Package ‘CoSMic’

January 22, 2022

Title COVID-19 spatial microsimulation for Germany

Version 0.11.1.0000

Description A calibration-microsimulation approach to reduce uncertainty for policy decisions on non-pharmacological interventions in the COVID-19 pandemic.
The package implements an age-structured spatial microsimulation model that extends the Susceptible-Exposed-Infectious-Recovered (SEIR) framework. Using an optimization approach based on subnational trends in the number of intensive care patients, it is able to calibrate the model to the ongoing spread of the epidemic and tries to estimate how the NPIs have affected it. Based on these estimates the model can provide national and sub-national forecasts for trends in the number of ICU patients and other indicators under different scenarios regarding NPIs.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Imports magrittr, dplyr, rlist, data.table, ggplot2, lhs,
GA, doRNG, tictoc, grid, gridExtra, pracma, RColorBrewer

Roxygen list(markdown = TRUE)

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

`CoSMic-package` *CoSMic: COVID-19 spatial microsimulation for Germany*

Description

A calibration-microsimulation approach to reduce uncertainty for policy decisions on non-pharmacological interventions in the COVID-19 pandemic. The package implements an age-structured spatial microsimulation model that extends the Susceptible-Exposed-Infectious-Recovered (SEIR) framework. Using an optimization approach based on subnational trends in the number of intensive care patients, it is able to calibrate the model to the ongoing spread of the epidemic and tries to estimate how the NPIs have affected it. Based on these estimates the model can provide national and sub-national forecasts for trends in the number of ICU patients and other indicators under different scenarios regarding NPIs.

Author(s)

Maintainer: Ralf Schneider <`ralf.schneider@hlrs.de`>

Authors:

- Christian Dudel <`dudel@demogr.mpg.de`>
- Matthias Rosenbaum-Feldbrügge <`Matthias.Rosenbaum-Feldbruegge@bib.bund.de`>
- Sebastian Klüsener <`sebastian.kluesener@bib.bund.de`>

`attenuate` *Helps with delaying and smoothing changes in R0*

Description

The function smoothes numeric vectors either by logistic or linear interpolation.

Usage

```
attenuate(x, steps = 5, type = "logistic")
```

`checkpoint.check.reload` *Reload a checkpoint.*

Description

The function loads data necessary to do a checkpoint restart and checks them for usability and differences.

Usage

```
checkpoint.check.reload(ep, sp)
```

Arguments

`ep`	An execution parameter list as decribed in `set.exec.params()`.
`sp`	A list with static model parameters as described in `set.static.params()`.

`connect_total` *Comuter matrix of complete population*

Description

....

Usage

```
data(connect_total)
```

Format

An object of class `"data.frame"` with 402 columns.

1. `"dist_id"` - An integer representing the county's unique identifier.
2. `"Name"` - The name of the county.
3. `"Area"` - The counties area in km².
4. `"Inhabitants"` - The population of the county.

Source

Direct link

References

German Federal Employment Agency, Pendlerverflechtungen der sozialversicherungspflichtig Beschäftigten nach Ländern - Deutschland (Jahreszahlen), (2019), https://statistik.arbeitsagentur.de.

`connect_work` *Comuter matrix of working population*

Description

....

Usage

```
data(connect_work)
```

Format

An object of class `"data.frame"` with 402 columns.

1. `"dist_id"` - An integer representing the county's unique identifier.
2. `"Name"` - The name of the county.
3. `"Area"` - The counties area in km².
4. `"Inhabitants"` - The population of the county.

Source

Direct link

References

German Federal Employment Agency, Pendlerverflechtungen der sozialversicherungspflichtig Beschäftigten nach Ländern - Deutschland (Jahreszahlen), (2019), https://statistik.arbeitsagentur.de.

`convert.Rp.to.Fp` *Convert R-model parameters to Fortran-model input files*

Description

The function prints the R-model parameter lists and input data to textfiles which can be used as input for the Fortran model version.

Usage

```
convert.Rp.to.Fp(
  filename.sp,
  sp,
  filename.ep,
  ep,
  iol,
  R0_effects,
  outpath = "./"
)
```

Arguments

`sp`	A list with static model parameters as described in `set.static.params()`.
`filename`	Path to the output file.

`CoSMic` *Function executing the simulation model.*

Description

Function executing the simulation model.

Usage

```
CoSMic(ep, sp, iol, pspace, sim.struc, op, opt)
```

Arguments

`ep`	Execution parameter list. Use `set.exec.params()` in order to create a valid layout.
`sp`	List with static model parameters. Use `set.static.params()` to create a valid layout.
`iol`	Input data list. Use `load.input()` to load needed fies and `init.connectivity()` in order to create a valid date layout.
`pspace`	List holding the parameter space with potentially variable model parameters. Use the setter function `set.pspace()` to add parameters.
`sim.struc`	List with population data. Use `init.spatial.population()` in order to create a valid layout.
`op`	List with steering parameters for the optimization process. Use `set.optimization.params()` in order to create a valid layout and `init.reference.data()` in order to init the optimization targets based on observed data.
`opt`	Numeric vector with model parameters subject to optimization.

Value

Depends upon the selected execution procedure given by ep$exec.procedure.

1. In case `ep$exec.procedure="Optimization"` a scalar target value is returned.
2. In case `ep$exec.procedure="Basic-Param"` a list with transient result data is returned.

ToDo

- Capture Error Messages in foreach and model loop
- Fix county plots
- Implement statisitcs output against opt.targets
- Implement normed standard deviation as target value in Global daths & icu_cases & local deaths
- Implement Error message in case R0county contains county id which is not selected for simulation.

`CoSMic.Opt` *Application of the GA algorithm to CoSMic.*

Description

The function applies the GA algorithm to the CoSMic simulation model function. it uses the wrapper function `ff()` as the objective function and `GA.Monitor()` to return intermediate results during the course of the optimization.

Usage

```
CoSMic.Opt(ep, sp, iol, pspace, sim.struc, op, cl)
```

Arguments

`ep`	Execution parameter list. Use `set.exec.params()` in order to create a valid layout.
`sp`	List with static model parameters. Use `set.static.params()` to create a valid layout.
`iol`	Input data list. Use `load.input()` to load needed fies and `init.connectivity()` in order to create a valid date layout.
`pspace`	List holding the parameter space with potentially variable model parameters. Use the setter function `set.pspace()` to add parameters.
`sim.struc`	List with population data. Use `init.spatial.population()` in order to create a valid layout.
`op`	List with steering parameters for the optimization process. Use `set.optimization.params()` in order to create a valid layout and `init.reference.data()` in order to init the optimization targets based on observed data.
`cl`	A parallel cluster prepared by `init.parallel.execution()`.

`counties` *Structure of German counties*

Description

The German county structure representing the NUTS-3 level for Germany and by that the spatial simulation structure in the CoSMic default setup.

Usage

```
data(counties)
```

Format

An object of class `"data.frame"` with four columns.

1. `"dist_id"` - An integer representing the county's unique identifier.
2. `"Name"` - The name of the county.
3. `"Area"` - The counties area in km².
4. `"Inhabitants"` - The population of the county.

Source

Statistisches Bundesamt

References

Federal Statistical Office of Germany. Kreisfreie Städte und Landkreise nach Fläche, Bevölkerung und Bevölkerungsdichte, (2018), https://www.destatis.de/DE/Themen/Laender-Regionen/Regionales.

export.to.slaves *Export Variables to slaves*

Description

For convenience this function wraps the `exportDoMPI` and `clusterExport` functions from the `doMPI` and `doParallel` packages.

Usage

```
export.to.slaves(ep, cl, varlist)
```

Arguments

`ep`	Execution parameter list. Use `set.exec.params()` in order to create a valid layout.
`cl`	A parallel cluster prepared by `init.parallel.execution()`.
`varlist`	Vector of character strings representing variable names to be exported to `cl`'s workers.

ff *CoSMic model function wrapper*

Description

This function wraps the CoSMic model function so that it can be used in the GA algorithm as the objective function.

Usage

```
ff(x, ep, sp, iol, pspace, sim.struc, op)
```

Arguments

`x`	Numeric vector with model parameters subject to optimization.
`ep`	Execution parameter list. Use `set.exec.params()` in order to create a valid layout.
`sp`	List with static model parameters. Use `set.static.params()` to create a valid layout.
`iol`	Input data list. Use `load.input()` to load needed fies and `init.connectivity()` in order to create a valid date layout.
`pspace`	List holding the parameter space with potentially variable model parameters. Use the setter function `set.pspace()` to add parameters.
`sim.struc`	List with population data. Use `init.spatial.population()` in order to create a valid layout.
`op`	List with steering parameters for the optimization process. Use `set.optimization.params()` in order to create a valid layout and `init.reference.data()` in order to init the optimization targets based on observed data.

Value

A scalar value calculated according to the settings given in op.

`finalize.parallel.execution` *Finalize parallel execution execution.*

Description

For convenience this function wraps the `closeCluster` and `stopCluster` functions from the `doMPI` and `doParallel` packages.

Usage

```
finalize.parallel.execution(ep, cl)
```

Arguments

`ep`	Execution parameter list. Use `set.exec.params()` in order to create a valid layout.
`cl`	A parallel cluster prepared by `init.parallel.execution()`.

`fres.to.dataframe` *Load Fortran results as data.frame*

Description

The function converts the result files of the Fortran model version in single model execution mode to data.frames.

Usage

```
fres.to.dataframe(data.dir, basename)
```

`ftrain.to.dataframe` *Load Fortran training results as data.frame*

Description

The function converts the result files of the Fortran model version in training execution mode to data.frames.

Usage

```
ftrain.to.dataframe(data.dir, basename, split.col = "SH1")
```

`GA.Monitor` *GA algorithm monitoring function*

Description

The function provides intermediate output after each iteration of the GA algorithm.

Usage

```
GA.Monitor(
  obj,
  digits = getOption("digits"),
  sp.int = static.params,
  op.int = opt.params
)
```

Arguments

`obj`	An object provided by the GA function.
`digits`	The number of digits provided by `getOption("digits")`.
`sp.int`	List with static model parameters as created by `set.static.params()`.
`op.int`	List with steering parameters for the optimization process as created by `set.static.params()`.

`init.connectivity` *Initialize regional connectivity*

Description

The function initializes the regional connectivity matrix according to the requested regions to simulate.

Usage

```
init.connectivity(iol, sp, ss)
```

Arguments

`iol`	Input data list. Use `load.input()` to load needed fies and `init.connectivity()` in order to create a valid date layout.
`sp`	List with static model parameters. Use `set.static.params()` to create a valid layout.
`ss`	List with population data. Use `init.spatial.population()` in order to create a valid layout.

Value

An input data list with modified `connect_work` and `connect_total` components. See `load.input()` about details on how the input data list has to be strucutred in order to be correctly modified by this function.

`init.lhc` *Prepare parameter space*

Description

The function initializes the data.frame carrying the different sets of model parameters resulting from the parameter variations set in the pspace list.

Usage

```
init.lhc(pspace, sp, rep.iter = TRUE)
```

Arguments

`pspace`	The parameter list pspace set by repeated calls to `set.pspace()`
`A`	list with static model parameters as described in `set.static.params()`.

Value

A data.frame with dimension [<# different evaluations> x <potentially_variable_model_params>] If all model parameters in pspace are fixed, i.e. not variable dim(lhc) will be [sp$iter x <potentially_variable_model_params>]

`init.parallel.execution` *Initalization of the parallel execution.*

Description

The function prepares and initializes the parallel execution of the `CoSMic()` model function on computer clusters in dependence from the requested execution procedure and selected parallel execution method.

Usage

```
init.parallel.execution(ep, sp = NULL, op = NULL)
```

`init.reference.data` *Initialization of reference data.*

Description

The function adds a component to the optimization parameter list passed in as parameter `op`. The added component `opt.target` contains observed data depending which data are provided on input to the function `load.input()`. The function additionally checks whether execution of the optimization procedure is possible based on the selected optimization targets and the provided data.

Usage

```
init.reference.data(iol, op, sp, sim.struc)
```

Arguments

`iol`	Input data list. Use `load.input()` to load needed fies and `init.connectivity()` in order to create a valid date layout.
`op`	List with steering parameters for the optimization process. Use `set.optimization.params()` in order to create a valid layout.
`sp`	List with static model parameters. Use `set.static.params()` to create a valid layout.
`sim.struc`	List with population data. Use `init.spatial.population()` in order to create a valid layout.

Value

The list with steering parameters for the optimization process passed in as parameter `op` with an additional component `opt.target` carying observed data, prepared to be used as target data in the optimization procedure of the `CoSMic()` function.

ToDo

Implement ot[[dea.nuts2]]

`init.spatial.population` *Initialization of the population and its spatial structure.*

Description

The function initializes the population and its spatial structure according to the layout requested by `sp$sim.regions`, i.e. the regions selected either at county or state level to be simulated.

Usage

```
init.spatial.population(iol, sp)
```

Arguments

`iol`	Input data list. Use `load.input()` to load needed fies and `init.connectivity()` in order to create a valid date layout.
`sp`	List with static model parameters. Use `set.static.params()` to create a valid layout.

Value

A list with population data. Structured as follows:

```
sim.struc : List of 3
          $ pop      : data.frame
                     $ dist_id: int
                     $ date   : chr
                     $ sex    : chr
                     $ age_gr : int
                     $ total  : int
          $ counties: int
          $ states   : data.frame
                     $ Code       : int
                     $ inhabitants: int
                     $ Shortcut   : chr
                     $ Name       : chr
```

`load.input` *Loading input data*

Description

Loading input data

Usage

```
load.input(
  data.dir = "./",
  trans.pr = NULL,
  pop.data = NULL,
  inf.cases = NULL,
  dead.cases = NULL,
  connect.total = NULL,
  connect.work = NULL,
  sts = NULL,
  cnts = NULL,
  R0.matrix.inp = NULL,
  dead.cases.by.state = NULL,
  dead.cases.by.country = NULL,
  icu.cases.by.county = NULL,
  icu.cases.by.state = NULL,
  icu.cases.by.country = NULL,
  lhc.data = NULL
)
```

`map.R0effects` *Map R0effects from NUTS-1 to NUTS-2*

Description

The function maps R0effects on NUTS-1 i.e. German state level to R0effects on NUTS-2 level.

Usage

```
map.R0effects(R0effect.nuts2, R0effect.states, rows = NULL)
```

Arguments

`R0effect.nuts2`	R0efects to map to
`R0effect.states`	R0efects to map from
`rows`	How many rows to map

`plot.R0effect` *Plot R0effects over R0changes*

Description

The function plots timelines of the R0effects per state or NUTS2 region.

Usage

```
## S3 method for class 'R0effect'
plot(R0effect, sp, outfile = NULL, silent = FALSE)
```

`plots.by.country` *Plot timelines accross the complete country*

Description

The function plots timelines accross the complete country. Either fully aggregated with global.plot = TRUE or aggregated once across the first column of the latin hypercube, across each direct parameter with more than one value and once across the parameter set of the first directv parameter.

Usage

```
plots.by.country(
  outfile,
  sp,
  seed_icu,
  seed_dea,
  iol,
  pspace,
  rr,
  ind.states = NULL,
  global.plot,
  x.min = NULL,
  x.max = NULL,
  relative = FALSE,
  silent = FALSE,
  split.in = NULL,
  y.max = NULL,
  prog = NULL
)
```

`plots.by.state` *Plot timelines accross each state*

Description

The function plots timelines accross each state, aggregated once across the first column in the latin hypercube, ance across each direct parameter with more than one value and once across the parameter set of the first directv parameter.

Usage

```
plots.by.state(
  outfile,
  sp,
  seed_icu,
  seed_dea,
  iol,
  pspace,
  rr,
  region,
  fix.lim,
  x.min = NULL,
  x.max = NULL,
  filtered = FALSE,
  fk.cases = rep(1/7, 7),
  Sec.Axis = "RMS",
  fk.sec = rep(1/15, 15),
  sec.text = FALSE,
  ind.states = NULL,
  silent = FALSE,
  relative = FALSE,
  split.in = NULL,
  y.max = NULL,
  prog = NULL
)
```

`pop` *German population structure*

Description

The German population structure on county level (NUTS-3) stratified by age groups and sex as off 31st of December 2018.

Usage

```
data(pop)
```

Format

An object of class `"data.frame"` with five columns.

1. `"dist_id"` - An integer representing the county'S unique identifier.
2. `"date"` - Date of data publication.
3. `"sex"` - Sex of the respective age group.
4. `"age_gr"` - The age group.
5. `"total"` - Inhabitants of the county in the respective age group and with the respective sex.

Source

Statistisches Bundesamt

References

Federal Statistical Office of Germany. Kreisfreie Städte und Landkreise nach Fläche, Bevölkerung und Bevölkerungsdichte, (2018), https://www.destatis.de/DE/Themen/Laender-Regionen/Regionales.

`R0effect` *R0effect*

Description

mu values for the German NUTS-2 regions representing the R0 reduction factor per week and region as described by [Klüsener-2020]. The dataset contains mu values for the simulation of all German NUTS-2 regions for 20 weeks beginning 9th of March 2020.

Usage

```
data(R0effect)
```

Format

An object of class "data.frame" with 38 columns, one per NUTS-2 region, and 20 rows, one per week. If the simulation timeframe is to be extended, one row per week has to be added.

References

[Klüsener-2020] Klüsener S. et.al, Forecasting intensive care unit demand during the COVID-19 pandemic: A spatial age-structured microsimulation model, (2020), medRxiv, doi:10.1101/2020.12.23.20248761, https:// www.medrxiv.org/content/10.1101/2020.12.23.20248761v1 .

`save.exec.params` *Function to save the current list of execution parameters.*

Description

Function to save the current list of execution parameters.

Usage

```
save.exec.params(ep)
```

Arguments

`ep`	An execution parameter list as decribed in `set.exec.params()`.

`save.input` *Function to save the current list of loaded input data.*

Description

Function to save the current list of loaded input data.

Usage

```
save.input(ep, iol)
```

Arguments

`ep`	An execution parameter list as decribed in `set.exec.params()`.
`iol`	A list with loaded input data as described in `load.input()`.

save.optimization.params

Function to save the current list of loaded input data.

Description

Function to save the current list of loaded input data.

Usage

```
save.optimization.params(ep, op)
```

Arguments

ep	An execution parameter list as decribed in `set.exec.params()`.
op	A list with parameters steering the optimization procedure as described in `set.optimization.param`

save.pspace

Function to save the current psapce list.

Description

Function to save the current psapce list.

Usage

```
save.pspace(ep, pspace)
```

Arguments

ep	An execution parameter list as decribed in `set.exec.params()`.
pspace	The parameter list pspace.

save.spatial.population

Function to save the current list of execution parameters.

Description

Function to save the current list of execution parameters.

Usage

```
save.spatial.population(ep, sim.struc)
```

Arguments

ep	An execution parameter list as decribed in `set.exec.params()`.
sim.struc	List with population data. Use `init.spatial.population()` in order to create a valid layout.

`save.static.params` *Function to save the current list of satic model parameters.*

Description

Function to save the current list of satic model parameters.

Usage

```
save.static.params(ep, sp)
```

Arguments

`ep`	An execution parameter list as decribed in `set.exec.params()`.
`sp`	A list with static model parameters as described in `set.static.params()`.

`seed` *Infected cases*

Description

Infected cases for seeding infections during model startup.

Usage

```
data(infections)
```

Format

An object of class `"data.frame"` with ... columns.

1. `"dist_id"` - An integer representing the county's unique identifier.
2. `"Name"` - The name of the county.
3. `"Area"` - The counties area in km².
4. `"Inhabitants"` - The population of the county.

Source

COVID-19 Datenhub

References

Robert Koch-Institute, COVID-19 Dashboard, (2020),
https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/nCoV_node.html.

`seed_dea` *Dead cases*

Description

Dead cases for seeding during model startup.

Usage

```
data(seed_dea)
```

Format

An object of class `"data.frame"` with ... columns.

1. `"dist_id"` - An integer representing the county's unique identifier.
2. `"Name"` - The name of the county.
3. `"Area"` - The counties area in km².
4. `"Inhabitants"` - The population of the county.

Source

COVID-19 Datenhub

References

Robert Koch-Institute, COVID-19 Dashboard, (2020),
https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/nCoV_node.html.

`set.exec.params` *Setup of execution parameters*

Description

Setup of execution parameters

Usage

```
set.exec.params(
  exec.procedure = "Basic-Param",
  parallel.method = "OMP",
  max.cores = 4,
  omp.cluster.dbg = FALSE,
  data.dir = "data",
  output.dir = NULL,
  model.version = "12.0",
  export_name = NULL,
  cp.write = FALSE,
  cp.time = 0,
  cp.reload = FALSE,
  cp.dir = NULL
)
```

Arguments

`exec.procedure`	Set the execution procedure. Valid values are "Basic-Param" or "Optimization" *Defaults to:* `"Basic-Param"`.
`parallel.method`	Set the parallelization method. Valid values are "OMP", "MPI" or "PSOCK" *Defaults to:* `"OMP"`.
`max.cores`	Set the maximum number of cores used in case parallel.method = "OMP". *Defaults to:* `4`
`omp.cluster.dbg`	Whether std.out from workers should be captured to a file called cl.out. *Defaults to:* `FALSE`
`data.dir`	Path to the directory from which input files are read. *Defaults to:* `"data"`
`model.version`	The model version string. *Currently defaults to:* `12.0`
`export_name`	File name addition for output files. *Defaults to:* `<model.version>-<YYYY-MM-DD_hh:mm:ss>`

Value

A list with parameters needed to set up the execution of the CoSMic function. The default structure is:

```
$exec.procedure
[1] "Basic-Param"
$parallel.method
[1] "OMP"
$max.cores
[1] 4
$omp.cluster.dbg
[1] FALSE
$data.dir
[1] "Data"
$model.version
[1] "12.0"
$export_name
[1] "v12.0-2020-11-07_21:53:00"
```

`set.optimization.params` *Setup of optimization parameters*

Description

Setup of optimization parameters

Usage

```
set.optimization.params(
  opt.target.icu,
  opt.target.deaths,
  opt.target.region,
  opt.names,
  opt.lb,
  opt.ub,
  opt.pop.size,
  opt.max.iter,
  use.sug.sol,
  ep,
  sp,
  pspc,
  opt.filter = NULL
)
```

set.pspace *Setup of parameters in parameter space*

Description

The function adds an element to the parameter space list pspace

Usage

```
set.pspace(param, values, type = "direct", s.dev = NULL)
```

Arguments

`param`	The name of the parameter to be set.
`values`	The values of the parameter to be set.
`type`	The parameter type. Allowed values are `direct` or `dist`. *Defaults to:* `direct`
`s.dev`	Deviations of the values in case of parameter type `dist`. *Defaults to:* `NULL`

Value

The function operates on the global scope and modifies the parameter list pspace.\

set.static.params *Setup of static parameters*

Description

Setup of static parameters

Usage

```
set.static.params(
  pspace,
  seed.in.inner.loop = FALSE,
  seed.base = NULL,
  country = "Germany",
  restrict = TRUE,
  sim.regions = c("Schleswig-Holstein", "Hamburg", "Niedersachsen", "Bremen"),
  sam_prop.ps = c(1, 1, 1, 1),
  sim_pop = "proportional",
  ini_infected = 10,
  seed_infections = "data",
  seed_date = "2020-03-09",
  seed_before = 7,
  time_n = NULL,
  inf_dur = 3,
  cont_dur = 2,
  ill_dur = 8,
  icu_per_day = c(0, 0, 0, 0, 0, 0, 0, 8),
  less_contagious = 0.7,
  R0_force = 0,
  immune_stop = TRUE,
  import_R0_matrix = FALSE,
 R0change = lapply(seq(1, by = 7, length.out = 20), function(x) {    c(x, x + 6) }),
  R0county = as.list(rep("ALL", 20)),
  R0delay = TRUE,
  R0delay_days = 5,
  R0delay_type = "linear",
  endogenous_lockdown = FALSE,
  lockdown_effect = 0.39,
  lockdown_connect = 0.5,
  lockdown_threshold = 100,
  lockdown_days = 10,
  control_age_sex = "age",
  iter = 4,
  lhc.samples = NULL,
  lhc.reload = FALSE,
  gplots = FALSE,
  cplots = FALSE,
  cplots.states = FALSE,
  cplots.nuts2 = FALSE,
  results = "Reduced",
  sp.states = NULL
)
```

setup.projection *Extraploate 0effects beyond determined values*

Description

The function extrapolates R0effects based on different methods.

Usage

```
setup.projection(
  R0effect,
  sp,
  method = "constant-daily",
  base = NULL,
  length = 8,
  length.days = 14
)
```

Arguments

`R0effect`	A data.frame with R0effects per week and region.
`sp`	An object with static CoSMic model parameters.
`method`	Method by which to extrapolate. Supported values are: `"constant-weekly"`: Extrapolates constantly the R0effect of week `base` to the next `length` weeks. `"constant-daily"`: Determines the averaged daily R0effect from the last `length.days` and extraploates it constantly to the next `length` weeks. *Defaults to:* `"constant-weekly"`.
`base`	The week based on which to extrapolate. If not given the last week i.e. dim(R0effect)[1] is used. *Defaults to:* `NULL`.
`length`	Number of week to extrapolate after base. *Defaults to:* `8`.
`length.days`	Number of days to take into account when extraploation based on daily quantities is done. *Defaults to:* `14`.

states *Structure of German federal states*

Description

The German federal state structure representing the NUTS-1 level for Germany.

Usage

```
data(states)
```

Format

An object of class `"data.frame"` with four columns.

1. `"Code"` - An integer representing the states unique identifier.
2. `"inhabitants"` - The population of the state.
3. `"Shortcut"` - The two letter identifier of the state.
4. `"Name"` - The name of the state.

Source

Statistisches Bundesamt

References

Federal Statistical Office of Germany. Kreisfreie Städte und Landkreise nach Fläche, Bevölkerung und Bevölkerungsdichte, (2018), https://www.destatis.de/DE/Themen/Laender-Regionen/Regionales.

`trans_pr` *Transition probabilities*

Description

The dataset describes the transition probabilities to intensive care units along with the chance to survive a Corona virus infction when beeing ill or being ill and in intensive care stratified by age groups.

Usage

```
data(trans_pr)
```

Format

An object of class `"data.frame"` with five columns.

1. `"age_gr"` - Age groups with 5 year stepping from 0 - 90 years.
2. `"sex"` - Label for stratification according t sex using labels "total", "m" and "f"
3. `"surv_ill"` - Chance of surviving an infection.
4. `"icu_risk"` - Risk for intensive care requirement when infected.
5. `"surv_icu"` - Chance of surviving intensive care.

Index