Potsdam Institute for Climate Impact Research

Bringing structure into data processing work-flows for MAgPIE

David M Chen, Kristine Karstens,
Miodrag Stevanović, Jan Philipp Dietrich et al.

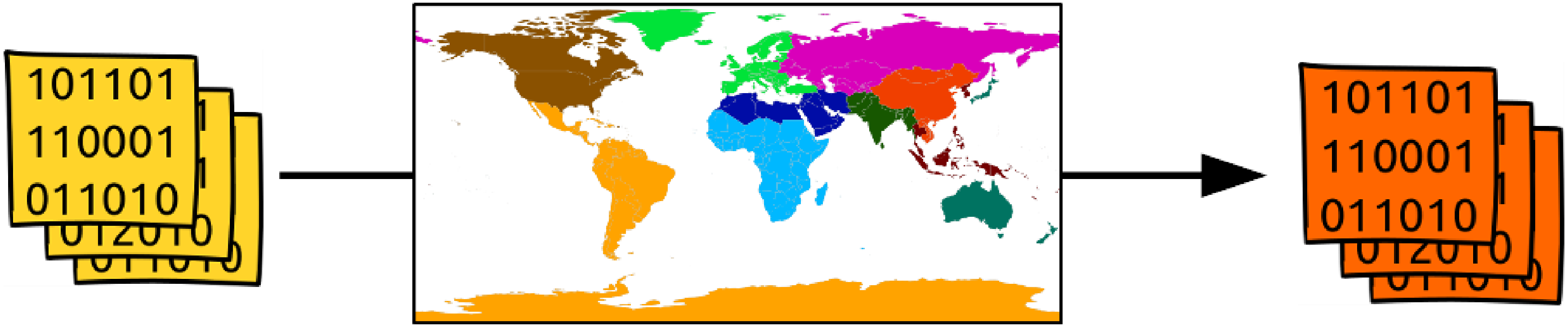
MAgPIE training workshop,MADRAT tutorial

11-12-2020

Member of Leibniz Association

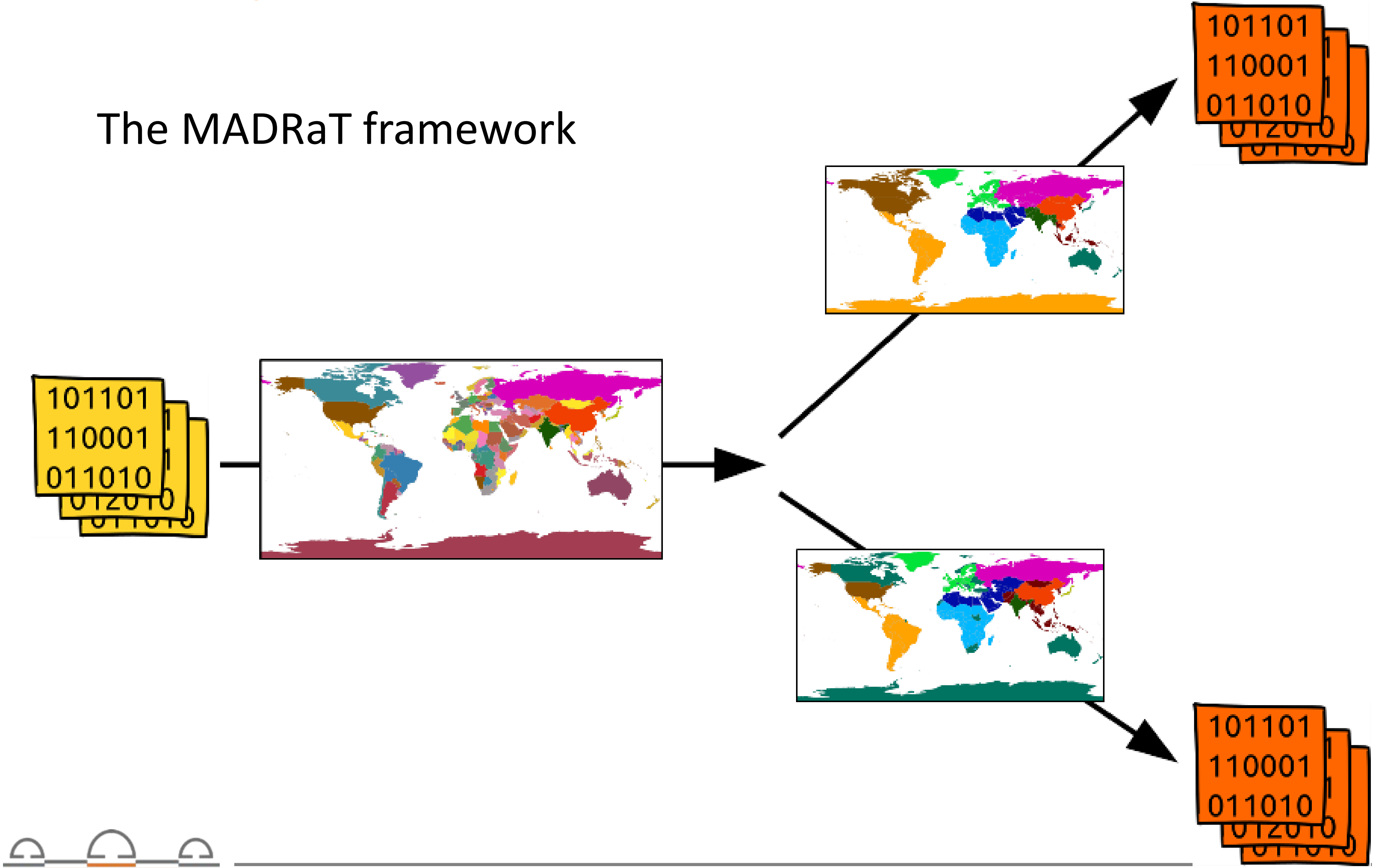
The problem

Preparing input data for the model

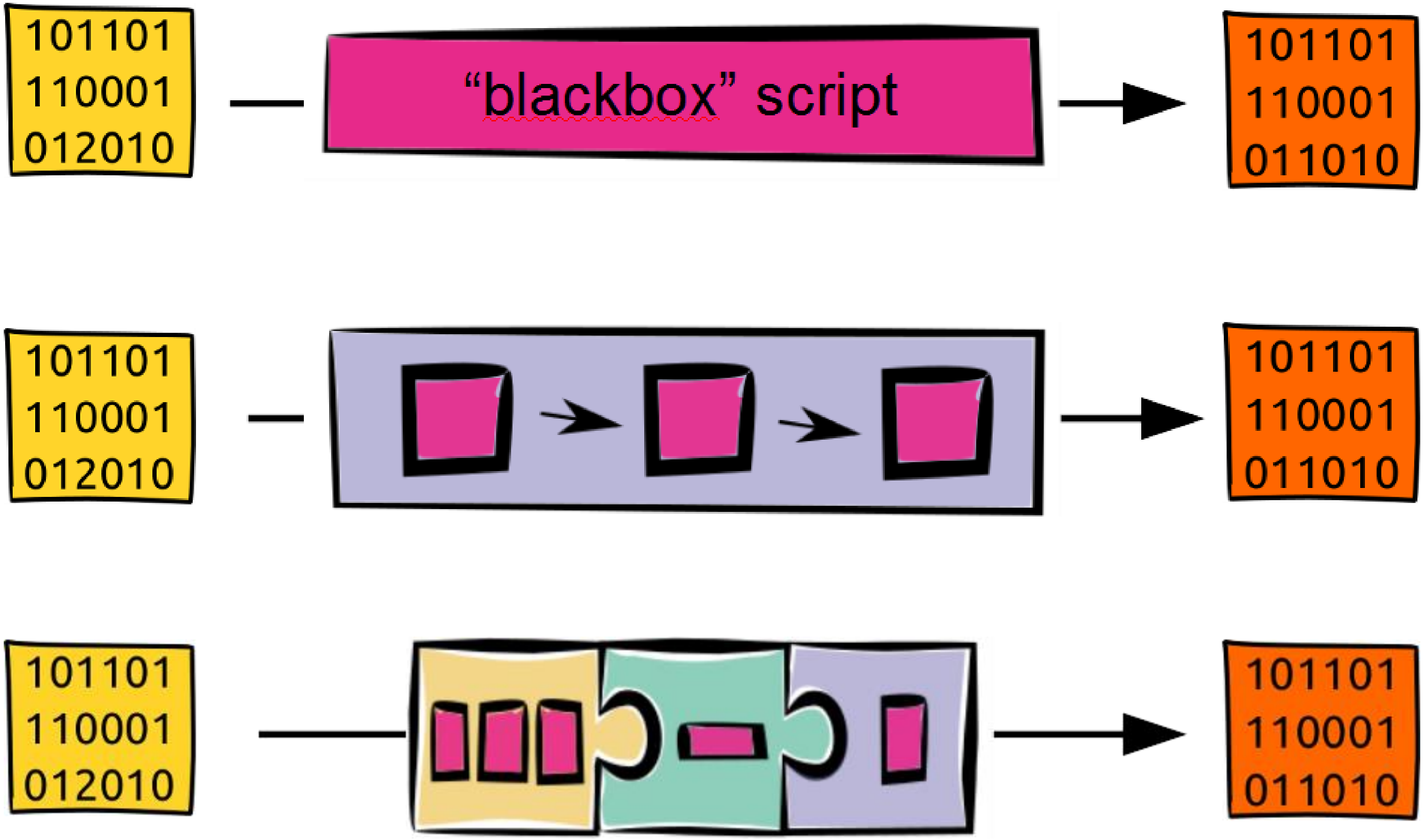


Our attempt to solve it

The MADRaT framework



Our attempt to solve it



The derived framework

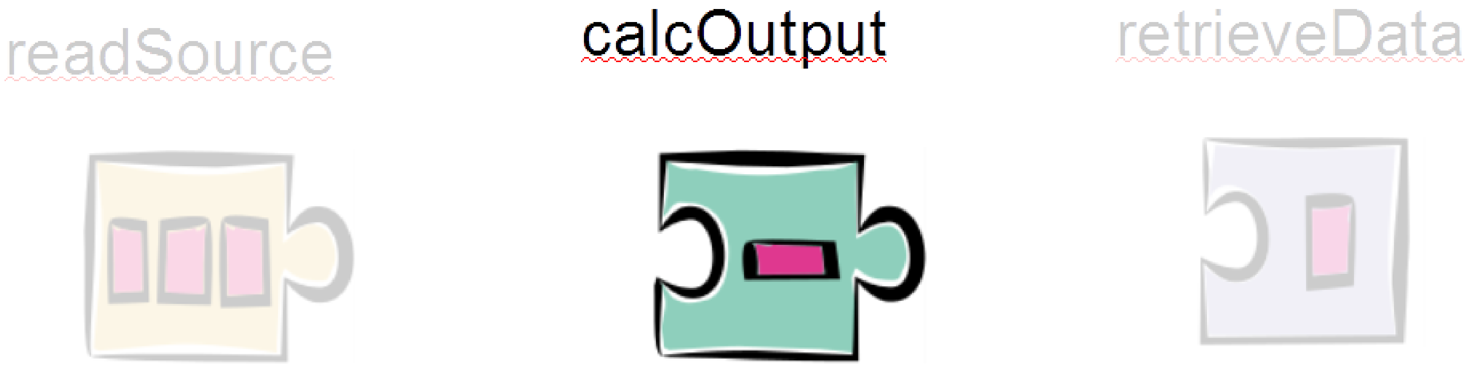
readSource

calcOutput

retrieveData

1. Download data (downloadSource)
2. Read data and convert to standardized data format
3. Bring data to desired regional resolution

The derived framework



readSource

calcOutput

retrieveData

1. Calculate required data
   1. Filtering of data
   2. Merging of data from different data sources
   3. Data harmonization
2. Provide spatial aggregation (e.g. weights)

The derived framework

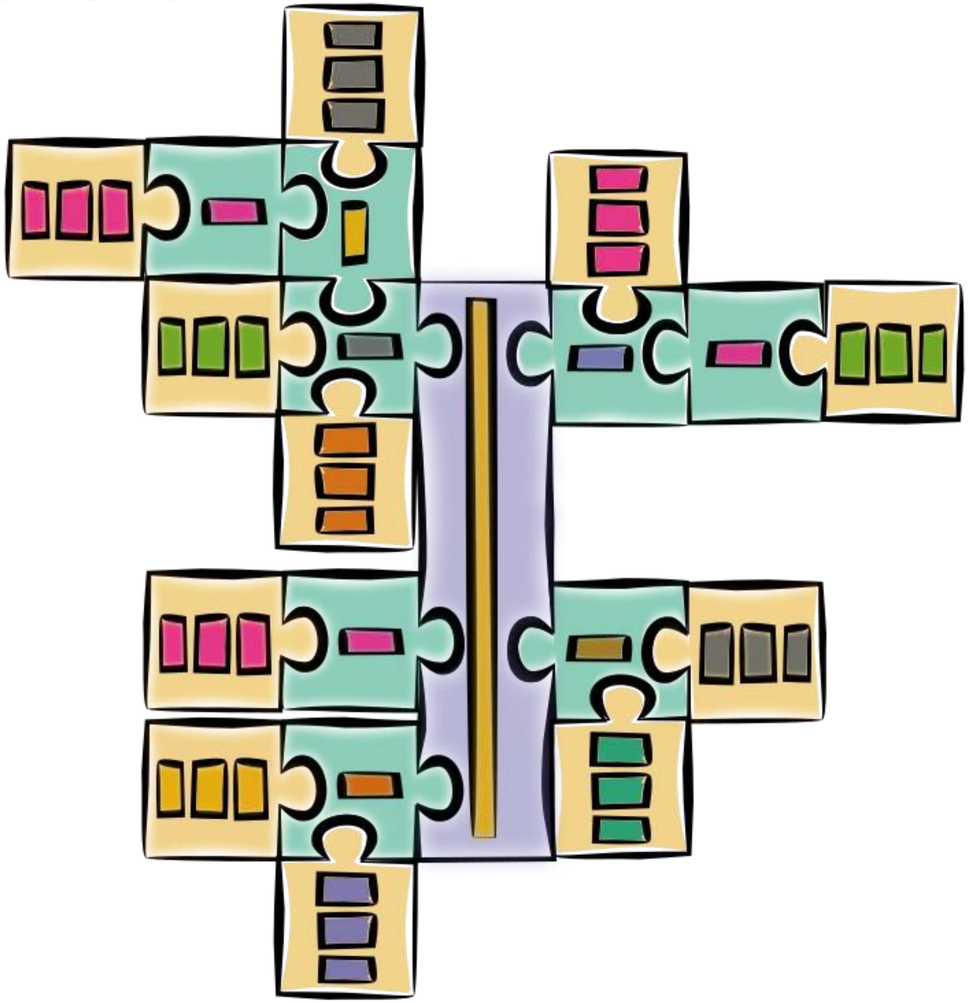
readSource

calcOutput

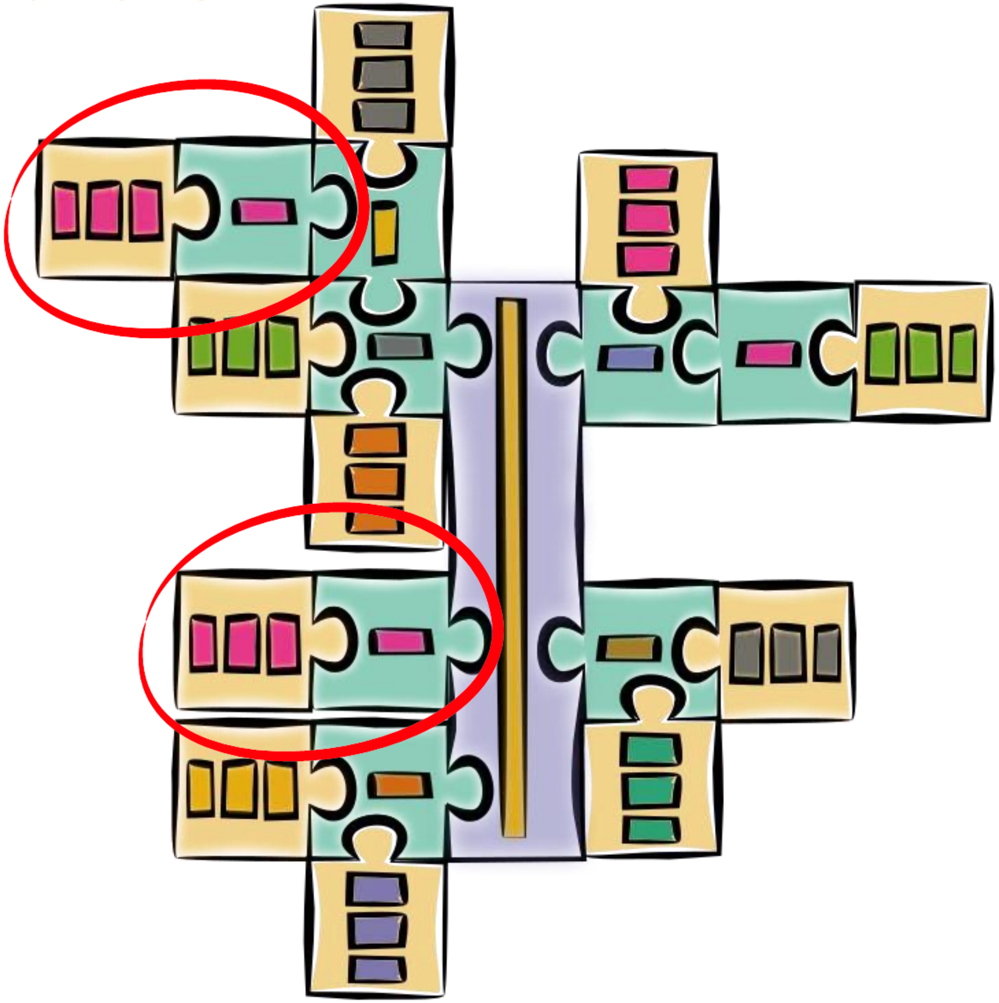
retrieveData

1. Collecting data sets
2. Coordinate packaging of aggregated data

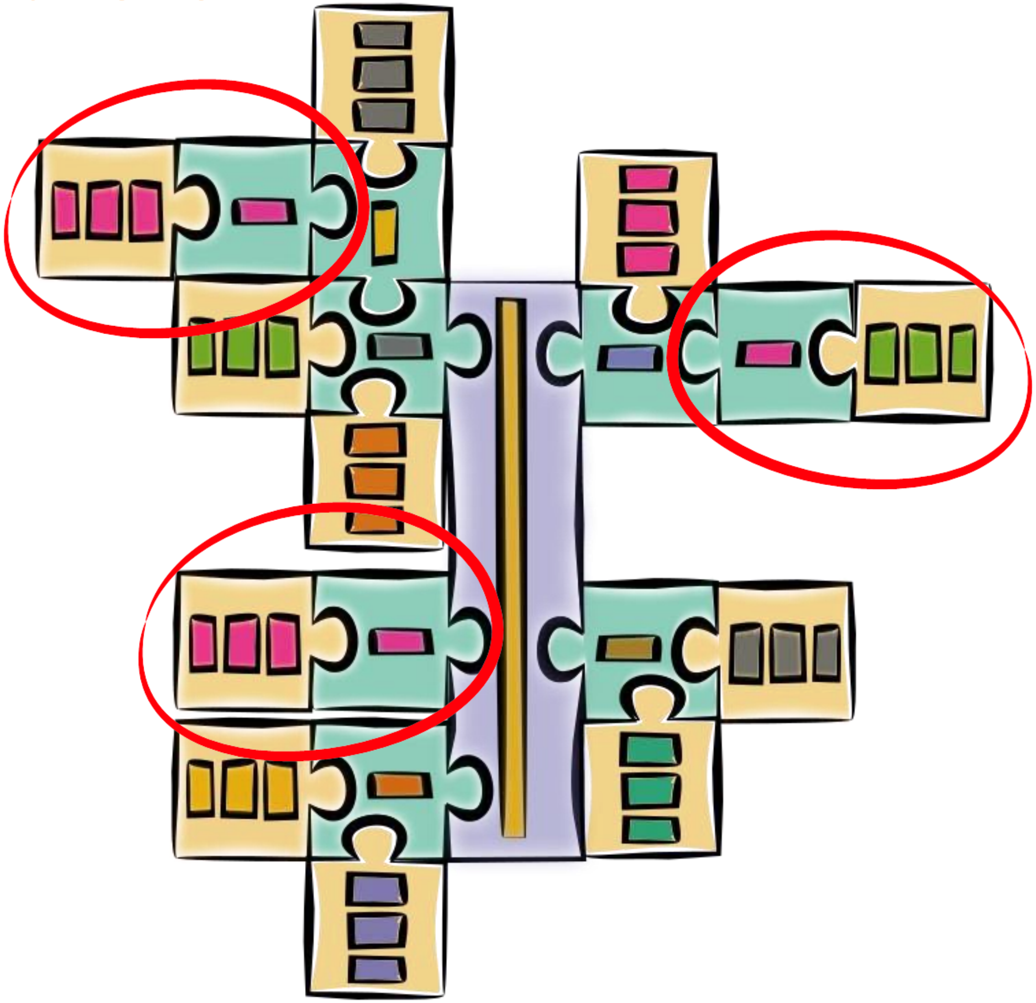
The derived framework



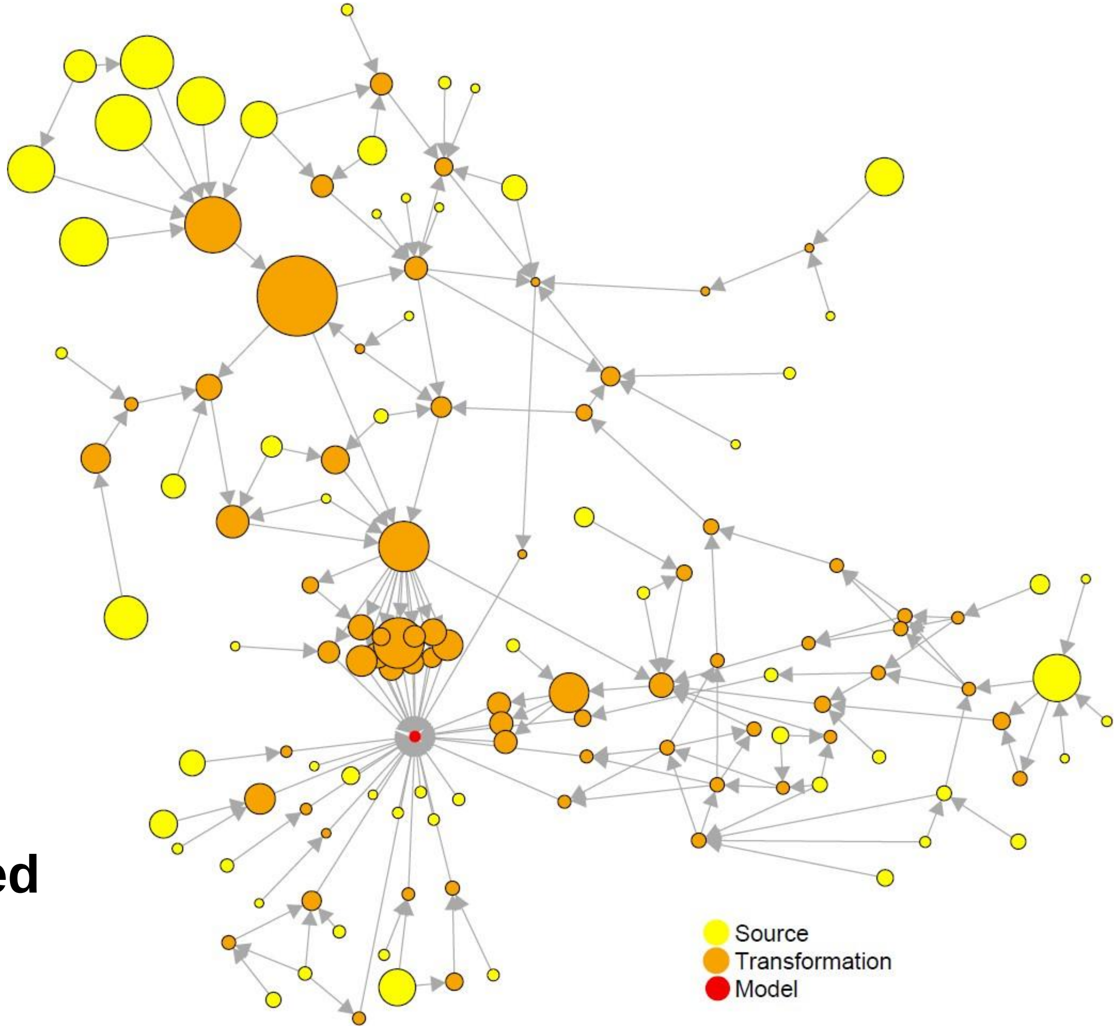
The derived framework



The derived framework



Unanticipated side effects



- **A lot of low hanging fruits:**
  - Meta-data generation
  - Sanity checks
  - Data processing networks
  - Data caching
  - Structured log file

- **Users report faster development**

- **Broader usage than planned**

- **Change in focus:**
  - **Spatial aggregation → reproducibility and transparency**

MADRaT

“May All Data be Reproducible and Transparent”

- R package
- License: BSD2
- Git: https://github.com/pik-piam/madrat
- CRAN: https://CRAN.R-project.org/package=madrat

- Contact: dietrich@pik-potsdam.de

Magclass Objects

Basic element in MADRaT: MAgPIE objects

Array consisting of 3 dimensions:

1st dim: Spatial

2nd dim: Temporal

3rd dim: Data

Magclass Objects

Basic element in MADRaT: MAgPIE objects

Array consisting of:

3 dimensions (Subdimensions possible):

1st dim: Spatial

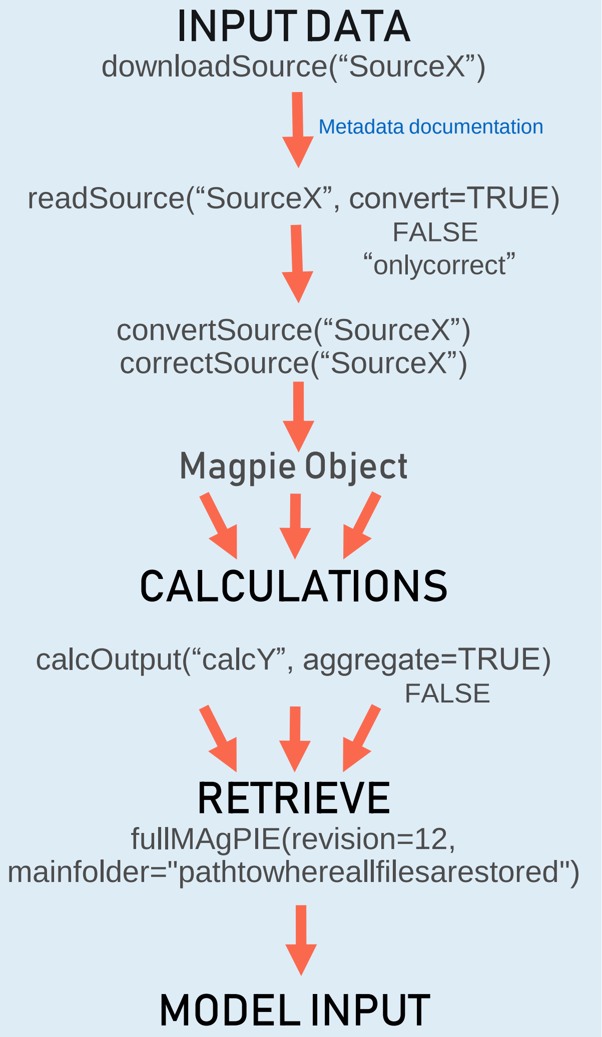
2nd dim: Temporal

3rd dim: Data

MADRaT Cheat Sheet

library(madrat)

MADRaT Workflow



INPUT DATA

downloadSource(“SourceX”)

Metadata documentation

readSource(“SourceX”, convert=TRUE)
FALSE
“onlycorrect”

convertSource(“SourceX”)
correctSource(“SourceX”)

Magpie Object

CALCULATIONS

calcOutput(“calcY”, aggregate=TRUE)
FALSE

RETRIEVE

fullMAgPIE(revision=12,
mainfolder="pathtowhereallfilesarestored")

MODEL INPUT

Magpie Objects

Array with 3 Dimensions

1: Spatial	2:Temporal	3: Data
Cellular **59199 cells**	Years **1965-2150**	Subdimensions concatenated with “.”
Country **249 ISO3**	Call with: char “y1965” OR int 1965	Avoid using “.” in naming
Region **12 Magpie Regions**		

MADRaT Config

```
## See config settings
library(madrat)
getConfig()

## Turn Cache on
setConfig(forcecahe=TRUE)
  # NOTE:  Running a function with cache on and an existing cache file means further developments will not appear in results ##

## Get Mappings folder
getConfig(“mappingfolder”)

## Change region mapping
setConfig(regionmapping=“new_mapping.csv”)
```

Link a Package to MADRaT

Save the code below as madrat.R in R folder of package

```
#' @importFrom madrat vcat toolCodeLabels
#' @importFrom digest digest

.onLoad <- function(libname, pkgname){
  madrat::setConfig(packages=c(madrat::getConfig("packages"),pkgname), .cfgchecks=FALSE,
.verbose=FALSE)

  # add labels for common ctype selections
  labels <- NULL
  for(t in c("c","n","h")) {
    ncells <- c(seq(10,90,10),seq(100,900,100),seq(1000,10000,1000))
    for(n in ncells) {
      tmp <- paste0(t,n)
      labels[tmp] <- digest::digest(list(ctype=tmp),"md5")
    }
  }
  toolCodeLabels(add=labels)
}
#create an own warning function which redirects calls to vcat (package internal)
warning <- function(...) vcat(0,...)
# create a own stop function which redirects calls to stop (package internal)
stop <- function(...) vcat(-1,...)
# create an own cat function which redirects calls to cat (package internal)
cat <- function(...) vcat(1,...)
```

Magclass Basics

Subset: mag[subset,,]

Avoid: mag[subset]

Useful magclass Functions

Further documentation in ?magclass::function()

as.magpie()	Converts dataframe to magclass
getItems()	List of all dimension names
getRegions()	Vector of object regions
getYears()	Vector of years as char or int class
getNames()	Vector of names of data

Spatial	
toolCountryFill()	Fills in/matches incomplete country dimension with NA / given value
toolAggregate()	Weighted aggregation, mapping file needed
toolCountry2isocode	Converts country names to ISO3 code
Temporal	
time_interpolate()	Linearly interpolates values between years
toolHoldConstant()	Hold values constant for given years
toolHoldConstantBeyondEnd()	Extend magpie object to 2150, holding missing years constant
Data Analysis	
mbind()	bind 2 magpie objects along a dim, like abind
add_columns()	Add new column to a given dimension “dim”
add_dimension()	Add new dimension, with name of first column in new dim
calibrate_it()	Calibrate one dataset to another over time, using set functions
dimOrder()	Re-order dimensions
dimSums	**Very useful! Sum over dims and sub-dimensions**
magpply()	Like apply family of functions, to replace loops
read.magpie()	Read magpie .mz files
write.magpie()	write a magpie object ot file, various file formats incl. ncdf4

Build a MADRaT-linked library

#run

lucode2::buildLibrary()

#Need 0 Errors, warnings, notes before commit

• CC BY SA David M Chen • david.chen@pik-potsdam.de • https://github.com/magpiemodel/tutorials • package version 1.83.3 • Updated: 12.2020

Magclass Exercise

load madrat in R via library(madrat)

population_magpie is automatically loaded by madrat

Assign it to pop by

```
pop <- population_magpie
```

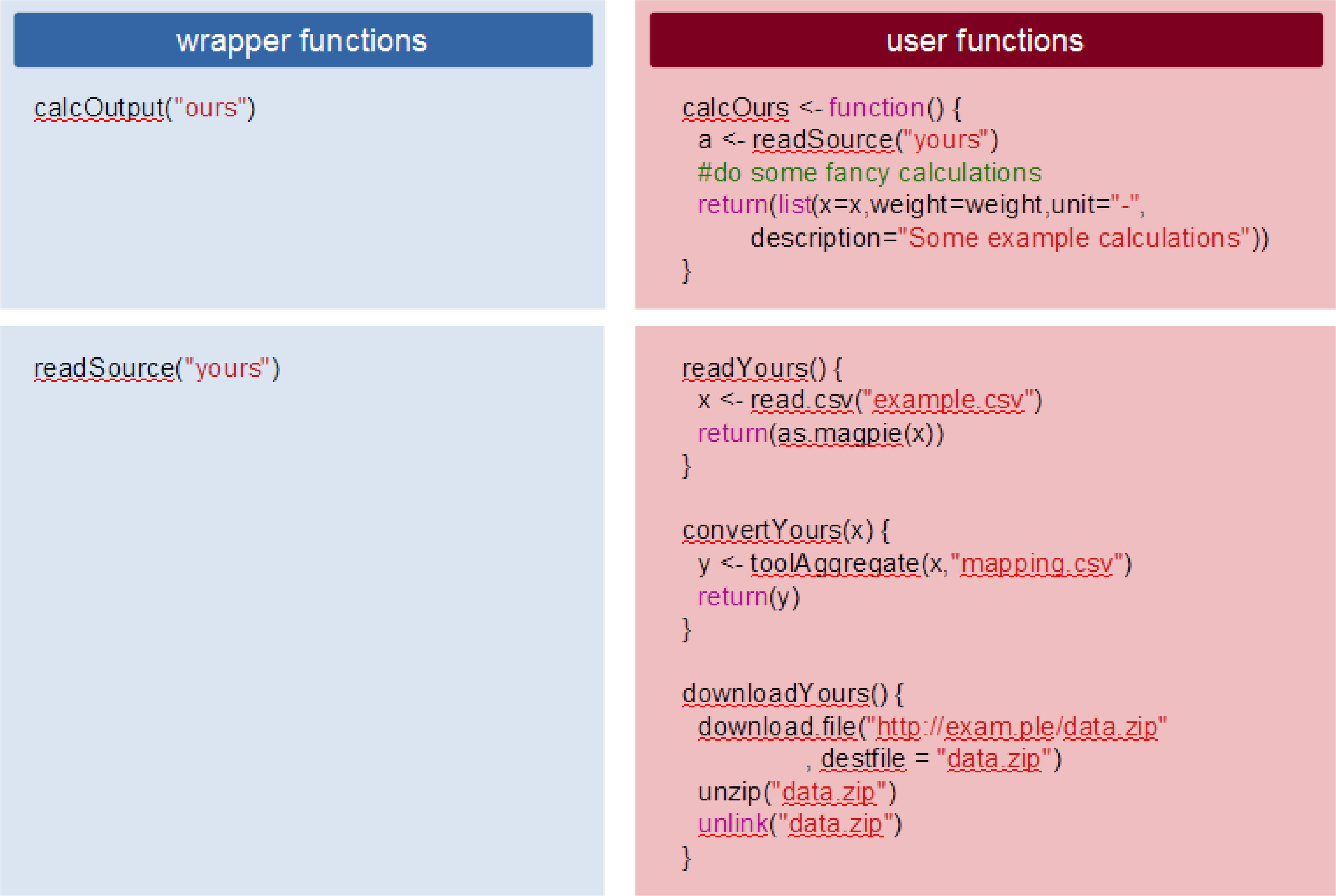
Using magclass functions, answer these questions:

1. What is the global population in 2100 for scenario A2? B1?

2. How does is the population of Sub-Saharan Africa (AFR) as share of global total change over the years?

3. Get population values for the years 2046-2049 by linearly interpolating between 2045 and 2050 values.

Backup Slides



wrapper functions

```
calcOutput("ours")
```

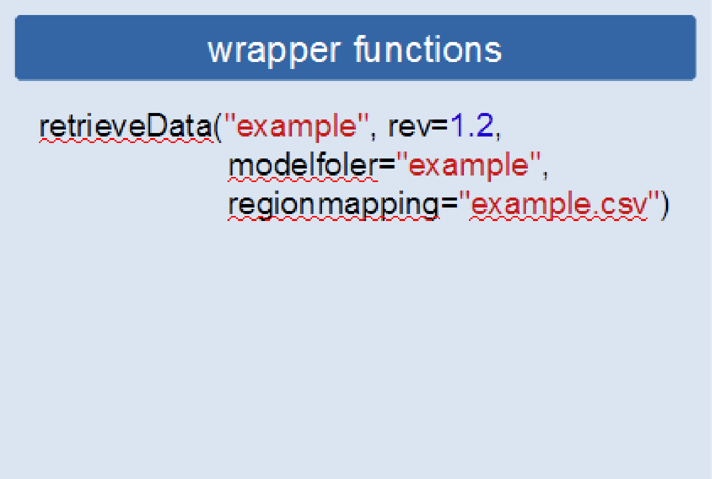
```
readSource("yours")
```

user functions

```
calcOurs <- function() {
  a <- readSource("yours")
  #do some fancy calculations
  return(list(x=x,weight=weight,unit="-",
        description="Some example calculations"))
}
```

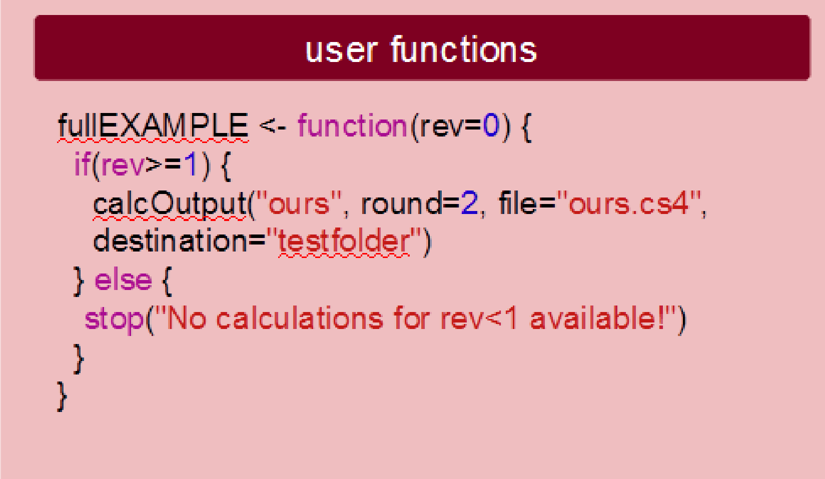
```
readYours() {
  x <- read.csv("example.csv")
  return(as.magpie(x))
}

convertYours(x) {
  y <- toolAggregate(x,"mapping.csv")
  return(y)
}

downloadYours() {
  download.file("http://exam.ple/data.zip"
               , destfile = "data.zip")
  unzip("data.zip")
  unlink("data.zip")
}
```

Backup Slides



wrapper functions

```
retrieveData("example", rev=1.2,
             modelfoler="example",
             regionmapping="example.csv")
```



user functions

```
fullEXAMPLE <- function(rev=0) {
  if(rev>=1) {
    calcOutput("ours", round=2, file="ours.cs4",
    destination="testfolder")
  } else {
   stop("No calculations for rev<1 available!")
  }
}
```

MADRaT Workshop

MADRaT Workshop - Software requirements

- R
  - https://www.r-project.org/
  - https://ftp.gwdg.de/pub/misc/cran/
- Rstudio
  - https://www.rstudio.com/products/rstudio/download/

- Libraries:

```
› install.packages("madrat")
› install.packages("magclass")
```

MADRaT Workshop – Setup

Load library and configure the madrat mainfolder:

```
> library(madrat)
> getConfig()

# Initialize madrat config with default settings..
# madrat mainfolder for data storage not set! Do you want to set it now? (y/n)
> y
# Please enter main folder path: “~/inputdata”
# Directory does not exist. Should it be created? (y/n)
> y
# Should this path be added to your global .Rprofile to be used permanently? (y/n)
> y
```

MADRaT components: downloadSource()

Download the source data by using the ***wrapper*** function:

```
> downloadSource("Tau", overwrite = TRUE)
```

```
>madrat:::downloadTau
# function ()
# {
# download.file("http://www.pik-potsdam.de/members/dietrich/tau-data.zip",
# destfile = "tau-data.zip")
# unzip("tau-data.zip")
# unlink("tau-data.zip")
# }
# <environment: namespace:madrat>
```

MADRaT components: `readSource()` I/III

Read the data available in the source.

```
> x <- readSource(type="Tau", subtype="paper", convert=FALSE)
```

Three steps, i.e. three ***wrapper*** functions:

1. `readSource()`
    - reads the data in as a magclass object
2. `correctSource()`
    - (optional) removes duplicates, replacing NAs etc.
3. `convertSource()`
    - compatibility conversion for flexible aggregation (ISO country standard).

MADRaT components: `readSource()` II/III

Develop the `readSrouce()` type function:

```
> madrat:::readTau
# function(subtype = "paper")
# {
# files <- c(paper = "tau_data_1995-2000.mz",
             historical = "tau_xref_history_country.mz")
# file <- toolSubtypeSelect(subtype, files)
# x <- read.magpie(file)
# x[x == -999] <- NA
# return(x)
# }
# <environment: namespace:madrat>
```

- Read-in the data as a magclass object.
- No other modifications are allowed.

Develop the `correctSource()`, in particular `correctTau()` function, if needed.

MADRaT components: `readSource()` III/III

Lastly, develop the `convertSrouce()` type function:

```
>madrat:::convertTau
# function (x)
# {
# tau <- x[, , "tau"]
# xref <- x[, , "xref"]
# xref[is.na(tau) | is.nan(tau)] <- 10^-10
# tau[is.na(tau) | is.nan(tau)] <- 1
# if (ncells(x) == 59199) {
# iso_cell <- sysdata$iso_cell
# iso_cell[, 2] <- getCells(x)
# tau <- toolAggregate(tau, rel = iso_cell, weight = collapseNames(xref))
# xref <- toolAggregate(xref, rel = iso_cell)
# }
# tau <- toolCountryFill(tau, fill = 1, TLS = "IDN", HKG = "CHN",
# SGP = "CHN", BHR = "QAT")
# xref <- toolCountryFill(xref, fill = 0, verbosity = 2)
# return(mbind(tau, xref))
# }
# <environment: namespace:madrat>
```

- Fill out the missing ISO-country data: `toolCountryFill()`

MADRaT components: `calcOutput()`

Extract information form a given source of data.

```
> x <- calcOutput(“TauTotal”, aggregate=FALSE, supplementary=FALSE)

>madrat:::calcTauTotal
# function ()
# {
# tau <- readSource("Tau", "paper")
# x <- collapseNames(tau[, , "tau.total"])
# weight <- collapseNames(tau[, , "xref.total"])
# return(list(x = x, weight = weight, min = 0, max = 10, unit = "1",
# description = "Agricultural Land Use Intensity Tau",
# note = c("data based on Dietrich J.P., Schmitz C., Müller C., Fader M.,
# Lotze-Campen H., Popp "Measuring agricultural land-use intensity - A global
# analysis using a model-assisted approach", "Ecological Modelling, Volume 232,
# 10 May 2012, Pages 109-118, ISSN 0304-3800, 10.1016/j."preprint
                                        .
                                        .
                                        .
# doi = "10.1016/j.ecolmodel.2012.03.002")))
# }
# <environment: namespace:madrat>
```

MADRaT components: `retrieveData()`

Prepare a dataset from a collection of data.

```
> retrieveData("example", rev=1)

>madrat:::fullEXAMPLE
# function (rev = 0)
# {
# writeLines("This is a test", paste0(getConfig("outputfolder"),
# "/test.txt"))
# file2destination("test.txt", "testfolder")
# if (rev >= 1) {
# calcOutput("TauTotal", years = 1995, round = 2, file = "fm_tau1995.cs4",
# destination = "testfolder/input")
# }
# }
# <environment: namespace:madrat>
```

- Creates a log file
- Creates a tgz packaged compressed data
- Puts the data in the “`output`” directory in the defined madrat mainfolder.

Use own functions with MADRaT

Source your own function in the global environment `setConfig(globalenv=TRUE):`

```
> library(madrat)
# add global environment to madrat search path
> setConfig(globalenv=TRUE)
# define simple calc-function
> calcPi <- function() {
> out <- toolCountryFill(NULL,fill=pi)
> return(list(x=out,
         weight=out,
         unit="1",
         description="Just pi"))
> }

# rund calcPi through wrapper function calcOutput
> calcOutput("Pi")
```

- same procedure also for all other MADRaT functions: `downloadXYZ`, `readXYZ`, `correctXYZ`, `convertXYZ` and `fullXYZ`.

Advanced: Create MADRaT-based R-package

The following lines of code should be added as `madrat.R` to the R folder of the package:

```
### madrat.R
#' @importFrom madrat vcat
> .onLoad <- function(libname, pkgname){
> madrat::setConfig(packages=c(madrat::getConfig("packages"),pkgname),
                   .cfgchecks=FALSE, .verbose=FALSE)
> }
#create an own warning function which redirects calls to vcat (package internal)
> warning <- function(...) vcat(0,...)
# create a own stop function which redirects calls to  stop (package internal)
> stop <- function(...) vcat(-1,...)
# create an own cat function which redirects calls to  cat (package internal)
> cat <- function(...) vcat(1,...)
```

- `.onLoad` - the package is linked to madrat as soon as it is loaded.