CCMPP Input Data Structures Quick Guide

# Introduction

This is a quick guide to using the S3 classes for CCMPP input objects. See the vignette “S3 Classes for CCMPP Input Data Structures” for full details of the classes.

# The S3 Classes for CCMPP

## Role in CCMPP Workflow

ccmppWPP\_workflow\_one\_country\_variant is the workhorse function for performing the CCMPP on a single country. It takes a single argument, a list of data frame, each containing data for a component of demographic change (population at baseline, fertility, etc.). The S3 class ccmpp\_input\_list was created to

1. Rigorously define the input list (names of elements, structure of the data frames);
2. Encode the definition in *R* through a set of *constructor* and *validator* functions.

The constructor functions provide a way to create valid inputs to ccmpp\_input\_list from data frames. The validator functions provide a mechanism for validating arbitrary lists as members of the ccmpp\_input\_list class.

A single line is added to ccmppWPP\_workflow\_one\_country\_variant to enact the validation:

ccmppWPP\_workflow\_one\_country\_variant <- function(wpp\_input) {  
  
 wpp\_input <- as\_ccmpp\_input\_list(wpp\_input) #<<<<<<<< ADDED: VALIDATE INPUTS  
  
 # extract objects needed for ccmpp  
 ccmpp\_input <- data\_parse\_ccmpp\_input(indata = wpp\_input)  
  
 ... etc.  
 ## rest of code..

## Structure of CCMPP Input Lists

Table 1. Names, demographic dimensions, and value types of data frame components of ccmpp\_input\_list objects.

|  |  |  |
| --- | --- | --- |
| Element name | Dimensions | Value type |
| fert\_rate\_age\_f | time, age | rate |
| srb | time | ratio |
| survival\_ratio\_input\_df | time, sex, age | proportion |
| mig\_net\_rate\_age\_sex | time, sex, age | rate |
| mig\_net\_count\_age\_sex | time, sex, age | count |
| mig\_net\_count\_tot\_b | time | count |
| mig\_parameter | time | categorical |
| pop\_count\_age\_sex\_base | time, sex, age | count |
| life\_table\_age\_sex | indicator, time, sex, age | real |

The elements of a ccmpp\_input\_list object are in Table @ref(tab:ccmpp-input-subclass-restrictions). Each is a data frame with column names taken from the following list:

* indicator
* time\_start
* time\_span
* age\_start
* age\_span
* sex
* value

Each data frame is also given an S3 class with the same name as its element. The element fert\_rate\_age\_f is a data frame with class fert\_rate\_age\_f, and so on. These data frames have two *attributes* (meta-data), including “dimension” and “value\_type”. The “dimension” attribute indicates the demographic breakdown of the data. Allowed dimensions are

* indicator
* time
* sex
* age

The “value\_type” attribute indicates the type of values the value column contains. Allowed “value\_type”s are

* count
* rate
* ratio
* proportion
* percentage
* real
* categorical

Type “real” is a catch-all for numeric values that do not fall under any other type. It is also used when the value column contains a mixture of the other types, as is the case for life\_table\_age\_sex objects.

## Creating Data Frame Components

To create a data frame for inclusion in a ccmpp\_input\_list, use the constructor function of the same name:

library(ccmppWPP) #load the package first  
data(wpp\_input\_example) #load the example data  
  
## The 'fert\_rate\_age\_f' data frame in the example is a basic R data frame:  
class(wpp\_input\_example$fert\_rate\_age\_f)  
#> [1] "data.frame"  
  
## Use the constructor to create a data frame with extra class memberships:  
fert\_example <- fert\_rate\_age\_f(wpp\_input\_example$fert\_rate\_age\_f)  
#> 'non\_zero\_fert\_ages' set to '12, 13, 14, ... , 51, 52, 53  
class(fert\_example)  
#> [1] "fert\_rate\_age\_f" "ccmpp\_input\_df"   
#> [3] "demog\_change\_component\_df" "data.frame"  
  
## Look at the attributes of the new object  
names(attributes(fert\_example))  
#> [1] "names" "row.names" "class"   
#> [4] "dimensions" "value\_type" "age\_span"   
#> [7] "time\_span" "non\_zero\_fert\_ages"

The classed data frame has the standard data frame attributes, “names”, “row.names”, “class”, and “dimensions”. Additionally, it has attributes “dimensions”, “value\_type”, “age\_span”, “time\_span”, and “non\_zero\_fert\_ages”. These can be inspected with functions of the same name, e.g.,:

age\_span(fert\_example)  
#> [1] 1  
non\_zero\_fert\_ages(fert\_example)  
#> [1] 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36  
#> [26] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

Special print and summary methods have been written to concisely display the data frames:

## Print method  
fert\_example  
#> # A 'fert\_rate\_age\_f' with 7,070 rows.  
#> # dimensions = 'time, age'.  
#> # value\_type = 'rate', age\_span = '1', time\_span = '1'.  
#> # non\_zero\_fert\_ages = '12, 13, ... 52, 53'.  
#> time\_start age\_start time\_span age\_span value  
#> 1950 0 1 1 [zero]  
#> 1950 1 . . [zero]  
#> 1950 2 . . [zero]  
#> . . . . .  
#> 1950 12 . . 3.7238e-05  
#> 1950 13 . . 0.00048408465  
#> 1950 14 . . 0.00176102842  
#> # ... etc.

## Summary method  
summary(fert\_example)  
#> dimensions: time, age  
#> time: range = [1950, 2019] span = 1  
#> age: range = [0, 100] span = 1  
#> value\_type: rate  
#> ------------------------------------------------------------  
#> non\_zero\_fert\_ages: 12, 13, 14, ... , 51, 52, 53  
#> ------------------------------------------------------------  
#> table:  
#> time\_start age\_start time\_span age\_span value   
#> Min. :1950 Min. : 0 Min. :1 Min. :1 Min. :0.00000   
#> 1st Qu.:1967 1st Qu.: 25 1st Qu.:1 1st Qu.:1 1st Qu.:0.00000   
#> Median :1984 Median : 50 Median :1 Median :1 Median :0.00000   
#> Mean :1984 Mean : 50 Mean :1 Mean :1 Mean :0.02160   
#> 3rd Qu.:2002 3rd Qu.: 75 3rd Qu.:1 3rd Qu.:1 3rd Qu.:0.01119   
#> Max. :2019 Max. :100 Max. :1 Max. :1 Max. :0.25797

Similar functions exist for the other data frame elements.

## Validating Data Frame Components

At any time, the validity of an object as a member of its class can be checked using the function validate\_ccmpp\_object. This function inspects the objects “class” and calls the appropriate subfunction to check that the data structure and attributes of the object match the class definition. If they match the object is simply returned as-is; this is standard *R* convention. If the object is not valid, an error is signalled:

## Not run:  
## validate\_ccmpp\_object(wpp\_input\_example$fert\_rate\_age\_f)  
## Error in validate\_ccmpp\_object.default(wpp\_input\_example$fert\_rate\_age\_f) :   
## 'x' is not an object with a valid CCMPP object class. 'class(x) = data.frame'. Valid classes are 'life\_table\_age\_sex', 'mig\_parameter', 'mig\_net\_count\_tot\_b', 'mig\_net\_count\_age\_sex', 'mig\_net\_rate\_age\_sex', 'srb', 'pop\_count\_age\_sex\_base', 'survival\_ratio\_input\_df', 'fert\_rate\_age\_f', 'ccmpp\_input\_df', 'demog\_change\_component\_df'.  
  
## Valid objects are simply returned to the console  
validate\_ccmpp\_object(fert\_example)  
#> # A 'fert\_rate\_age\_f' with 7,070 rows.  
#> # dimensions = 'time, age'.  
#> # value\_type = 'rate', age\_span = '1', time\_span = '1'.  
#> # non\_zero\_fert\_ages = '12, 13, ... 52, 53'.  
#> time\_start age\_start time\_span age\_span value  
#> 1950 0 1 1 [zero]  
#> 1950 1 . . [zero]  
#> 1950 2 . . [zero]  
#> . . . . .  
#> 1950 12 . . 3.7238e-05  
#> 1950 13 . . 0.00048408465  
#> 1950 14 . . 0.00176102842  
#> # ... etc.

## Creating Input Lists

### From Data Frames

The function ccmpp\_input\_list can be used to create a valid input list from constituent data frames:

ccmpp\_input\_list\_example <-  
 ccmpp\_input\_list(pop\_count\_age\_sex\_base = wpp\_input\_example$pop\_count\_age\_sex\_base,  
 life\_table\_age\_sex = wpp\_input\_example$life\_table\_age\_sex,  
 fert\_rate\_age\_f = wpp\_input\_example$fert\_rate\_age\_f,  
 srb = wpp\_input\_example$srb,  
 mig\_net\_count\_age\_sex = wpp\_input\_example$mig\_net\_count\_age\_sex,  
 mig\_net\_rate\_age\_sex = wpp\_input\_example$mig\_net\_rate\_age\_sex,  
 mig\_net\_count\_tot\_b = wpp\_input\_example$mig\_net\_count\_tot\_b,  
 mig\_parameter = wpp\_input\_example$mig\_parameter)  
#> 'non\_zero\_fert\_ages' set to '12, 13, 14, ... , 51, 52, 53  
  
class(ccmpp\_input\_list\_example)  
#> [1] "ccmpp\_input\_list" "list"

### From a List

If the user has already assembled a list with the necessary components and proper element names, as\_ccmpp\_input\_list can be used to *coerce* it to a member of the class:

ccmpp\_input\_list\_example\_2 <-  
 as\_ccmpp\_input\_list(wpp\_input\_example)  
#> 'non\_zero\_fert\_ages' set to '12, 13, 14, ... , 51, 52, 53  
  
class(ccmpp\_input\_list\_example\_2)  
#> [1] "ccmpp\_input\_list" "list"

## Validating Input Lists

The same function used above, validate\_ccmpp\_object can be used to validate a list

validate\_ccmpp\_object(ccmpp\_input\_list\_example)  
#>   
#> $pop\_count\_age\_sex\_base  
#> ------------------------------------------------------------  
#> # A 'pop\_count\_age\_sex\_base' with 202 rows.  
#> # dimensions = 'time, sex, age'.  
#> # value\_type = 'count', age\_span = '1', time\_span = '1'.  
#> # 'sex' has levels: female, male.  
#> time\_start sex age\_start time\_span age\_span value  
#> 1950 male 0 1 1 182890  
#> 1950 male 1 . . 174512  
#> 1950 male 2 . . 166560  
#> 1950 male 3 . . 159048  
#> 1950 male 4 . . 151990  
#> 1950 male 5 . . 145402  
#> # ... etc.  
#>   
#> $life\_table\_age\_sex  
#> ------------------------------------------------------------  
#> # A 'life\_table\_age\_sex' with 127,260 rows.  
#> # dimensions = 'indicator, time, sex, age'.  
#> # value\_type = 'real', age\_span = '1', time\_span = '1'.  
#> # 'sex' has levels: female, male.  
#> # 'indicator' has levels: lt\_ex, lt\_lx, lt\_nAx, lt\_ndx, lt\_nLx.....  
#> indicator time\_start sex age\_start time\_span age\_span value  
#> lt\_ex 1950 male 0 1 1 67.12475  
#> lt\_ex 1950 male 1 . . 69.28331  
#> lt\_ex 1950 male 2 . . 68.46299  
#> lt\_ex 1950 male 3 . . 67.60957  
#> lt\_ex 1950 male 4 . . 66.72946  
#> lt\_ex 1950 male 5 . . 65.82817  
#> # ... etc.  
#>   
#> $fert\_rate\_age\_f  
#> ------------------------------------------------------------  
#> # A 'fert\_rate\_age\_f' with 7,070 rows.  
#> # dimensions = 'time, age'.  
#> # value\_type = 'rate', age\_span = '1', time\_span = '1'.  
#> # non\_zero\_fert\_ages = '12, 13, ... 52, 53'.  
#> time\_start age\_start time\_span age\_span value  
#> 1950 0 1 1 [zero]  
#> 1950 1 . . [zero]  
#> 1950 2 . . [zero]  
#> . . . . .  
#> 1950 12 . . 3.7238e-05  
#> 1950 13 . . 0.00048408465  
#> 1950 14 . . 0.00176102842  
#> # ... etc.  
#>   
#> $srb  
#> ------------------------------------------------------------  
#> # A 'srb' with 70 rows.  
#> # dimensions = 'time'.  
#> # value\_type = 'ratio', time\_span = '1'.  
#> time\_start time\_span value  
#> 1950 1 1.058  
#> 1951 . 1.058  
#> 1952 . 1.058  
#> 1953 . 1.058  
#> 1954 . 1.058  
#> 1955 . 1.057  
#> # ... etc.  
#>   
#> $mig\_net\_count\_age\_sex  
#> ------------------------------------------------------------  
#> # A 'mig\_net\_count\_age\_sex' with 14,140 rows.  
#> # dimensions = 'time, sex, age'.  
#> # value\_type = 'count', age\_span = '1', time\_span = '1'.  
#> # 'sex' has levels: female, male.  
#> time\_start sex age\_start time\_span age\_span value  
#> 1950 male 0 1 1 -1906.5913  
#> 1950 male 1 . . -772.8935  
#> 1950 male 2 . . 52.2876  
#> 1950 male 3 . . 765.6268  
#> 1950 male 4 . . 1291.9252  
#> 1950 male 5 . . 1651.7646  
#> # ... etc.  
#>   
#> $mig\_net\_rate\_age\_sex  
#> ------------------------------------------------------------  
#> # A 'mig\_net\_rate\_age\_sex' with 14,140 rows.  
#> # dimensions = 'time, sex, age'.  
#> # value\_type = 'rate', age\_span = '1', time\_span = '1'.  
#> # 'sex' has levels: female, male.  
#> time\_start sex age\_start time\_span age\_span value  
#> 1950 male 0 1 1 -0.0104247982  
#> 1950 male 1 . . -0.0044288845  
#> 1950 male 2 . . 0.0003139265  
#> 1950 male 3 . . 0.0048138097  
#> 1950 male 4 . . 0.0085000671  
#> 1950 male 5 . . 0.0113599854  
#> # ... etc.  
#>   
#> $mig\_net\_count\_tot\_b  
#> ------------------------------------------------------------  
#> # A 'mig\_net\_count\_tot\_b' with 70 rows.  
#> # dimensions = 'time'.  
#> # value\_type = 'count', time\_span = '1'.  
#> time\_start time\_span value  
#> 1950 1 87133.78  
#> 1951 . 99469.67  
#> 1952 . 108079.69  
#> 1953 . 113503.69  
#> 1954 . 116281.54  
#> 1955 . 116953.09  
#> # ... etc.  
#>   
#> $mig\_parameter  
#> ------------------------------------------------------------  
#> # A 'mig\_parameter' with 140 rows.  
#> # dimensions = 'indicator, time'.  
#> # value\_type = 'categorical', time\_span = '1'.  
#> # 'indicator' has levels: mig\_assumption, mig\_type.  
#> # 'value' has levels: counts, end.  
#> indicator time\_start time\_span value  
#> mig\_assumption 1950 1 end  
#> mig\_assumption 1951 . end  
#> mig\_assumption 1952 . end  
#> mig\_assumption 1953 . end  
#> mig\_assumption 1954 . end  
#> mig\_assumption 1955 . end  
#> # ... etc.

# Working with CCMPP Input Classes

In general, ccmpp\_input\_lists and the component data frames will behave just like regular *R* lists and data frames. However, it is useful to note that subsetting a data frame component of a ccmpp\_input\_list will drop the special class and the attributes. This is because a data frame must have exactly the right columns, column names, and structure to be a valid member of the class. Subsetting or otherwise manipulating a data frame will likely make it invalid as a member.

class(fert\_example)  
#> [1] "fert\_rate\_age\_f" "ccmpp\_input\_df"   
#> [3] "demog\_change\_component\_df" "data.frame"

x <- fert\_example[1:5,]  
#> Warning in `[.demog\_change\_component\_df`(fert\_example, 1:5, ): Subsetting a  
#> 'fert\_rate\_age\_f' will not preserve the class or attributes. See '?subset\_time'  
#> and friends for an alternative approach.  
class(x)  
#> [1] "data.frame"  
attributes(x)  
#> $names  
#> [1] "time\_start" "age\_start" "time\_span" "age\_span" "value"   
#>   
#> $row.names  
#> [1] 1 2 3 4 5  
#>   
#> $class  
#> [1] "data.frame"

y <- fert\_example$time\_start  
class(y)  
#> [1] "numeric"  
attributes(x)  
#> $names  
#> [1] "time\_start" "age\_start" "time\_span" "age\_span" "value"   
#>   
#> $row.names  
#> [1] 1 2 3 4 5  
#>   
#> $class  
#> [1] "data.frame"

To subset a data frame there are special functions subset\_[dimension], where “[dimension]” is one of the dimensions in Table @ref(tab:ccmpp-input-subclass-restrictions).

fert\_example\_1950s\_only <- subset\_time(fert\_example, times = 1950:1959)  
#> 'non\_zero\_fert\_ages' set to '12, 13, 14, ... , 51, 52, 53  
summary(fert\_example\_1950s\_only)  
#> dimensions: time, age  
#> time: range = [1950, 1959] span = 1  
#> age: range = [0, 100] span = 1  
#> value\_type: rate  
#> ------------------------------------------------------------  
#> non\_zero\_fert\_ages: 12, 13, 14, ... , 51, 52, 53  
#> ------------------------------------------------------------  
#> table:  
#> time\_start age\_start time\_span age\_span value   
#> Min. :1950 Min. : 0 Min. :1 Min. :1 Min. :0.00000   
#> 1st Qu.:1952 1st Qu.: 25 1st Qu.:1 1st Qu.:1 1st Qu.:0.00000   
#> Median :1954 Median : 50 Median :1 Median :1 Median :0.00000   
#> Mean :1954 Mean : 50 Mean :1 Mean :1 Mean :0.03717   
#> 3rd Qu.:1957 3rd Qu.: 75 3rd Qu.:1 3rd Qu.:1 3rd Qu.:0.03010   
#> Max. :1959 Max. :100 Max. :1 Max. :1 Max. :0.25797

Alternatively, subset the object as normal and recreate it as a member of the class:

### Tidyverse style:  
library(magrittr)  
library(dplyr)  
#>   
#> Attaching package: 'dplyr'  
#> The following objects are masked from 'package:stats':  
#>   
#> filter, lag  
#> The following objects are masked from 'package:base':  
#>   
#> intersect, setdiff, setequal, union  
fert\_example\_1950s\_only\_2 <-  
 dplyr::filter(fert\_example, time\_start < 1960) %>%  
 fert\_rate\_age\_f()  
#> 'non\_zero\_fert\_ages' set to '12, 13, 14, ... , 51, 52, 53  
class(fert\_example\_1950s\_only\_2)  
#> [1] "fert\_rate\_age\_f" "ccmpp\_input\_df"   
#> [3] "demog\_change\_component\_df" "data.frame"  
  
### Base R style:  
fert\_example\_1950s\_only\_3 <-  
 fert\_example[fert\_example$time\_start < 1960, ]  
#> Warning in `[.demog\_change\_component\_df`(fert\_example, fert\_example$time\_start  
#> < : Subsetting a 'fert\_rate\_age\_f' will not preserve the class or attributes.  
#> See '?subset\_time' and friends for an alternative approach.  
fert\_example\_1950s\_only\_3 <- fert\_rate\_age\_f(fert\_example\_1950s\_only\_3)  
#> 'non\_zero\_fert\_ages' set to '12, 13, 14, ... , 51, 52, 53  
class(fert\_example\_1950s\_only\_3)  
#> [1] "fert\_rate\_age\_f" "ccmpp\_input\_df"   
#> [3] "demog\_change\_component\_df" "data.frame"

# What Next?

1. Get feedback on the general set-up, names of things, operation, etc.
2. Decide if the output list would benefit from having classes defined like the input list. This would be straightforward given the infrastructure already created.
3. Complete documentation files for functions and classes (e.g., ?ccmpp\_input\_list).
4. Consider other methods that would be useful. E.g., would a plot method be useful to vizualize the inputs? This would allow quick and easier creation of a standard set of vizualizations to check the inputs/outputs.