DEVCOM DAC-TN-2022-020
November 2022

The Human–Systems Integration (HSI) Metric Tradespace Exploration Environment (HMTee) Technical Documentation

by Christopher Garneau

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so specified by other official documentation.

WARNING

Information and data contained in this document are based on the input available at the time of preparation.

TRADE NAMES

The use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial hardware or software. The report may not be cited for purposes of advertisement.



DEVCOM DAC-TN-2022-020
November 2022

The Human–Systems Integration (HSI) Metric Tradespace Exploration Environment (HMTee) Technical Documentation

by Christopher Garneau

DEVCOM Analysis Center

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED (From - To)
November 2022	Technical Note	10/01/2021 – 09/30/2022

4. TITLE AND SUBTITLE
The Human–Systems Integration (HSI) Metric Tradespace Exploration Environment (HMTee) Technical Documentation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Christopher Garneau

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Director
DEVCOM Analysis Center
6896 Mauchly Street
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER

DEVCOM DAC-TN-2022-020

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The Human–Systems Integration (HSI) Metric Tradespace Exploration Environment (HMTee) is a convenient R Shiny application for human factors and HSI analysis problems. This report is intended to serve as technical documentation for developers or others with technical expertise to learn about the implementation of HMTee and integrate other software or HSI models with HMTee functionality. It provides an overview of the technical approach, documents the HMTee Application Programming Interface, and provides an exemplar model for analyzing required lifting strength via the Revised NIOSH (National Institute for Occupational Safety and Health) Lifting Equation to illustrate the process of adding new analytical capabilities to HMTee.

15. SUBJECT TERMS
Human–Systems Integration (HSI), Analysis Tools, Modeling and Simulation (M&S), Human Factors Engineering (HFE), Application Programming Interface (API)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Christopher Garneau
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	36	19b. TELEPHONE NUMBER (include area code) 410-278-5814

Table of Contents

List of Figures v
Acknowledgements vi

1. INTRODUCTION 1

2. HMTEE IMPLEMENTATION 2
   - 2.1. Language and Platform: R and R Shiny 2
   - 2.2. Dependencies 2
   - 2.3. Distribution 3
   - 2.4. Interaction Model Integration 3
   - 2.5. Requirements of Models to Be Integrated into HMTee 4
   - 2.6. Optional Capabilities of Models 4
   - 2.7. Customizable HMTee Elements 4

3. HMTEE APPLICATION PROGRAMMING INTERFACE 5
   - 3.1. modelName 5
   - 3.2. library 5
   - 3.3. humanParameters 6
   - 3.4. excludeParameters 6
   - 3.5. filters 6
     - 3.5.1. parameter 6
     - 3.5.2. comparator 7
     - 3.5.3. values 7
   - 3.6. mappings 7
     - 3.6.1. name 7
     - 3.6.2. inputType 8
     - 3.6.3. minVal 8
     - 3.6.4. maxVal 8
   - 3.7. supportedGraphTypes 8
   - 3.8. generatedGraphs 8
     - 3.8.1. name 9
     - 3.8.2. filepath 9
   - 3.9. graphData 9
   - 3.10. model 9
   - 3.11. modelUnits 10
   - 3.12. modelOptions 10
     - 3.12.1. name 10
     - 3.12.2. parameter 10
     - 3.12.3. choices 11

4. EXEMPLAR HMTEE INTERACTION MODEL 12
   - 4.1. Problem Definition 12
   - 4.2. Model Formulation 13
     - 4.2.1. Implementation as R Function(s) 13
     - 4.2.2. HMTee Interaction Model Function 14

Table of Contents

4.2.3. HMTee Interaction Model Configuration 16
4.3. Results 19

5. CONCLUSION 24

6. REFERENCES 25

List of Acronyms 26

Distribution List 28

List of Figures

Figure 1. Interaction model selection 19
Figure 2. System Components pane (left) and Add System Component modal (right) 20
Figure 3. Human Parameters pane (left) and Add Filter modal (right) 21
Figure 4. Model Options modal 22
Figure 5. Interactions/results output pane 23

Acknowledgements

DAC funded this report and development of the Human–Systems Integration (HSI) Metric Tradespace Exploration Environment (HMTee) as part of its Innovation Program (FY22 cohort). The DAC Innovation Program was established to harness the passion of the workforce, enhance collaboration across divisions, and redefine success. Project leads are allotted up to 20% of their time to work on their projects.

The author would like to thank the Innovation Program panel and the mentors for this project (Dr Kathryn Loftis and Ms Debra Patton) for their support. The author also acknowledges the helpful feedback from the DAC Data Analytics Community of Practice regarding implementing and distributing R Shiny applications.

1. INTRODUCTION

The Human–Systems Integration (HSI) Metric Tradespace Exploration Environment (HMTee) is a convenient R Shiny application for human factors and HSI analysis problems. It constitutes a digital reference for identifying quantitative guidelines and metrics for relevant HSI factors, enables practitioners to visualize results of an analysis, and produces visual evidence of performance simulations for decisionmakers. It permits an analyst, researcher, or practitioner to explore and document assumptions, parameters, and performance metrics, and subsequently perform tradeoffs among competing factors.

This report is intended to serve as technical documentation for developers or others with technical expertise to learn about the implementation of HMTee and integrate other software or HSI models with HMTee functionality. This guide includes an overview of the technical approach, a description of components required of models that integrate with HMTee (i.e., the HMTee Application Programming Interface [API]), and documentation of an exemplar model to be integrated with the tool. This guide is a companion to another report by the U.S. Army Combat Capabilities Development Command (DEVCOM) Analysis Center, known as DAC, *The HSI Metric Tradespace Exploration Environment (HMTee): A Paradigm for Integrating Quantitative Models of Human–System Performance* (Garneau, 2022). This guide is intended to be regularly updated and the latest version maintained with the software distribution.

2. HMTEE IMPLEMENTATION

This section discusses the HMTee implementation approach and information of relevance to those wishing to develop and integrate new interaction models with HMTee.

2.1. Language and Platform: R and R Shiny

HMTee relies upon models built in R, which is "a free software environment for statistical computing and graphics."[*] HMTee itself is built using R Shiny, which is "an R package that makes it easy to build interactive web apps straight from R."[†] R Shiny was selected as the implementation platform because it offers a flexible and extensible way to make an analysis easy to use and visualize by any user, even users who are not familiar with R (i.e., it capitalizes on the power of R without requiring users to have pre-existing knowledge of the language). Many in the greater analysis and modeling and simulation (M&S) community—including many practitioners in DAC—are familiar with R and Shiny, and data analysts in the future should be able to readily expand the models or functionality included in HMTee.

Shiny has traditionally used the Twitter Bootstrap framework, which provides a large array of user interface components and layout options. While Shiny is not the ideal framework for creating scalable enterprise applications, it works well for projects that are primarily analytical in nature and maintained by a small number of developer(s).

2.2. Dependencies

As of version 1.0, HMTee uses the R packages listed in this section; the purpose of the package within HMTee is noted alongside each package name and version number. As updates to these dependencies are made available by their respective developers, they will be incorporated into future versions of HMTee.

- **R:** Base environment required to execute any R function; v4.2.0
- **RStudio:** Development environment required for developing any R Shiny application; v22.07.1
- **shiny:** Required for any R Shiny application; v1.7.1
- **shinyjs**: Additional Shiny capabilities; v2.1.0
- **shinyWidgets**: Additional Shiny capabilities (e.g., slider skinning); v0.7.2
- **shinycssloaders:** Loading spinner for graphs; v1.0.0

[*] https://www.r-project.org/
[†] https://shiny.rstudio.com/

- **rjson:** Utilities for loading/exporting JavaScript Object Notation (JSON) files; v0.2.21
- **bslib:** Bootstrap theming; v0.3.1
- **stringr:** Utilities for working with text strings (e.g., `str_replace_all`); v1.4.0
- **shinyvalidate:** Tool for validating inputs in Shiny; v0.1.2
- **png:** Utility for writing PNG image files to the “tmp” directory for generated graphs; v0.1-7
- **pdftools:** Utility for converting PDF files for generated graphs; v3.3.0

2.3. Distribution

HMTee is currently available to end users or developers via direct transfer of a zip file containing the complete set of files in the HMTee working directory required to run HMTee. It is also tracked via a git repository hosted by the SITCORE instance of GitLab.[*] SITCORE (The Sensor Information Testbed Collaborative Research Environment) is a DEVCOM Army Research Laboratory Open Campus initiative. Access to the private HMTee GitLab project is by request only.

This guide assumes that a developer has downloaded the latest version of HMTee to their development machine (either by copying the HMTee zip file or cloning the git repository). The HMTee working directory will then contain the **config** subdirectory where the interaction model registry and individual interaction model configuration files may be found.

A `changelog.txt` file is included in the working directory of the distribution to track significant changes across versions of HMTee.

2.4. Interaction Model Integration

Interaction models are the building blocks that provide analytical capabilities in HMTee. The ability to readily expand HMTee through new interaction models is a key feature of HMTee. To integrate a new model with HMTee, a developer must do two things:

1. Register their interaction model in `config/interaction_models.txt`
2. Supply a `model.json` configuration file (see Section 3)

The interaction model registry is a simple comma-separated-value table with the following structure:

[*] https://gitlab.sitcore.net/dac-hsi-tools/hmtee

interaction_name	category	config
Display name of the interaction model	HSI domain of model; options are: `hfe`, `mpt`, `safety`, `forceprotection`, `habitability`, `meta`	name of the `model.json` configuration file (without the .json extension)

2.5. Requirements of Models to Be Integrated into HMTee

Models that are to be incorporated into HMTee must meet the following basic requirements:

- Models must be implemented as R functions; the functions may be bundled into an R package but this is not a requirement.
- Models must express an output measure as a function of one or more human parameter(s).

2.6. Optional Capabilities of Models

Additionally, models may also implement the following capabilities:

- Generate one or more plots and save to an output directory.
- Provide a list of output data that may be used to generate interactive plots.
- Implement filters on human parameters.
- Implement additional model options to be passed to the model function(s) interactively.

2.7. Customizable HMTee Elements

The following elements of the HMTee interface are customizable based on the fields entered in the interaction model configuration (`model.json`):

- Interaction Model Selection Modal
- Add System Component Modal
- Model Options Modal
- Human Parameters Filter Modal
- Interaction Model Output and Graphing Pane

See Section 4.3 for a screenshot of each of these components configured for the exemplar in Section 4. For a complete overview of the HMTee interface, see Garneau (2022).

3. HMTEE APPLICATION PROGRAMMING INTERFACE

Interaction models are integrated into HMTee via a JSON configuration file. This section discusses the specification of each of the elements in a `model.json` interaction model. Each of the following sections describes a top-level parameter, with additional parameters listed and described by subsections.

The following conventions are observed within the model JSON:

- A double exclamation point (`!!`) preceding any entry indicates that there is code to execute when the model is loaded.
- A double "at" sign (`@@`) preceding any entry indicates that there is code to execute when the model needs it.
- A double tilde (`~~`) preceding `options`, `filters`, `parameters`, or `parameterVals` indicates that these fields are to be replaced by vectors of their respective values.

The configuration file must have all parameters defined; if a parameter is not needed for a particular model, an empty string or array (as appropriate) should be defined for the parameter.

3.1. modelName

Acceptable data types: string

The `modelName` parameter is the name of the model and must match both the configuration file name (i.e., `model.json`) as well as the `config` property in the model registry.

3.2. library

Acceptable data types: statement

If a model uses an R package, the `library` parameter provides an opportunity to specify it. Note that this parameter must be expressed as a statement preceded by double exclamation points. For instance:

- To load a single package dependency: `!!library(libraryName)`
- To load multiple package dependencies: `!!lapply(c("libraryName1","libraryName2"),require, character.only=TRUE)`

If the interaction model has not been implemented as a package and is instead stored within an R file (e.g., `script.R`) in the working directory, it may be loaded with the `library` parameter via: `!!source("script.R")`.

3.3. humanParameters

Acceptable data types: string array, statement

The `humanParameters` property specifies the list of human attributes that are available for the model. This property specifies the list of parameters onto which a system component may map. The vector of selected parameters is passed to the model via `~~parameters`. Note that this property may also be used to specify system components that affect a model of human performance but do not map directly to a human parameter.

3.4. excludeParameters

Acceptable data types: string array, statement

The `excludeParameters` property specifies the list of human attributes within `humanParameters` that should not be displayed in the human attributes list. This is useful, for example, if `humanParameters` is populated by a function that includes certain parameters that are not to be listed as mappable human parameters.

3.5. filters

Acceptable data types: object array

The `filters` property allows for specification of a list of filters that may be applied to human attributes. The vector of selected filters is sent to the model via `~~filters` in the following format:

```
    c(<parameter1>, <comparator1>, <value1>, <parameter2>,
<comparator2>, <value2>, ...)
```

The model function may use these values in any way desired. Each specified filter must be structured as an object with the following properties.

3.5.1. parameter

Acceptable data types: string

The `parameter` property specifies the human parameter that is to be filtered. The `parameter` will be passed to the model function upon execution and is also used as the display name in the HMTee interface (as of version 1.0).

3.5.2. comparator

Acceptable data types: R relational operator expressed as string

The `comparator` property specifies the relationship between the `parameter` and the `values`. `comparator` should be an R relational operator (i.e., acceptable values are: `"<"`, `">"`, `"<="`, `">="`, `"=="`, or `"!="`).

3.5.3. values

Acceptable data types: string array, number array, statement

The `values` property lists the values that may be used to filter the specified `parameter`.

3.6. mappings

Acceptable data types: object array

The `mappings` property allows for specification of a list of acceptable mapping types that establish the possible relationships between system components and human parameters. Each mapping type must be structured as an object with the following properties.

3.6.1. name

Acceptable data types: string (from list of options)

The `name` property provides a description of the mapping type to be displayed in HMTee. In version 1.0, HMTee looks for the following mapping types:

- One-to-one
- One-to-one with offset
- Scaled
- `Unmapped` (functionally identical to `One-to-one`, but may make more sense to the user if a system component is configurable without mapping to a human parameter)

3.6.2. inputType

Acceptable data types: string (from list of options)

The `inputType` property defines the type of interface element that HMTee should display to the user for capturing the constraints on a system parameter. In version 1.0, options are:

- `range-slider` (numerical range with lower and upper limit)
- `slider` (numerical value with lower and upper limit)

Any other string supplied to `inputType` will result in nothing being displayed for the component when added to an analysis.

3.6.3. minVal

Acceptable data types: number, statement

For `slider` or `range-slider` input types, the `minVal` property sets the lower limit.

3.6.4. maxVal

Acceptable data types: number, statement

For `slider` or `range-slider` input types, the `maxVal` property sets the upper limit.

3.7. supportedGraphTypes

Acceptable data types: string array (from list of options)

The `supportedGraphTypes` property indicates the type of interactive/custom graphs that the interaction model will support. This property does not affect any graphs generated within the interaction model functions. In version 1.0, options are: `Scatter (Population)` or none (empty array).

3.8. generatedGraphs

Acceptable data types: object array

The `generatedGraphs` property allows for specification of a list of graphs generated within the interaction model that are saved to a file. Each item in `generatedGraphs` must be structured as an object with the following properties.

3.8.1. name

Acceptable data types: string

The `name` property provides the name for the graph type that is displayed in HMTee.

3.8.2. filepath

Acceptable data types: string

The `filepath` property specifies the path to the saved image, relative to the top level of the working directory. For instance, if the model saves a PDF graph to the **plots** folder, `filepath` would be: `"/plots/graph.pdf"`.

3.9. graphData

Acceptable data types: object array

The `graphData` property specifies the output data from the model that may be used to create interactive custom graphs. Currently (version 1.0), HMTee looks for an object array for this property containing two objects; one object containing the `accommodated` property and the other object containing the `disaccommodated` property. These datasets are used in the “Scatter (Population)” graph type to plot the portion of the population that is accommodated and disaccommodated. Each of these datasets should contain an R data frame with named columns corresponding with the configured `humanParameters` (which are then selectable for interactive graphing).

3.10. model

Acceptable data types: statement

The `model` property specifies the statement with the R function (or functions) that are to be executed when a user selects “Run Model(s)” in HMTee. The value returned by the model function should be of the numerical or string data type; this returned value appears next to the model name as the output in the “Interaction Models” pane.

Recall that this statement should begin with `@@` and include references to `~~parameters`, `~~parameterVals`, `~~options`, and `~~filters`, as appropriate. While none of these calls are required for execution, even a basic interaction model should include references to `~~parameters` and `~~parameterVals` (unless the model is intended to return a value that is independent of the selected parameters).

For example, if a model may be expressed as the following R function

```
model <- func_name(measures=c("stature","sittingheight"),
measureVals=c(1800,900), filters=c("age","==", 30))
```

then the `model` statement included in the interaction model configuration should be

```
@@func_name(measures=~~parameters,measureVals=~~parameterVals,
filters=~~filters)
```

where the `~~parameters`, `~~parameterVals` and `~~filters` are populated by values selected by the HMTee user.

3.11. modelUnits

Acceptable data types: string

The `modelUnits` parameter specifies the units to be displayed next to the model output in the Interaction Model pane. If no units are to be displayed, specify an empty string.

3.12. modelOptions

Acceptable data types: object array

The `modelOptions` property allows for specification of a list of options to be passed to the model. Once a user has selected one or more options, the result is that `~~options` is replaced at model execution by a string of one or more `parameter=value` pairs for each of the configured options. Each option must be structured as an object with the following properties.

3.12.1. name

Acceptable data types: string

The `name` property provides the name for the option that is displayed in HMTee.

3.12.2. parameter

Acceptable data types: string

The `parameter` property provides the name for the option that is passed to the model upon execution.

3.12.3. choices

Acceptable data types: object array

The `choices` property allows for specification of a list of choices for the given options. Each choice must be structured as an object with the following properties.

3.12.3.1. name

Acceptable data types: string

The `name` property provides the name for the option choice that is displayed in HMTee.

3.12.3.2. value

Acceptable data types: number or string

The `value` property provides the value for the option choice that would be passed to the model statement upon execution in the `parameter=value` pair.

3.12.3.3. default

Acceptable data types: string (`"true"` or `"false"`)

The `default` property indicates whether the option should be selected by default without user input. Only one choice should be set with `default` as `"true"`.

4. EXEMPLAR HMTEE INTERACTION MODEL

Suppose an HSI practitioner is interested in determining whether required manual lifting tasks for a system are acceptable. Since their team regularly assesses lifting tasks as part of system performance, the practitioner would like to implement the widely used Revised NIOSH (National Institute for Occupational Safety and Health) Lifting Equation (RNLE) in HMTee (Waters et al., 2021). While many online calculators exist, there does not currently exist a publicly available implementation of the RNLE in R. Moreover, this practitioner would like to include recent research in their analysis that considers additional characteristics for inclusion in the RNLE like Body Mass Index (BMI), age, and gender (Barim et al., 2019). The practitioner would also like to include a graph with the output metric and constituent multipliers from the RNLE.

4.1. Problem Definition

From the *Applications Manual for the Revised NIOSH Lifting Equation* (Barim et al., 2019), the Recommended Weight Limit (RWL) for a manual lifting task is based on a simple equation that provides weightings for each of six task variables (expressed as coefficients that decrease the load weight to be lifted):

$$RWL = LC \times HM \times VM \times DM \times AM \times FM \times CM$$

where:

LC = Load Constant
HM = Horizontal Multiplier
VM = Vertical Multiplier
DM = Distance Multiplier
AM = Asymmetric Multiplier
FM = Frequency Multiplier
CM = Coupling Multiplier

Detailed discussion of these multipliers is left to the Applications Manual. The R functions to calculate RWL (Section 4.2.1) implement the appropriate lookup tables to determine each of these multipliers given the appropriate inputs.

Barim et al. (2019) suggests the following modification to the RWL:

$$RWL\_ = RWL \times GM \times BMIM \times AGEM \times IVDM$$

where:

GM = Gender Multiplier
BMIM = BMI Multiplier
AGEM = Age Multiplier
IVDM = Low Back Intervertebral Disc Size

These additional multipliers are associated with simple rules based on the given gender (male = 1; female = 2/3), BMI (under 30 = 1; over 30 = 30/BMI), and age (under 40 = 1; over 40 = 1-1%*[age 40]). IVDM is not considered here since values for this parameter are not readily available.

The Lifting Index (LI), which estimates physical stress associated with a job, may be calculated from the Load Weight (L) and RWL or RWL_ via the equation:

$$LI = L / RWL\_$$

Generally, the physical stress of a lifting task increases with an increasing LI; LI > 1 poses an increased risk of lifting-related low back pain and LI = 3 is the approximate maximum of a highly stressful lifting task.

4.2. Model Formulation

The task parameters associated with the multipliers in Section 4.1 are horizontal and vertical hand location (H and V), vertical distance/displacement of a load to be lifted (D), asymmetry angle (A), frequency rate (Fr), duration (Tm), and coupling type (CT). Age, gender, and BMI are also to be considered. To calculate LI (the desired outcome metric), a function must be developed such that

$$LI = f(L, H, V, D, A, Fr, Tm, CT, age, gender, bmi)$$

Formulation of the model functions and configuration follows. Note that the functions and configuration in Sections 4.2.1, 4.2.2, and 4.2.3 are available within the HMTee distribution in `config/rnle.R` and `config/rnle.JSON`.

4.2.1. Implementation as R Function(s)

The following R functions return the desired output metric given the previous inputs; the functions `HM`, `VM`, `DM`, `AM`, `FM`, `CM`, `GM`, `BMIM`, and `AGEM` (not shown here) implement lookup tables from NIOSH (Barim et al., 2019) and rules from Waters et al. (2021).

```
rnle <- function(L,H,V,D,A,Fr,Tm,CT,units){
      if (units=='cm'){
            L <- L/0.45
```

```
            H <- H/2.5
            V <- V/2.5
            D <- D/2.5
            LC <- 23 #kg
      } else {
            LC <- 51 #lbs
      }

      RWL <- LC * HM(H) * VM(V) * DM(D) * AM(A) * FM(Fr,Tm,V) * VM(CT,V)
      LI <- round(L/RWL,digits=1)
      return(list(li=LI,rwl=RWL))
}

rnle_plus <- function(L,H,V,D,A,Fr,Tm,CT,gender,bmi,age,units){
      RWL <- rnle(L,H,V,D,A,Fr,Tm,CT,units)$rwl * GM(gender) * BMIM(bmi) *
AGEM(age)
      LI <- round(L/RWL,digits=1)
      return(list(li=LI,rwl=RWL))
}
```

4.2.2. HMTee Interaction Model Function

While the functions in Section 4.2.1 yield the desired outcome metric given the required parameters, they are not in a convenient format to be included in the HMTee interaction model configuration. It is also necessary at this step to consider how the various components of the RNLE will be presented in HMTee. Recall that HMTee provides the ability to specify System Components, filters on Human Parameters, and Model Options. These three parts will be utilized as follows for the RNLE interaction model:

- **Available system components/parameters**: `load`, `hand_location_horizontal`, `hand_location_vertical`, and `vertical_distance`; these parameters were chosen because they are required for virtually all lifting analyses
- **Model options**: Asymmetry Angle (`A`), Frequency Rate (`Fr`), Duration (`Tm`), Coupling Type (`CT`), and `units`
- **Filters**: `age`, `bmi`, and `gender`

The following function packages `rnle_plus` for inclusion in HMTee.

```
rnle_model <-
function(parameters=c(),parameterVals=c(),filters=c(),A,Fr,Tm,CT,units){
```

```
      L <- 0
      H <- 0
      V <- 30
      D <- 0
      age <- 25
      bmi <- 25
      gender <- "male"
      # assign parameters...
      for (i in seq(1:length(parameters)){
            if (parameters[i]=="load"){
                  L <- as.numeric(parameterVals[i])
            } else if (parameters[i]=="hand_location_horizontal"){
                  H <- as.numeric(parameterVals[i])
            } else if (parameters[i]=="hand_location_vertical"){
                  V <- as.numeric(parameterVals[i])
            } else if (parameters[i]=="vertical_displacement"){
                  D <- as.numeric(parameterVals[i])
            }
      }
      # assign filters…
      if (!is.na(filters)&&length(filters)>0&&filters!="NA"){
            for (i in seq(1:(length(filters)/3))){
            # if filter value can be converted to a number, do it...
            if
(!suppressWarnings(is.na(as.numeric(eval(parse(text=filters[3*i])))))){
                  filters[3*i]<-as.numeric(eval(parse(text=filters[3*i])))
            }
            # execute assignment...
            eval(parse(text=paste0(filters[3*i-2],"=",filters[3*i]))) # note
replacement of comparator with "="
            }
      }

      li <- rnle_plus(L,H,V,D,A,Fr,Tm,CT,gender,bmi,age,units)$li

      pdf("plots/rnle.pdf",width=6,height=4)
      barplot(c(HM(H),VM(V),DM(D),AM(A),FM(Fr,Tm,V),CM(CT,V),GM(gender),BMIM(
bmi),AGEM(age),li),ylab='Multiplier', main='Task Multipliers and Lifting
Index', names.arg=c("HM","VM","DM","AM","FM","CM","GM","BMIM","AGEM","LI"),
```

```
col=c("black","black","black","black","black","black","black","black","black"
,"#FFCC06"),cex.names=0.7)
      dev.off()


      return(li)
}
```

4.2.3. HMTee Interaction Model Configuration

Now that a suitable interaction model function has been configured, the interaction model JSON configuration file can be developed following the API in Section 3, as follows. Note that for the model options, up to four choices per option (e.g., “Asymmetry Angles” of 0, 45, 90, and 135) were implemented to abbreviate the length of the configuration file, but more could be included to yield greater resolution if needed.

```
{
        "modelName": "rnle",
        "library": "!!source('config/rnle.R')",
        "humanParameters":
["load","hand_location_horizontal","hand_location_vertical","vertical_distanc
e"],
        "excludeParameters": [],
        "filters": [
          {
            "parameter":"age",
            "comparator": ["=="],
            "values":[25,30,35,40,45,50,55,60,65]
         },{
           "parameter":"bmi",
           "comparator": ["=="],
           "values":[28,29,30,31,32,33,34,35,36,37,38,39,40]
         },{
           "parameter":"gender",
           "comparator": ["=="],
           "values":["male","female"]
         }
         ],
        "mappings": [
                {
                        "name": "One-to-one",
                        "inputType": "slider",
                        "minVal": "0",
                        "maxVal": "60"
                }
        ],
        "suggestedGraphs":[],
        "generatedGraphs":
        [
          {
            "name": "Bar",
            "filepath": "/plots/rnle.pdf"
```

```
          }
         ],
         "graphData": [],
        "model":
"@@rnle_model(parameters=~~parameters,parameterVals=~~parameterVals,filters=~
~filters,~~options)",
        "modelUnits": " (LI)",
        "modelOptions": [
          {
            "name": "Assymmetry Angle (deg)",
            "parameter": "A",
            "choices": [
              {
                "name": "0",
                "value": "0",
                "default": "true"
              },{
                "name": "45",
                "value": "45",
                "default": "false"
              },{
                "name": "90",
                "value": "90",
                "default": "false"
              },{
                "name": "135",
                "value": "135",
                "default": "false"
              }
             ]
          },
          {
            "name": "Frequency (lifts/min)",
            "parameter": "Fr",
            "choices": [
              {
                "name": "<=0.2",
                "value": "0.1",
                "default": "true"
              },{
                "name": "1",
                "value": "1",
                "default": "false"
              },{
                "name": "5",
                "value": "5",
                "default": "false"
              },{
                "name": "10",
                "value": "10",
                "default": "false"
              }
             ]
          },
          {
            "name": "Duration (hours)",
```

```
            "parameter": "Tm",
            "choices": [
              {
                "name": "<1",
                "value": "0.5",
                "default": "true"
              },{
                "name": "1-2",
                "value": "1.5",
                "default": "false"
              },{
                "name": "2-8",
                "value": "2.5",
                "default": "false"
              }
            ]
          },
          {
            "name": "Coupling Type",
            "parameter": "CT",
            "choices": [
              {
                "name": "Good",
                "value": "Good",
                "default": "true"
              },{
                "name": "Fair",
                "value": "Fair",
                "default": "false"
              },{
                "name": "Poor",
                "value": "Poor",
                "default": "false"
              }
            ]
          },
          {
            "name": "Units",
            "parameter": "units",
            "choices": [
              {
                "name": "Inch/Pounds",
                "value": "in",
                "default": "true"
              },{
                "name": "Centimeters/Kilograms",
                "value": "cm",
                "default": "false"
              }
            ]
          }
        ]
}
```

4.3. Results

The last step to configure the interaction model is to add it to the interaction model registry (`config/interaction_models.txt`) as follows:

```
interaction_name,category,config
"Physical environment (size)",hfe,marc
"Job skills",mpt,jass
"Manual lifting",hfe,rnle
```

HMTee may now be run with the new interaction model. Upon adding the “Manual lifting” interaction to a blank analysis (Figure 1), the user may add system components for the four configured parameters (Figure 2) and select values for the components. Any of the configured filters (Figure 3) and options (Figure 4) may then be applied. Upon selecting “Run Model(s)”, the user is provided with the LI for the configured components, along with a graph of the Task Multipliers and Resulting LI (Figure 5). Note that the Session Log output makes it convenient to compare two lifting tasks or two endpoints of the same task (e.g., origin and destination). To save an output graph, users may take a screenshot of the output pane; future versions of HMTee will include a built-in option to save the graph to a file (e.g., PNG).

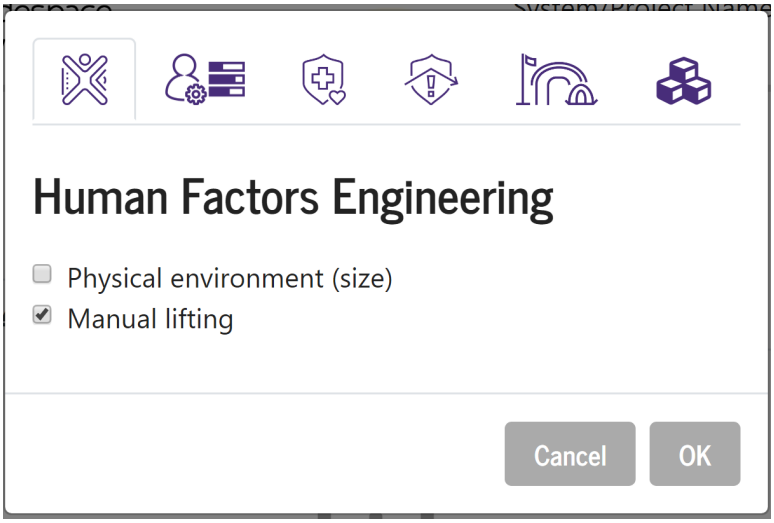

Figure 1. Interaction model selection

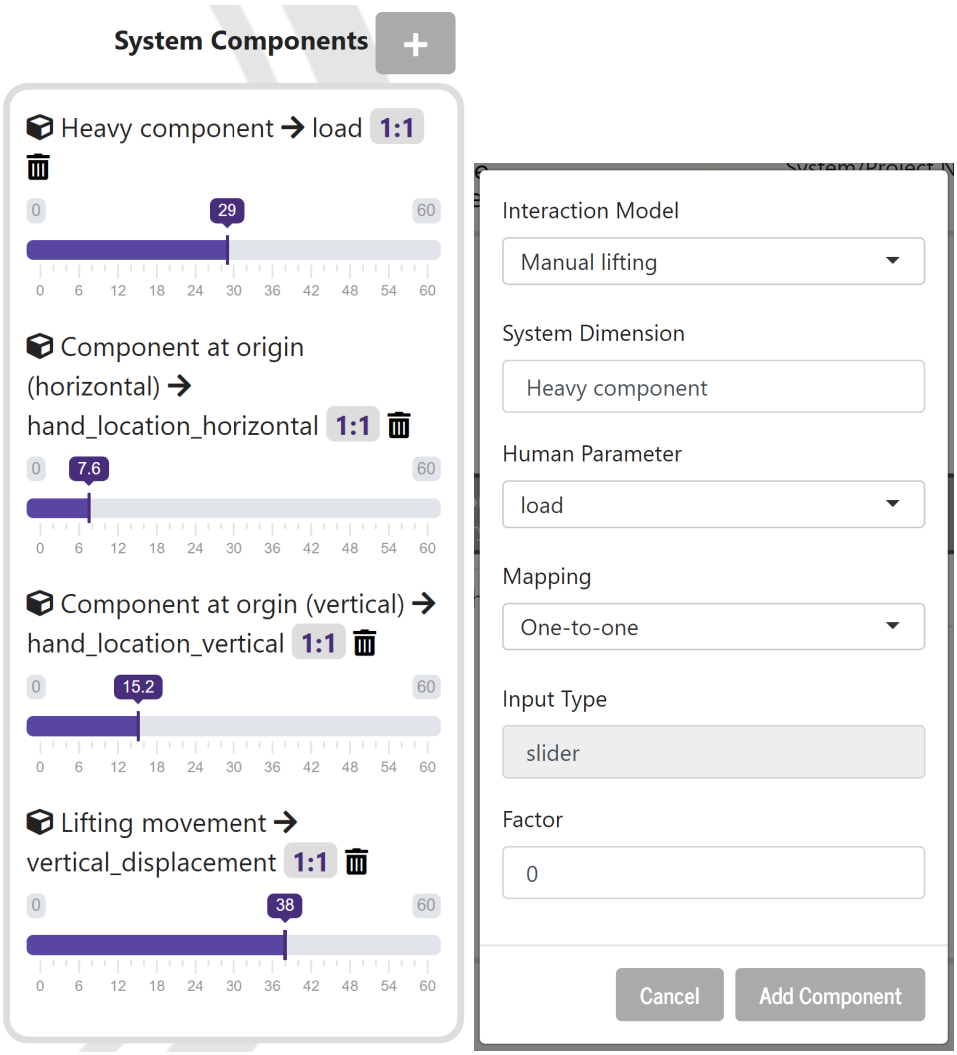

Figure 2. System Components pane (left) and Add System Component modal (right)

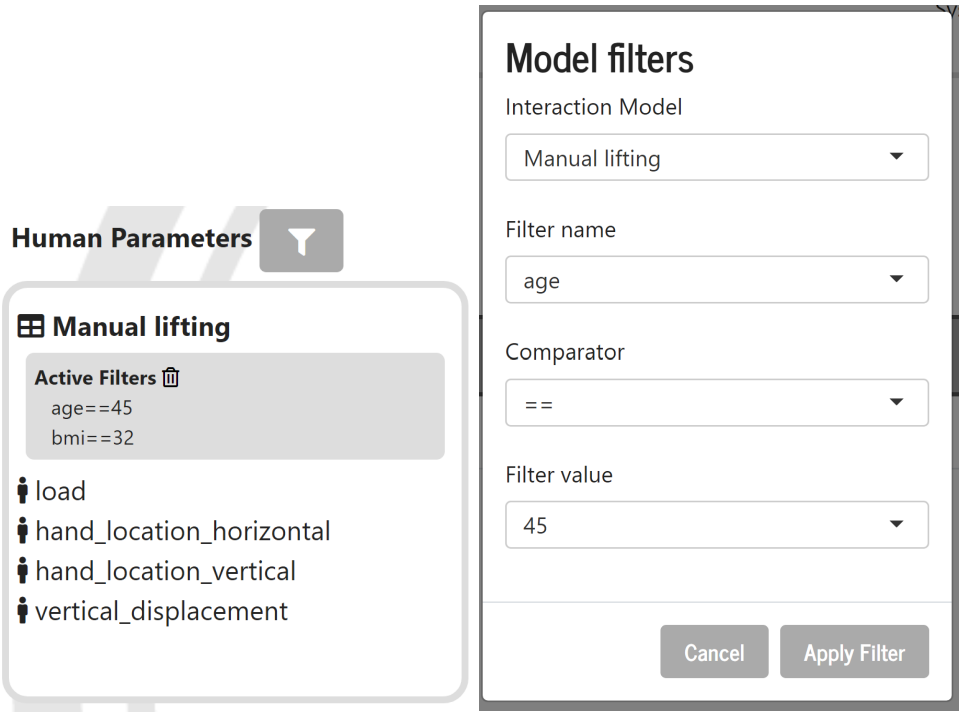

Figure 3. Human Parameters pane (left) and Add Filter modal (right)

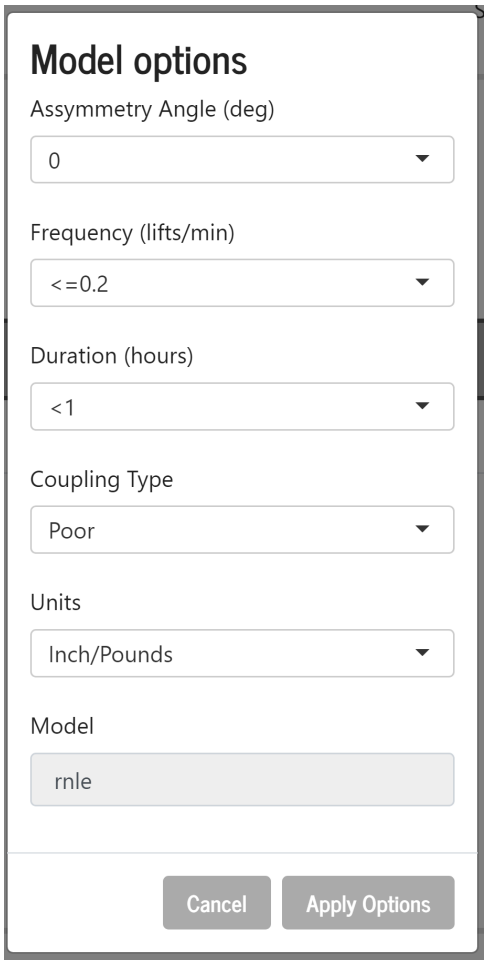

Figure 4. Model Options modal

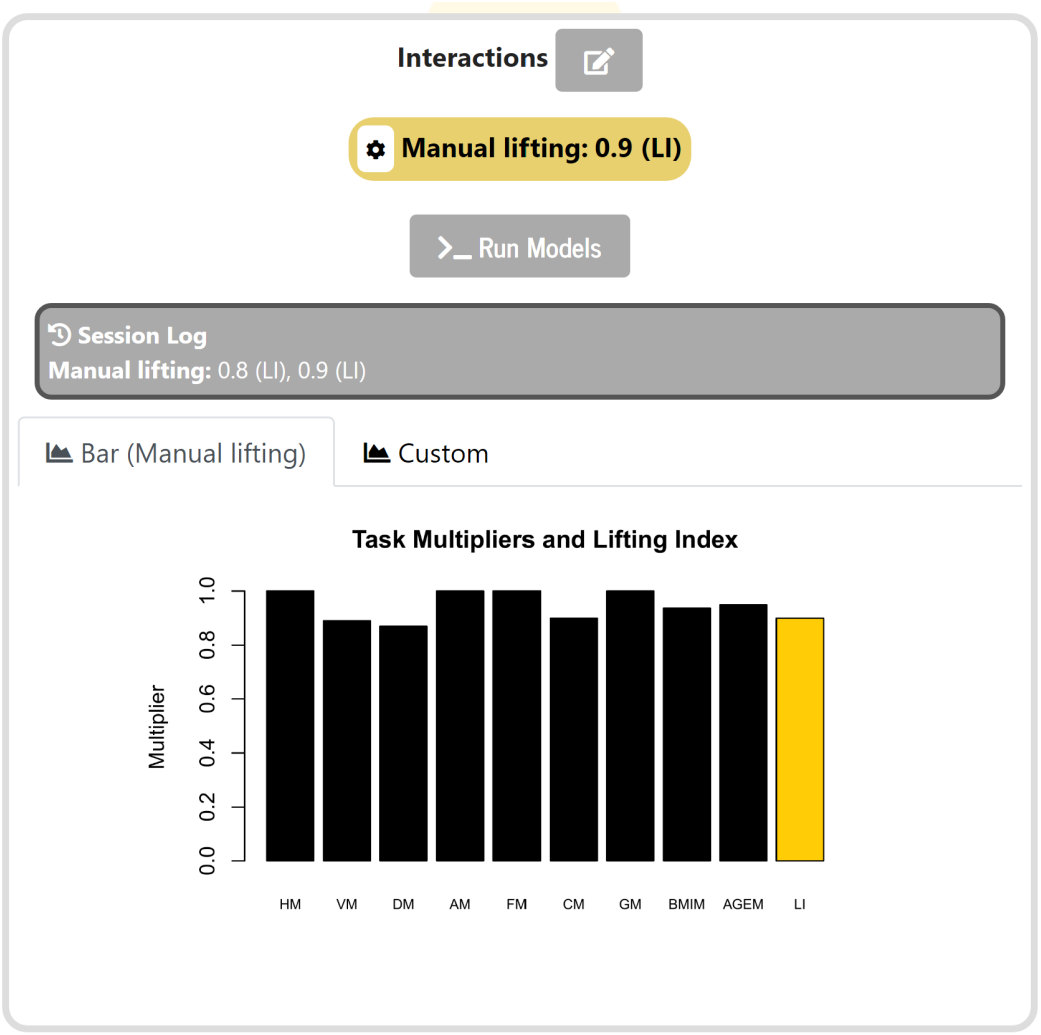

Figure 5. Interactions/results output pane

5. CONCLUSION

HMTee provides a toolset for optimizing system designs or requirements with respect to the domains of HSI. This documentation described the implementation of the software in the R Shiny environment, discussed the interaction model paradigm as a means for integrating analytical capabilities, documented the path to implementing new interaction models via an API, and provided an exemplar illustrating the process.

6. REFERENCES

Garneau, C. (Forthcoming). The HSI metric tradespace exploration environment (HMTee): a paradigm for integrating quantitative models of human–system performance. U.S. Army Combat Capabilities Development Command Analysis Center.

Barim, M.S., Sesek, R.F., Fehmi Capanoglu M., Drinkaus, P., Schall, Jr., M.C., Gallagher, S., & Davis, G.A. (2019, January). Improving the risk assessment capability of the revised NIOSH lifting equation by incorporating personal characteristics. *Applied Ergonomics*, 74, 67–73.

Waters, T.R., Putz–Anderson, V., & Garg, A. (2021, September). *Applications manual for the revised NIOSH lifting equation.* (DHHS [NIOSH] Publication No. 94-110). U.S. Department of Health and Human Services, Centers for Disease Control and Prevention, National Institute for Occupational Safety and Health. https://doi.org/10.26616/NIOSHPUB94110revised092021

LIST OF ACRONYMS

A	asymmetry angle
AGEM	Age Multiplier
AM	Asymmetric Multiplier
API	Application Programming Interface
BMI	Body Mass Index
BMIM	BMI Multiplier
CM	Coupling Multiplier
CT	coupling type
D	vertical distance/displacement of a load to be lifted
DAC	DEVCOM Analysis Center
DEVCOM	U.S. Army Combat Capabilities Development Command
DM	Distance Multiplier
Fr	frequency rate
FM	Frequency Multiplier
FY	fiscal year
GM	Gender Multiplier
IVDM	Low Back Intervertebral Disc Size
JSON	JavaScript Object Notation
H	horizontal
HM	Horizontal Multiplier
HMTee	Human Systems Integration Metric Tradespace Exploration Environment
HSI	Human–Systems Integration
L	Load Weight
LC	Load Constant
LI	Lifting Index
M&S	Modeling and Simulation
NIOSH	National Institute for Occupational Safety and Health

PDF	Portable Document Format
PNG	Portable Network Graphics
RNLE	Revised NIOSH Lifting Equation
RWL	Recommended Weight Limit
SITCORE	The Sensor Information Testbed Collaborative Research Environment
Tm	duration
V	vertical
VM	Vertical Multiplier

ORGANIZATION

DEVCOM Analysis Center
FCDD-DAH-C/C. Garneau
FCDD-DAH-C/A. Bodenhamer
FCDD-DAH-N/D. Patton
FCDD-DAG-S/K. Loftis
FCDD-DAW-J/A. Barnett
FCDD-DAW-W/S. Snead
FCDD-DAW-W/ R. Bowers
FCDD-DAW-W/T. Myers
FCDD-DAW-W/A. Kulaga
FCDD-DAD-T/T. Stadterman
6896 Mauchly St.
Aberdeen Proving Ground, MD 21005-5071

DEVCOM Army Research Laboratory
FCDD-RLD-DCI/Tech Library
2800 Powder Mill Rd.
Adelphi, MD 20783

Defense Technical Information Center
ATTN: DTIC-O
8725 John J. Kingman Rd.
Fort Belvoir, VA 22060-6218