol(global_dist)] <- paste0("LR_pvalue",i)
}

# Transmission stat: collect LR stats, null probs and calculate p-values.
# Initialize w/ fam configs, LR (same for all carrier probs), taus and K
semiglobal_dist <- cbind(fam_configs,
                     D2_global_sharing_byBinID[[1]][, c("LR","tau_A","tau_B")],
```

```
                        K=apply(fam_configs, 1, sum))
  for(i in 1:length(carrier_probs)) {
    # add null_configProb col from ith D2_global_sharing_byBinID matrix
    semiglobal_dist <- cbind(semiglobal_dist,
                             D2_global_sharing_byBinID[[i]][,"null_configProb"])
    nullconf <- paste0("null_configProb",i)
    names(semiglobal_dist)[ncol(semiglobal_dist)] <- nullconf
    # Calculate p-values
    if(useK){
      semiglobal_dist$LR_pvalue <- sapply(1:nrow(semiglobal_dist), function(x){
        sum(semiglobal_dist[semiglobal_dist[,"LR"] >= semiglobal_dist[x,"LR"]
                            & semiglobal_dist$K >= semiglobal_dist$K[x],nullconf],
            na.rm = TRUE)})
    } else {
       semiglobal_dist$LR_pvalue <- sapply(1:nrow(semiglobal_dist), function(x){
        sum(semiglobal_dist[semiglobal_dist[,"LR"] >= semiglobal_dist[x,"LR"],nullconf],
            na.rm = TRUE)})

    }
    names(semiglobal_dist)[ncol(semiglobal_dist)] <- paste0("LR_pvalue",i)
  }


  # Local approaches: collect LR stats, null probs and calculate p-values.
  # The following is CN's code
  condsemiglobal_dist <- cbind(fam_configs,
                               D3_global_sharing_byBinID[, -c(1:length(study_FamIDs))])

  condsemiglobal_dist$K = apply(fam_configs, 1, sum)

  #create a list of names of individuals with the genetically compelling subtype
  GCsub_list <- apply(peds[which(peds$available & peds$subtype == subtypes[[1]]), c("FamID", "ID")],
                      1, function(x){paste0(x, collapse = ":")})

  condsemiglobal_dist$K_sub <- sapply(1:nrow(fam_configs), function(x){
    sum(colnames(fam_configs)[fam_configs[x, ] == 1] %in% GCsub_list)
  })

#  print(paste0("Local P-value: ", Sys.time()))
  if(useK){
    #calculate the p-value for the likelihood ratio statistic
    condsemiglobal_dist$LR_pvalue <-
      sapply(1:nrow(condsemiglobal_dist),
      function(x){
      sum(condsemiglobal_dist$null_configProb[condsemiglobal_dist$LR >=
                                                condsemiglobal_dist$LR[x]
                                              & condsemiglobal_dist$K >=
                                                condsemiglobal_dist$K[x]
                                              & condsemiglobal_dist$distID ==
                                                condsemiglobal_dist$distID[x]],
          na.rm = TRUE)
    })
  } else {
```

```
    condsemiglobal_dist$LR_pvalue <-
      sapply(1:nrow(condsemiglobal_dist),
             function(x){
      sum(condsemiglobal_dist$null_configProb[condsemiglobal_dist$LR >=
                                                condsemiglobal_dist$LR[x]
                                              & condsemiglobal_dist$distID ==
                                                condsemiglobal_dist$distID[x]],
          na.rm = TRUE)
    })
  }


#  print(paste0("RVS-Based P-value: ", Sys.time()))
  condsemiglobal_dist$RVS_pvalue <-
    sapply(1:nrow(fam_configs), function(x){
    sum(condsemiglobal_dist$null_configProb[condsemiglobal_dist$null_configProb <=
                                              condsemiglobal_dist$null_configProb[x]
                                    & condsemiglobal_dist$K >=
                                      condsemiglobal_dist$K[x]
                                    & condsemiglobal_dist$distID ==
                                      condsemiglobal_dist$distID[x]],
        na.rm = TRUE)
  })

  condsemiglobal_dist$modRVS_pvalue <-
    sapply(1:nrow(fam_configs),
           function(x){
    sum(condsemiglobal_dist$null_configProb[condsemiglobal_dist$null_configProb <=
                                              condsemiglobal_dist$null_configProb[x]
                                    & condsemiglobal_dist$K >=
                                      condsemiglobal_dist$K[x]
                                    & condsemiglobal_dist$K_sub >=
                                      condsemiglobal_dist$K_sub[x]
                                    & condsemiglobal_dist$distID ==
                                      condsemiglobal_dist$distID[x]],
        na.rm = TRUE)
  })


  global_dist$binID <-
    apply(global_dist[, 1:ncol(fam_configs)], 1,
          function(x){
    base::strtoi(paste0(x, collapse = ""), base = 2)
  })

  semiglobal_dist$binID <-
    apply(semiglobal_dist[, 1:ncol(fam_configs)], 1,
          function(x){
    base::strtoi(paste0(x, collapse = ""), base = 2)
  })

  condsemiglobal_dist$binID <-
    apply(condsemiglobal_dist[, 1:ncol(fam_configs)], 1, function(x){
```

```
    base::strtoi(paste0(x, collapse = ""), base = 2)
  })
```

3.5 Return look-up tables

- Now we can return the look-up tables.
- The format of the output object for these look-up tables is different from `compute_distributions()`, which returns data frames of results for (i) the global LR, (ii) transmission and (iii) the local methods. The `compute_distributions()` data frames include the statistic values and p-values we need, but also other information about the configuration that we don't need for our simulation study (e.g. the corresponding MLEs of the two transmission probabilities, the number of affected individuals who carry the RV for the configuration, and separate numerator and denominator of each of the three LR statistics). For us, splitting the results across three data frames is awkward.
- To streamline, we instead return a list with two matrices. The first matrix is a table of statistic values with rows for each global configuration and columns for each method. The second is a table of p-values for each global configuration (rows) and method (columns). In both matrices we use the compact base10 representation of the binary configuration vector, `binID`, to identify the global configurations.
- We also return a "key" that tells us the order of affected individuals in the configurations. This allows us to translate `binID`s to configurations. The key is the column names of the `fam_configs` matrix.

```
  global_stats <- paste0("LR",1:length(carrier_probs))
  global_pvals <- paste0("LR_pvalue",1:length(carrier_probs))
    statvals <- cbind(global_dist$binID, # same ID in all three d.f.s
                      global_dist[,global_stats],
                      semiglobal_dist[,"LR"], # same for all carrier probs
                      condsemiglobal_dist[,"LR"],
                      1/condsemiglobal_dist[,"RVS_pvalue"], # inverse of prob
                      1/condsemiglobal_dist[,"modRVS_pvalue"]) # inverse of prob
  names(statvals) <- c("binID",
                       paste0("globalLR",1:length(carrier_probs)),
                      "globaltransLR","localLR","RVS","modRVS")
  pvals <- cbind(global_dist$binID, # same in all three d.f.s
                 global_dist[,global_pvals],
                 semiglobal_dist[,global_pvals],
                 condsemiglobal_dist[,"LR_pvalue"],
                 condsemiglobal_dist[,"RVS_pvalue"],
                 condsemiglobal_dist[,"modRVS_pvalue"])
  names(pvals) <- c("binID",
                      paste0("globalLR",1:length(carrier_probs)),
                      paste0("globaltrans",1:length(carrier_probs)),
                      "localLR","RVS","modRVS")
  # More convenient for extracting rows to have the lookup tables
  # be matrices rather than data frames.
  output <- list(statvals=as.matrix(statvals), pvals = as.matrix(pvals),
                 binKey = colnames(fam_configs))
```

```
  return(output)
}
```

3.6 Install R packages and test `cd_new()`

- We require the `SimRVPedigree`, `SimRVSequences` and `RVMethods`, which we can install from Github. `RVMethods` depends on the `gRain` package, which in turn depends on a Bioconductor package called `RBGL`.

- We load an example dataset from a study involving two families and the `SimRVPedigree` package for its `ped` data structure.
- Finally, we provide a call to function `cd_new()` for testing purposes.
- The code in the chunk below is all commented out. Remove the comments for testing.

```
# install.packages("devtools")
# library(devtools)
# install_github("https://github.com/simrvprojects/SimRVPedigree")
# install_github("https://github.com/simrvprojects/SimRVSequences")
# install.packages("BiocManager")
# BiocManager::install("RBGL")
# install.packages("gRain")
# install_github("https://github.com/simrvprojects/RVMethods")
# library(RVMethods)
# data(study_pedigrees)
# library(SimRVPedigree)
# # Set up for a function call you can use for testing cd_new()
# peds = study_pedigrees[study_pedigrees$FamID %in% c(58, 304), ]
# subtypes = c("HL", "NHL")
# carrier_prob = 0.00032 # True carrier prob
# carrier_probs = carrier_prob*c(1/10,1/2,1,2,10)
# # Call cd_new()
# cd_new(peds,subtypes,carrier_probs) #leave remaining arguments at their defaults
```

3.7 Generate the R script

- To create the R script `cd_new.R` to port to the Compute Canada cluster for running simulations, cut-and-paste the following into the R command line on your PC:

```
knitr:: purl(input="cd_new2.Rmd",output="cd_new2.R")
```

- This command will return a file `cd_new.R` that includes every code chunk in this file `cd_new.Rmd`, including a code chunk at the very top of the file that sets `knitr` options.
- Delete the code chunk at the top of `cd_new.R` that sets the `knitr` options and the code chunk on line 252 of `cd_new.R` that contains the `knitr::purl()` command.
- If you like, you may then `source("cd_new.R")` into the R session on your PC to define the `cd_new()` function there.
- You should port `cd_new.R` over to the Compute Canada cluster for use by the simulation scripts `simalt.R` and `simnull.R`.

A Appendix

A.1 Converting conditional familial configuration probabilities to unconditional

- A study consists of m pedigrees indexed $i = 1, ..., m$.
- Given a value p_c of the carrier probability, Christina's function `test_allcombos_withZero()` returns the familial configuration probabilities $P(\overrightarrow{C}_i; \tau, p_c)$. Specifically, for each pedigree $i = 1, ..., m$, the function returns a matrix of probabilities with rows corresponding to τ's and columns corresponding to configurations. Familial configuration probabilities are calculated as (see bottom of page 41 of thesis):

$$P(\overrightarrow{C}_i; \tau, p_c) = \begin{cases} r_{f_i} P(\overrightarrow{C}_i; \tau | H_i = 1) & \text{if } \overrightarrow{C}_i \neq 0 \\ r_{f_i} P(\overrightarrow{C}_i; \tau | H_i = 1) + (1 - r_{f_i}) & \text{if } \overrightarrow{C}_i = 0, \end{cases}$$

where $r_{f_i} = n_{f_i} p_c$ is the "founder introduction probability" for a pedigree with n_{f_i} founders and H_i is the number of founders that carry the RV, assumed to be either 0 or 1. We call $P(\overrightarrow{C}_i; \tau, p_c)$ the

unconditional familial configuration probability and $P(\overrightarrow{C}_i;\tau|H_i=1)$ the conditional (on the RV being introduced through one founder) familial configuration probability.

- We see that the unconditional probabilities are functions of the carrier probability. To enable recycling of computations across different carrier probability values, we convert these to conditional probabilities, which do not depend on the carrier probability. When needed, we can convert the conditional probabilities to unconditional probabilities for a given value of the carrier probability.
- We wrote the following two new R functions to convert matrices of unconditional familial configuration probabilities to conditional and *vice versa*, based on the relationship between conditional and unconditional probabilities shown in the displayed equation above.

```
uncond2cond <- function(fam_likeGrid,ped,carrier_prob) {
  like_mat <- fam_likeGrid$like_mat
  num_found = length(unique(ped$ID[which(is.na(ped$dadID) & is.na(ped$momID))]))
  fint_prob = num_found*carrier_prob #founder intro prob, using first carrier prob
  like_mat[,1:(ncol(like_mat)-1)] <-
      like_mat[,1:(ncol(like_mat)-1)] / fint_prob #non-zero configs
  like_mat[,ncol(like_mat)] <-
      (like_mat[,ncol(like_mat)] - (1-fint_prob))/fint_prob # zero config
  fam_likeGrid$like_mat <- like_mat
  return(fam_likeGrid)
}
# Also do the reverse function
cond2uncond <- function(fam_likeGrid,ped,carrier_prob) {
  like_mat <- fam_likeGrid$like_mat
  num_found = length(unique(ped$ID[which(is.na(ped$dadID) & is.na(ped$momID))]))
  fint_prob = num_found*carrier_prob #founder intro prob, using first carrier prob
  like_mat[,1:(ncol(like_mat)-1)] <-
      fint_prob*like_mat[,1:(ncol(like_mat)-1)] # non-zero configs
  like_mat[,ncol(like_mat)] <-
      fint_prob*like_mat[,ncol(like_mat)] + (1-fint_prob) # zero config
  fam_likeGrid$like_mat <- like_mat
  return(fam_likeGrid)
}
```