Requirements Specification

# Requirements Specification: mrgmisc 0.1.0

## Scope

The purpose of this document is to define specific criteria for each testing task. Testing shall be conducted in accordance with the requirements within this document. The Requirement Specifications ensure that each requirement is tested.

## User Story: ASNM-S001 as.nmctl

As a user, I want to be able to read and write control files for NONMEM.

**Product risk**: Low

**Requirements**

* ASNM-R001: read.nmctl appropriately reads in NONMEM ctl files
* ASNM-R002: write.nmctl appropriately writes out NONMEM ctl files

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-ASNM-001 | read.nmctl expected performance |
| MRG-ASNM-002 | write.nmctl expected performance: Able to write out control files |

## User Story: AUCI-S002 Calculate AUCt-inf

As a user, I want to calculate the area under the curve across an infinite range.

**Product risk**: Low

**Requirements**

* AUCI-R001: auc\_inf accurately returns the area under the curve across an infinite range
* AUCI-R002: auc\_inf appropriately handles NA values and returns expected output if all DV is NA
* AUCI-R003: auc\_inf appropriately handles cases where all DV is 0

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-AUCI-001 | AUC\_inf works as intended |
| MRG-AUCI-002 | AUC\_inf NA testing |
| MRG-AUCI-003 | AUC\_inf unexpected input testing: Correct output if all |

## User Story: AUCP-S001 Create partial AUC

As a user, I want to perform a simple area under the curve calculation over a discrete range.

**Product risk**: Low

**Requirements**

* AUCP-R001: auc\_partial returns the area under the curve across a discrete range

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-AUCP-001 | auc\_partial works with simple AUC calculation |

## User Story: CHNK-S001 Chunk

As a user, I want to group elements in vectors and lists with easily customizable grouping rules.

**Product risk**: Low

**Requirements**

* CHNK-R001: chunk accurately creates groups with different input formats
* CHNK-R002: chunk\_grp accurately creates groups while respecting grouped variables
* CHNK-R003: chunk\_list and chunk\_grp\_list accurately creates groups in list format customized to specified groups.

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-CHNK-001 | chunk expected output: Chunking |
| MRG-CHNK-002 | chunks unique groups as evenly as possible |
| MRG-CHNK-003 | chunk with groups: Chunks |

## User Story: CKDF-S001 Chunk dataframe

As a user, I want to create easily customizable groups within a dataframe.

**Product risk**: Low

**Requirements**

* CKDF-R001: chunk\_df creates expected number of groups
* CKDF-R002: chunk\_df handles inappropriate inputs and offers multiple output formats

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-CKDF-001 | chunk\_df expected output |
| MRG-CKDF-002 | chunk\_df unexpected input |

## User Story: CPNM-S001 Capitalize names

As a user, I want to easily capitalize the column names of my tibble.

**Product risk**: Low

**Requirements**

* CPNM-R001: capitalize\_names sets the first letter of the column names to uppercase

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-CPNM-001 | capitalize\_names outputs expectation |

## User Story: IDPL-S001 Create groups of IDs

As a user, I want to split IDs within a data set into appropriate groups for subplots.

**Product risk**: Low

**Requirements**

* IDPL-R001: ids\_per\_plot splits IDs into groups to use for subsequent plotting

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-IDPL-001 | ids\_per\_plot expected output |

## User Story: MMAX-S001 max\_through

As a user, I want to find the maximum value found up to each index of the input.

**Product risk**: Low

**Requirements**

* MMAX-R002: max\_through accurately finds the maximum value up to each index in the input
* MMAX-R003: min\_through and max\_through appropriately hands inputs of different formats

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-MMAX-002 | max\_through properly calculates values |
| MRG-MMAX-003 | min\_through and max\_through |

## User Story: MMIN-S001 max\_through

As a user, I want to find the minimum value found up to each index of the input.

**Product risk**: Low

**Requirements**

* MMAX-R001: min\_through accurately finds the minimum value up to each index in the input
* MMAX-R003: min\_through and max\_through appropriately hands inputs of different formats

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-MMAX-001 | min\_through properly calculates values |
| MRG-MMAX-003 | min\_through and max\_through |

## User Story: NASM-S001 nasum

As a user, I want to quickly find the total number of NA values in each column.

**Product risk**: Low

**Requirements**

* NASM-R001: nasum finds all the NA values present in each column of a data frame

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-NASM-001 | Finds NA within dataframe |

## User Story: ORBI-S001 ordinal\_to\_binary\_

As a user, I want to expand a single column to the appropriate number of binary flag columns.

**Product risk**: Low

**Requirements**

* ORBI-R001: ordinal\_to\_binary\_ creates the expected number of binary flag variables
* ORBI-R002: ordinal\_to\_binary\_ has appropriate outputs for special cases

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-ORBI-001 | ordinal\_to\_binary\_ expected output |
| MRG-ORBI-002 | ordinal\_to\_binary\_ unique case |

## User Story: PADL-S001 pad\_left

As a user, I want to add a customizable prefix to every element in a vector.

**Product risk**: Low

**Requirements**

* PADL-R001: pad\_left adds the given character to the left of each element in a vector

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-PADL-001 | pad\_left expected outcome |

## User Story: POOL-S001 pool

As a user, I want to determine which elements are shared or unique between two vectors.

**Product risk**: Low

**Requirements**

* POOL-R001: pool determines the shared and unique elements between two vectors

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-POOL-001 | pool creates appropriate groupings from two vectors with similarities |

## User Story: PRNS-S001 parens

As a user, I want to add parenthesis to a given element.

**Product risk**: Low

**Requirements**

* PRNS-R001: parens adds paranthesis to the given character

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-PRNS-001 | parens works as expected |

## User Story: PSMT-S001 posmat

As a user, I want to coerce a matrix to be positive definite.

**Product risk**: Low

**Requirements**

* PSMT-R001: posmat produces an accurate output from a given square matrix input
* PSMT-R002: posmat responds appropriately to unintended input formats

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-PSMT-001 | posmat expected output |
| MRG-PSMT-002 | posmat error: Correct error occurs with non-square matrix |

## User Story: RDOT-S001 replace\_dots

As a user, I want to easily convert dots in a vector to NA.

**Product risk**: Low

**Requirements**

* RDOT-R001: replace\_dots converts dots in a vector to NA
* RDOT-R002: replace\_dots appropriately hands special case inputs

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-RDOT-001 | repalce\_dots replaces . with NA |
| MRG-RDOT-002 | replace\_dots unexpected input |

## User Story: RESM-S001 resample\_df

As a user, I want to resample a dataframe with determined stratifications in the data.

**Product risk**: Low

**Requirements**

* RESM-R001: resample\_df appropriately resamples a dataframe
* RESM-R002: resample\_df allows for customization in stratification and replacement
* RESM-R003: resample\_df appropriately responds to unintended inputs

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-RESM-001 | resample\_df properly recombines when it needs to make extra draws |
| MRG-RESM-002 | resample\_df expected output: Works with no |
| MRG-RESM-003 | resample\_df unexpected input |

## User Story: SBDF-S001 set\_bins\_df

As a user, I want to create bins for data frame columns with optionally customizable bin ranges.

**Product risk**: Low

**Requirements**

* SBDF-R001: set\_bins\_df creates bins within a dataframe with specified breaks or default quartiles
* SBDF-R002: set\_bins\_df respects upper and lower bound arguments
* SBDF-R003: set\_bins\_df creates bins using inclusive arguments
* SBDF-R004: set\_bins\_df creates bins using between argument
* SBDF-R005: set\_bins\_df respects quiet argument
* SBDF-R006: set\_bins\_df appropriately applies name and label arguments

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-SBDF-001 | set\_bins\_df expected output |
| MRG-SBDF-002 | set\_bins\_df bounds argument |
| MRG-SBDF-003 | set\_bins\_df inclusive tests: inclusive = |
| MRG-SBDF-004 | set\_bins\_df between tests |
| MRG-SBDF-005 | set\_bins\_df quiet tests: quiet= |
| MRG-SBDF-006 | set\_bins\_df name and label tests |

## User Story: SNAP-S001 snap

As a user, I want to coerce values to nearest of given candidates.

**Product risk**: Low

**Requirements**

* SNAP-R001: snap coerces values to nearest of given candidates

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-SNAP-001 | snap expected output |

## User Story: SPAC-S001 s\_pauc

As a user, I want to easily summarize any given quantile.

**Product risk**: Low

**Requirements**

* SPAC-R001: s\_pauc appropriately summarize the given quantile

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-SPAC-001 | s\_pauc expected output |

## User Story: SQUN-S001 s\_quantiles

As a user, I want to summarize any quantile from a column in a data frame.

**Product risk**: Low

**Requirements**

* SQUN-R001: s\_quantiles summarizes quantiles for an individual column
* SQUN-R002: s\_quantiles appropriately responds to unexpected inputs

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-SQUN-001 | s\_quantiles expected output |
| MRG-SQUN-002 | s\_quantiles unexpected input: Appropriate error occurs |

## User Story: STBN-S001 set\_bins

As a user, I want to create bins from a column of data with optionally customizable bin ranges.

**Product risk**: Low

**Requirements**

* STBN-R001: set\_bins creates bins with specified breaks or default quartiles
* STBN-R002: set\_bins respects upper and lower bound arguments
* STBN-R003: set\_bins creates bins using inclusive arguments
* STBN-R004: set\_bins creates bins using between argument
* STBN-R005: set\_bins respects quiet argument

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-STBN-001 | set\_bins expected output |
| MRG-STBN-002 | set\_bins bound tests |
| MRG-STBN-003 | set\_bins inclusive tests: inclusive = |
| MRG-STBN-004 | set\_bins between tests |
| MRG-STBN-005 | set\_bins quiet tests: quiet= |

## User Story: WRNM-S001 Write NONMEM compatible csv file

As a user, I want to be able to create csv files that are compatible with NONMEM.

**Product risk**: Low

**Requirements**

* WRNM-R001: write\_nonmem easily writes a csv file compatible with NONMEM
* WRNM-R002: write\_nonmem appropriately handles special cases

**Tests**

| **Test ID** | **Test name** |
| --- | --- |
| MRG-WRNM-001 | write\_nonmem writes file to correct location |
| MRG-WRNM-002 | write\_nonmem special cases |

## Signature Page:

**Authored by:**

**Reviewed by:**

**Quality Assurance Approved by:**