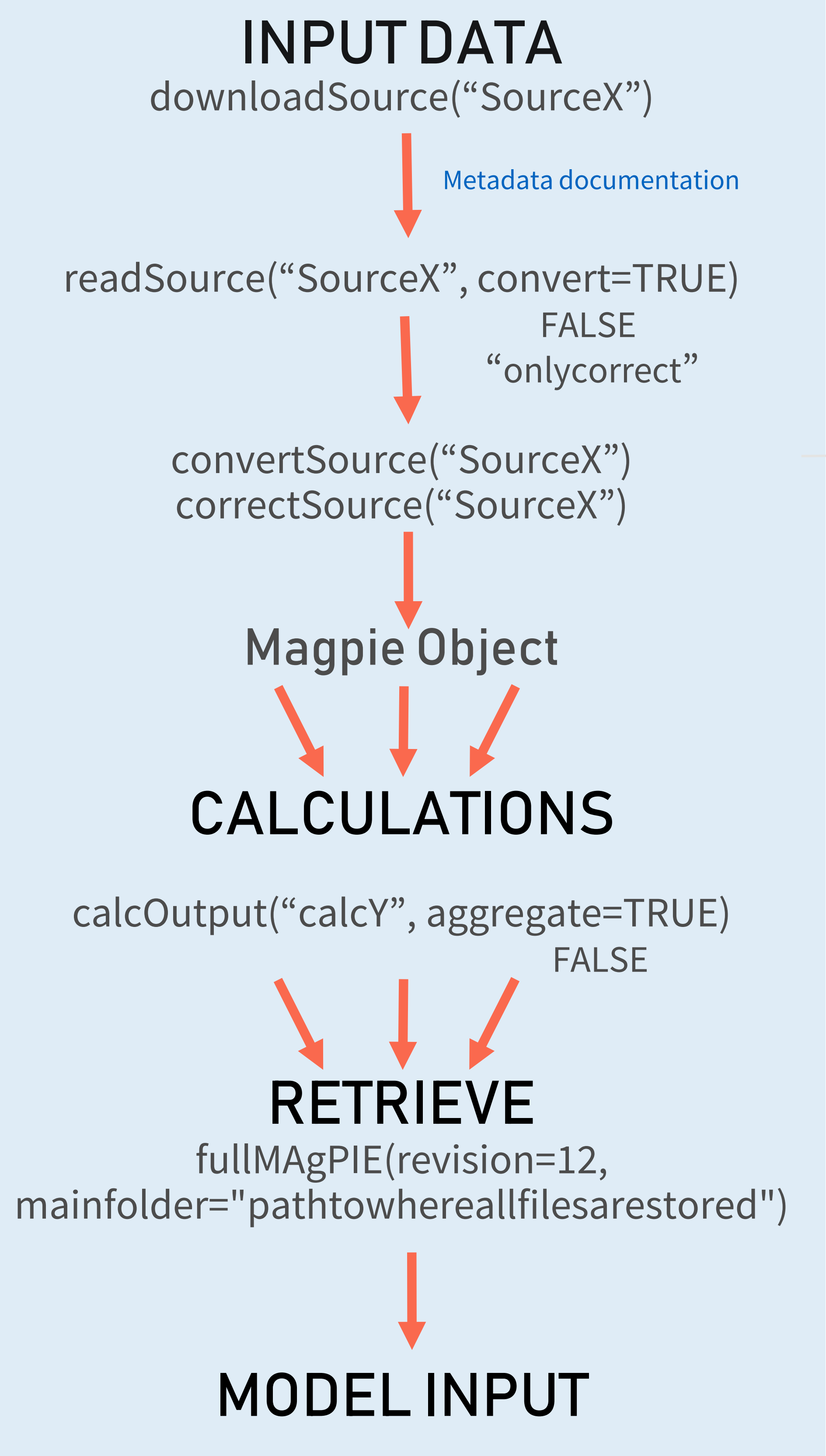
MADRaT Cheat Sheet

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

Magclass: Magpie Objects

Array with 3 Dimensions

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

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

Further documentation in ?magclass::function()

as.magpie()	Converts (tidy) 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

Useful magclass Functions

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

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