POTSDAM INSTITUTE FOR CLIMATE IMPACT RESEARCH

Bringing structure into data processing work-flows for MAgPIE

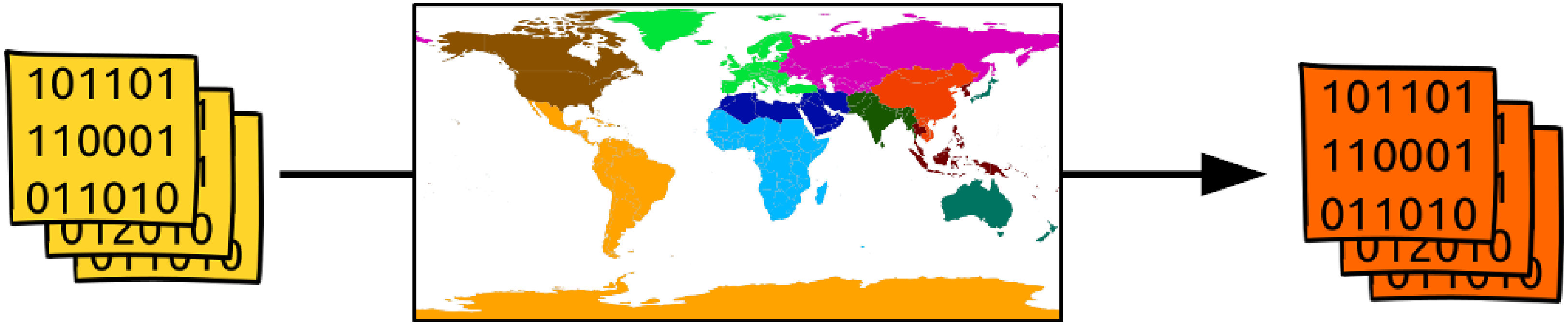
Miodrag Stevanović, Jan Philipp Dietrich et al.

MAgPIE training workshop, PIK, Potsdam

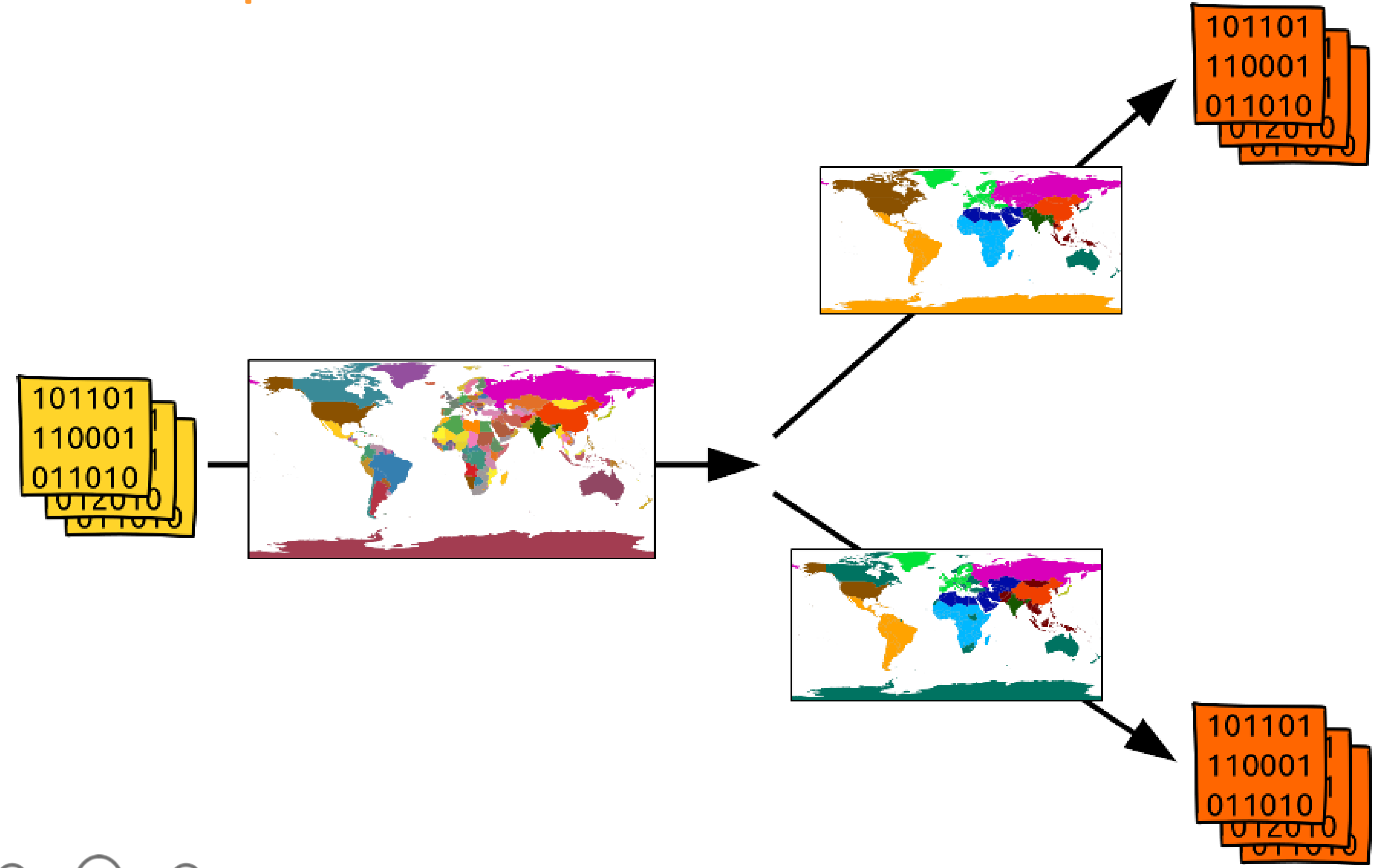
09&10-09-2019

Member of Leibniz Association

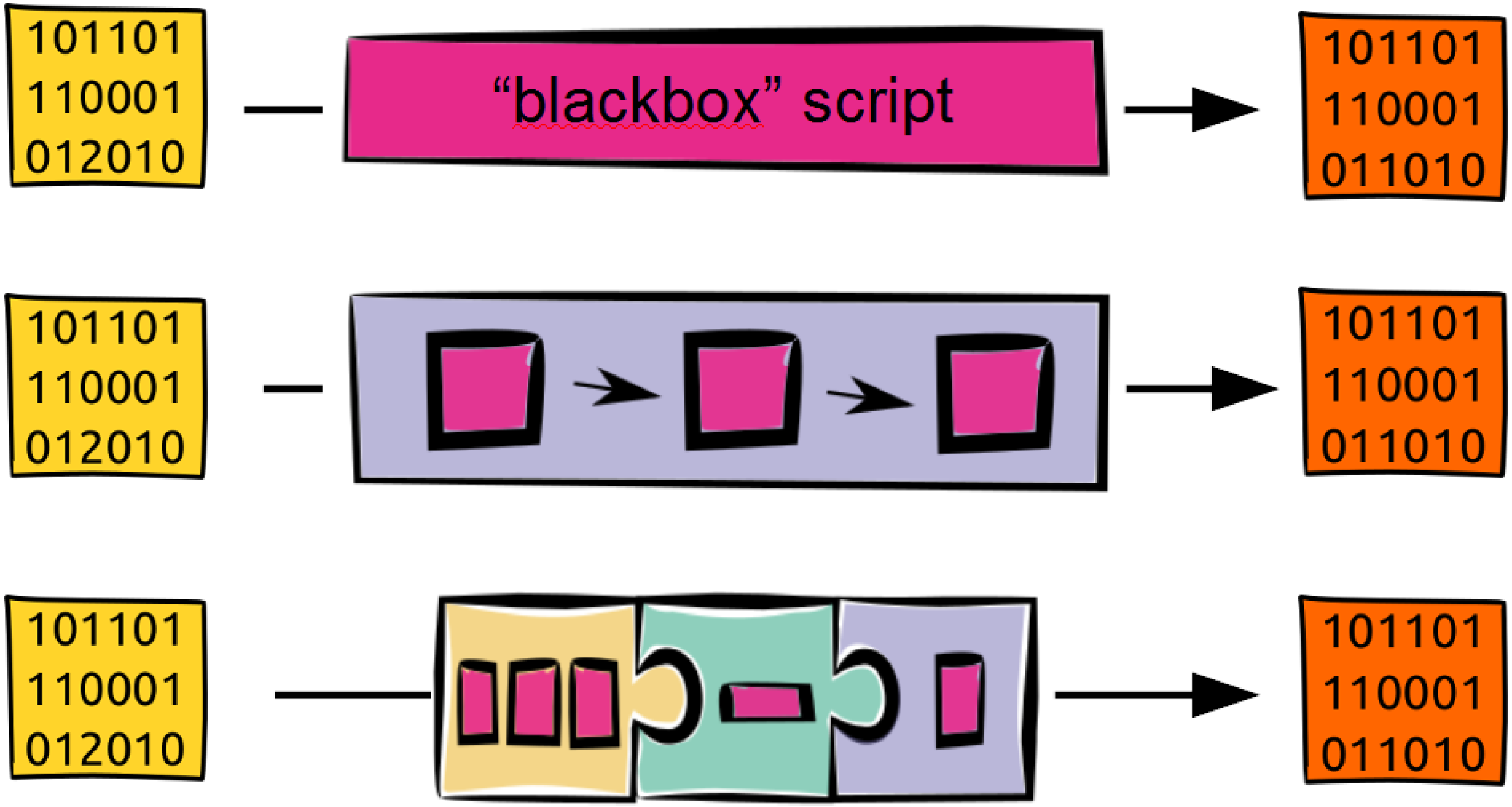
The problem



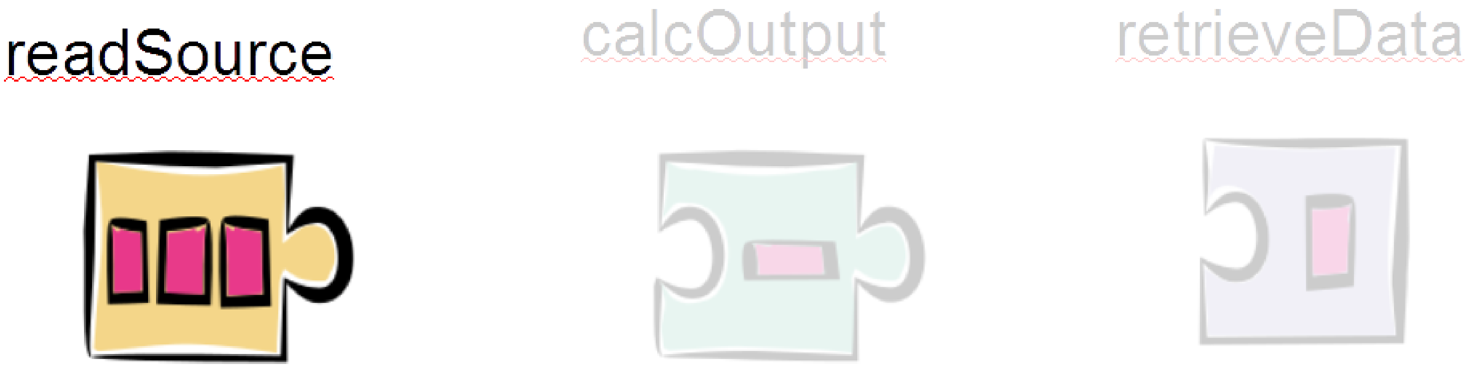
Our attempt to solve it



Our attempt to solve it

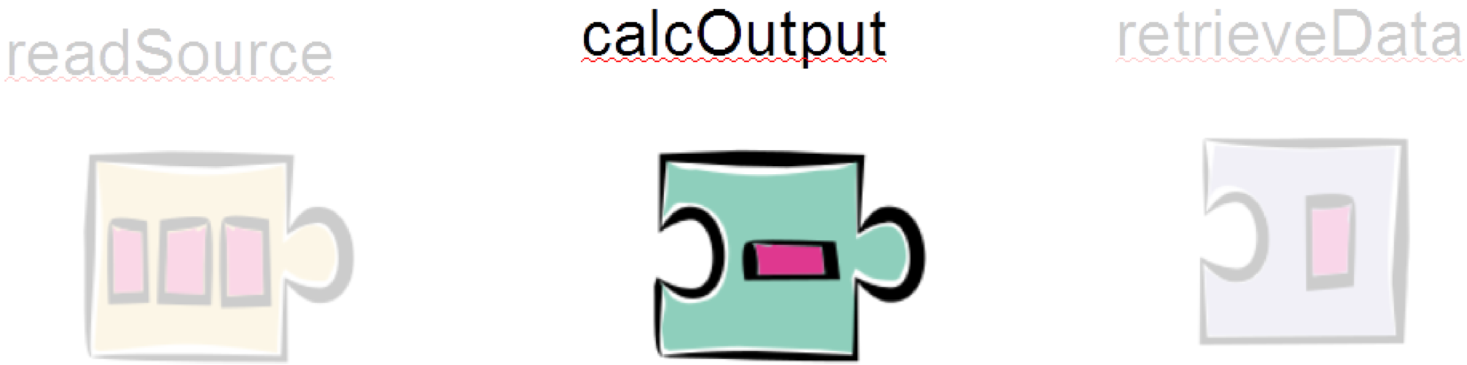


The derived framework



1. Download data
2. Read data and convert to standardized data format
3. Bring data to country-resolution

The derived framework



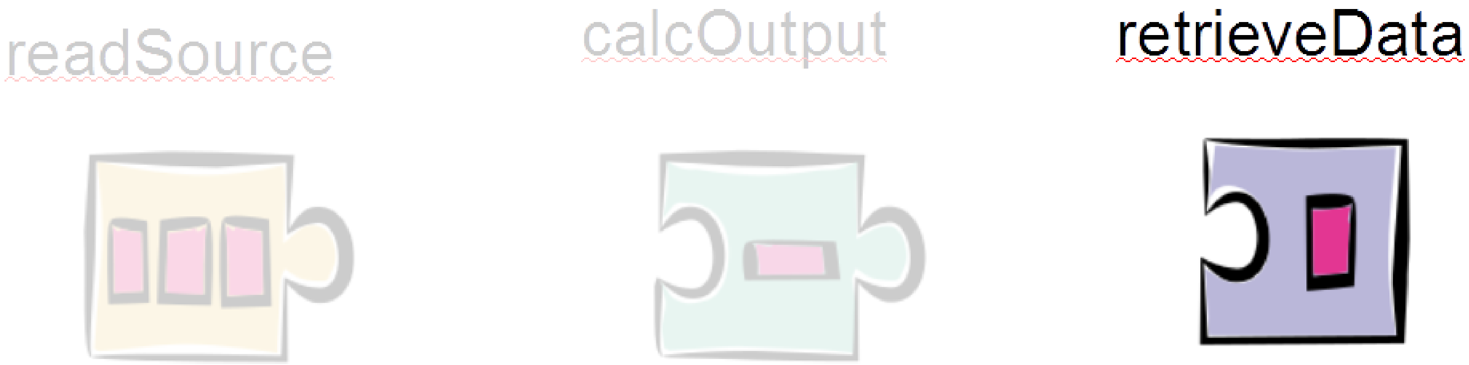
readSource

calcOutput

retrieveData

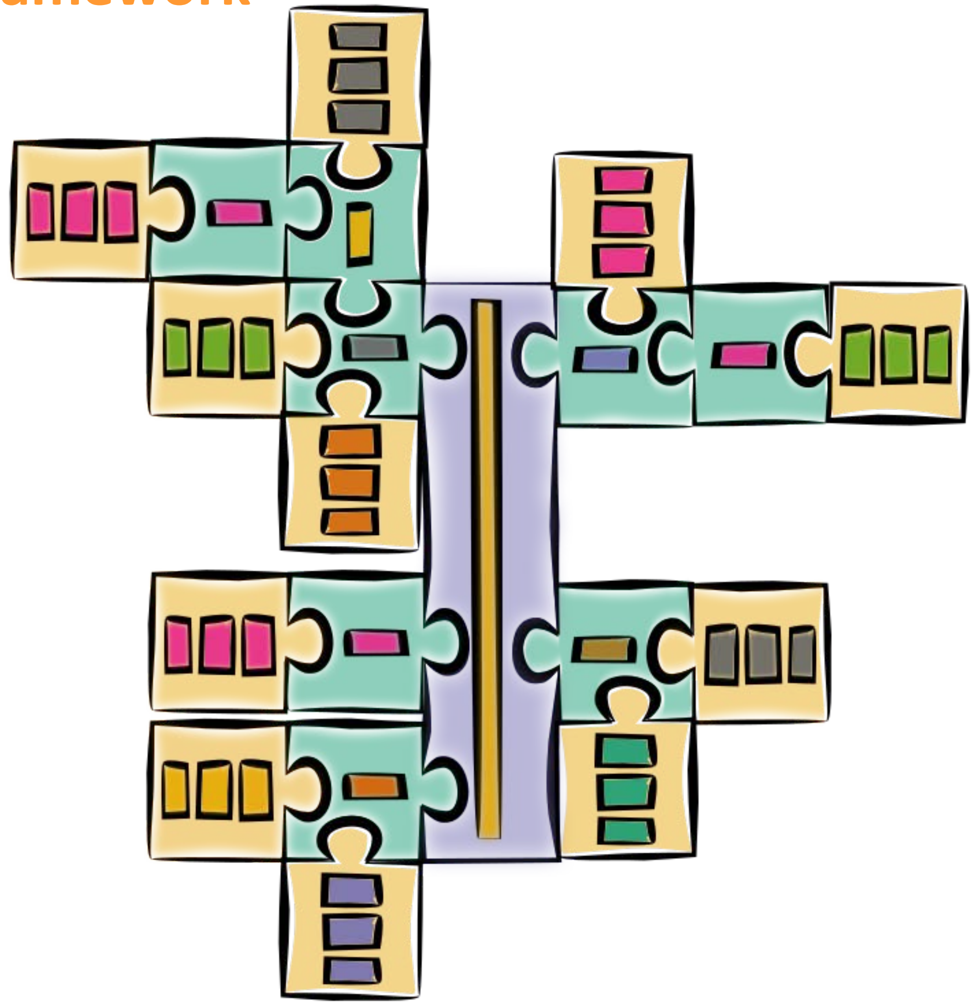
1. Calculate required data
   1. Filtering of data
   2. Merging of data from different data sources
   3. Data harmonization
2. Provide spatial aggregation (e.g. weights)

The derived framework

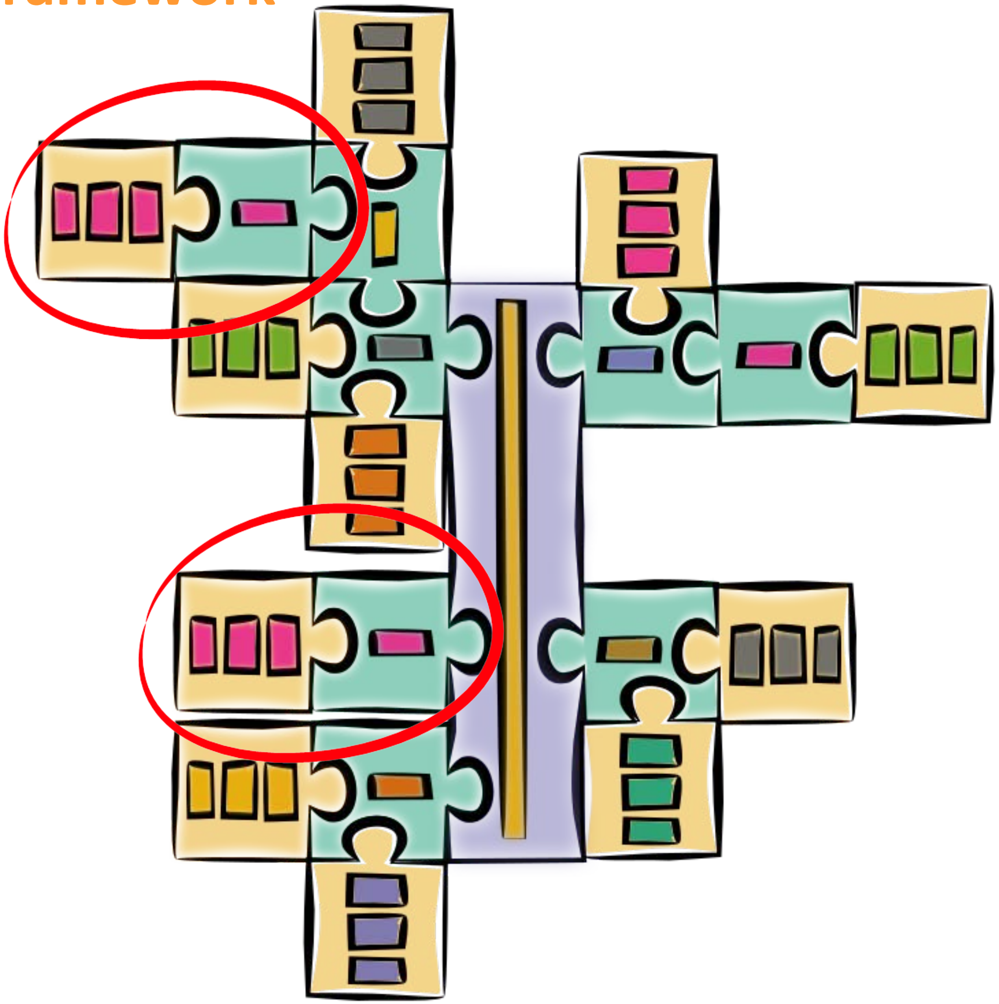


1. Collecting data sets
2. Coordinate packaging of aggregated data

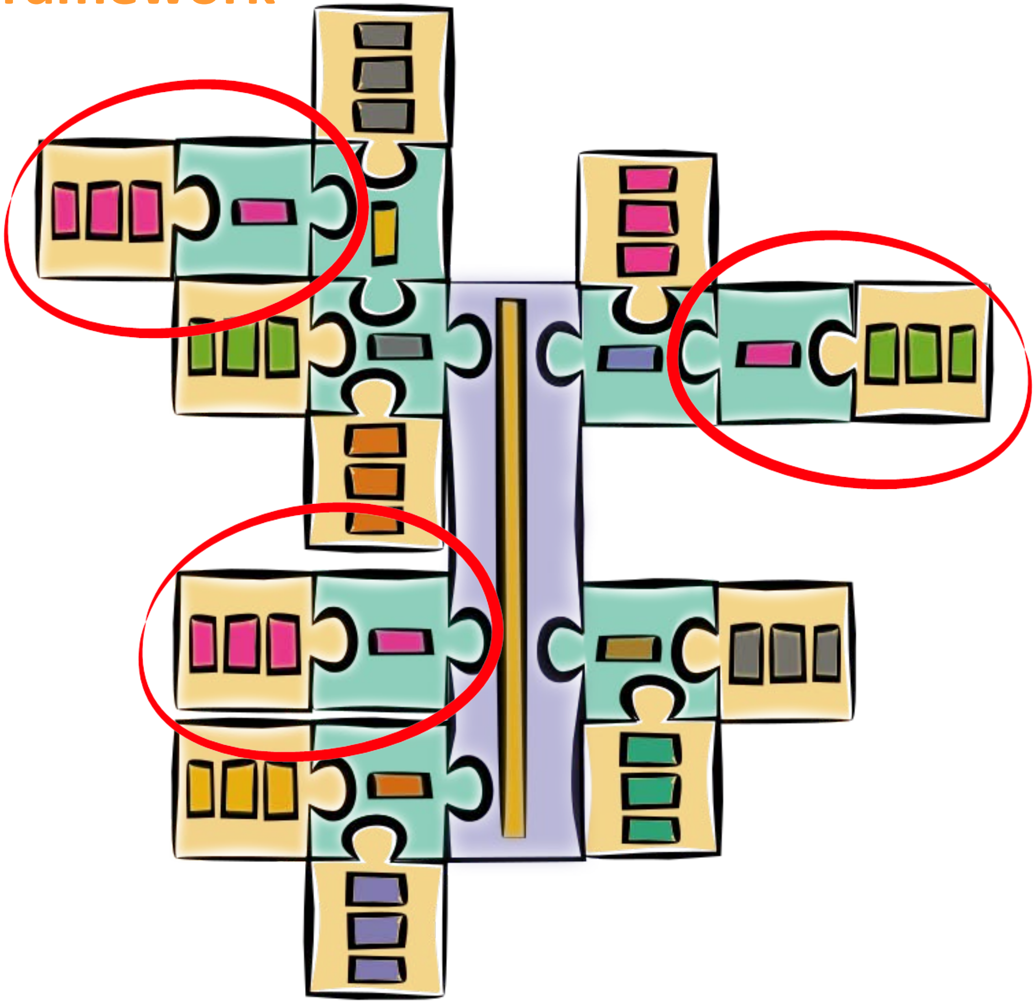
The derived framework



The derived framework

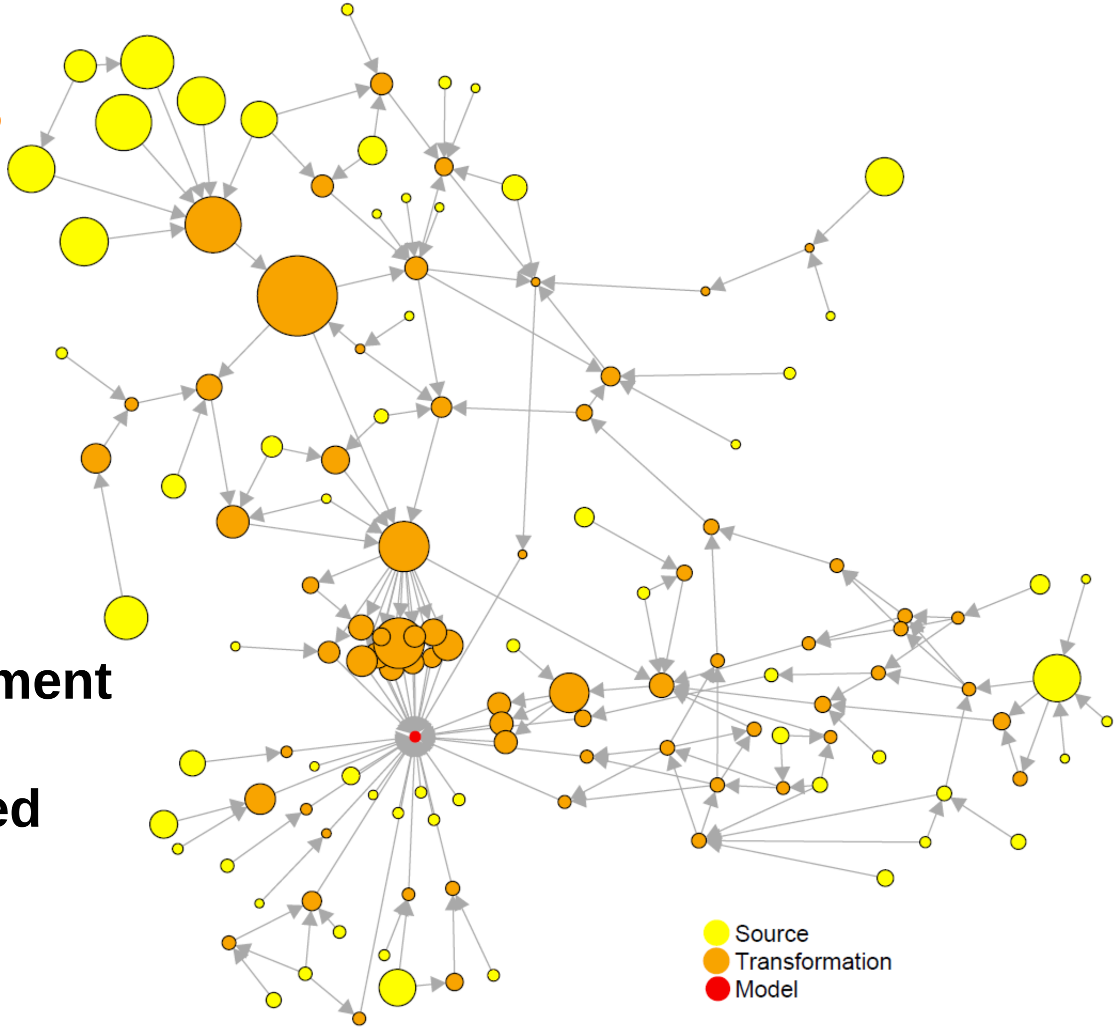


The derived framework



Unanticipated side effects

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



- **User report faster development**

- **Broader usage than planned**

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

MADRaT

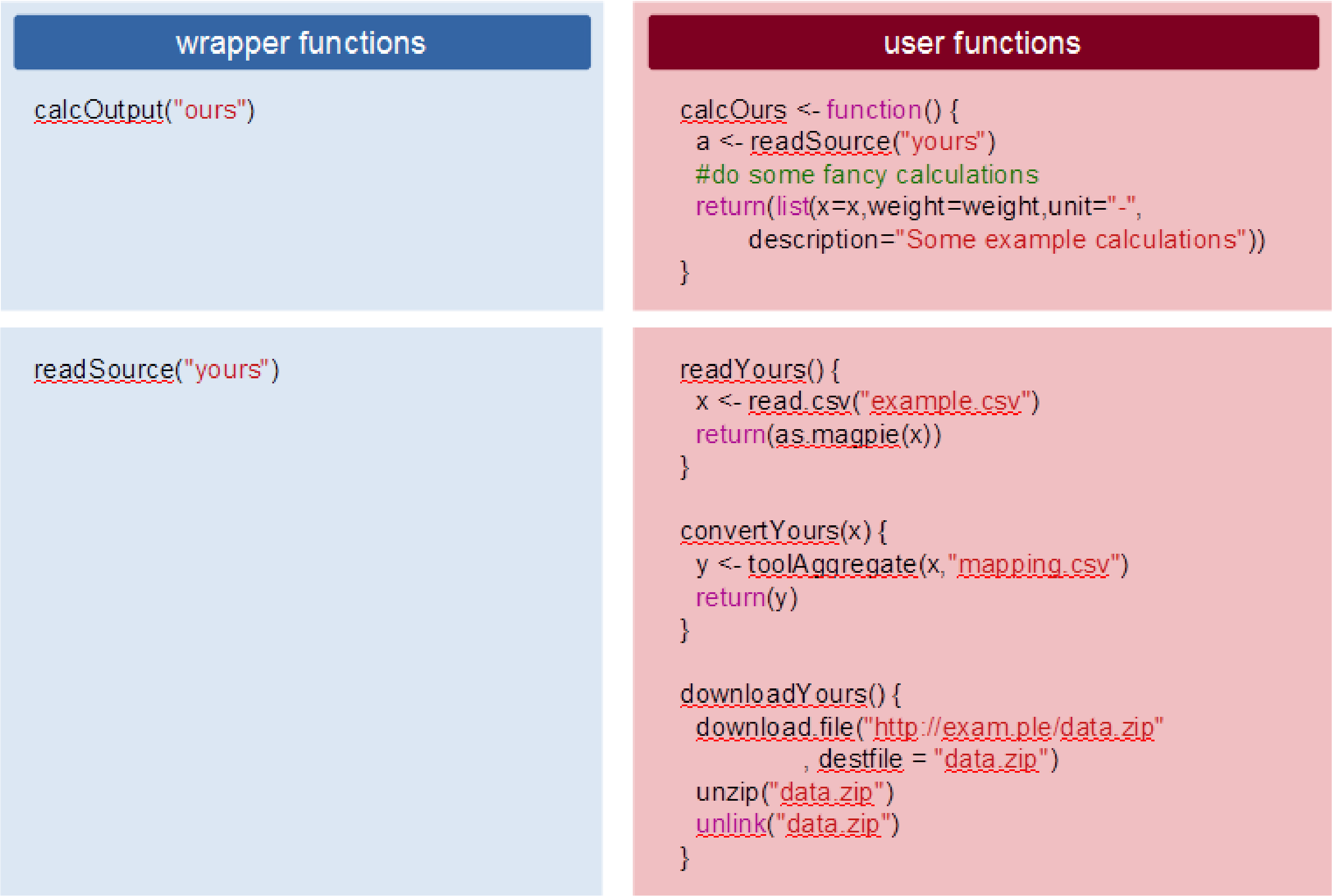
“May All Data be Reproducible and Transparent”

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

- Contact: dietrich@pik-potsdam.de

Backup

Backup Slides



wrapper functions

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

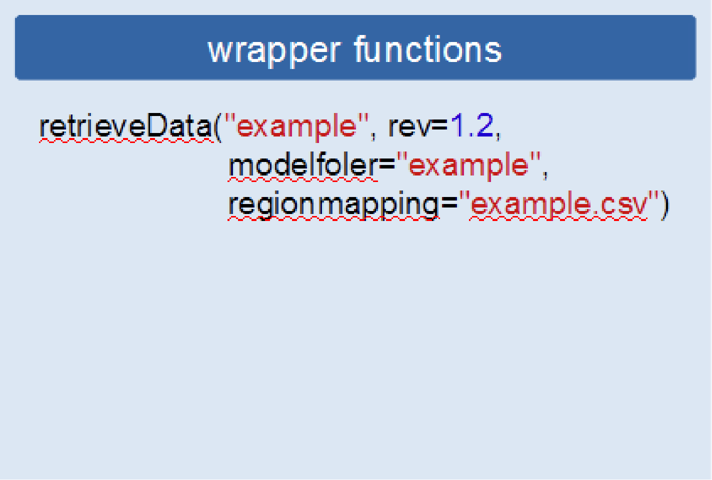
```
readSource("yours")
```

user functions

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

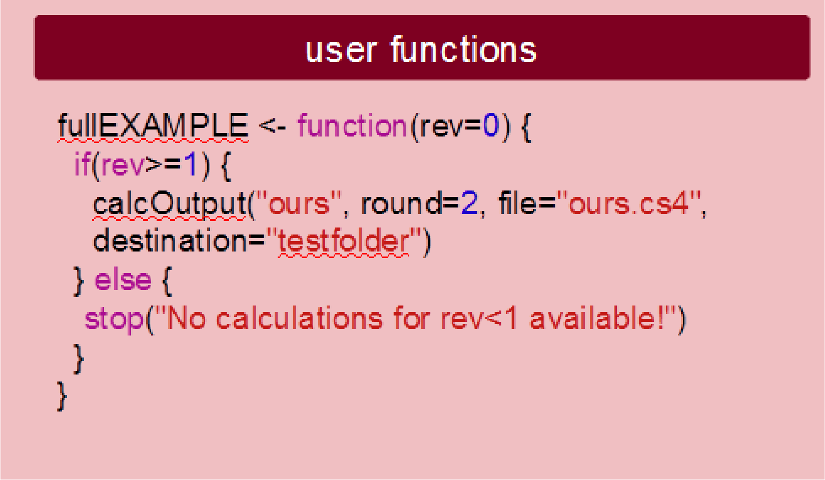
```
readYours() {
  x <- read.csv("example.csv")
  return(as.magpie(x))
}

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.