Package 'covid19.analytics'

April 13, 2020

Type Package

Title Load and Analyze Live Data from the CoViD-19 Pandemic

Version 1.0.1

Date 2020-04-12

Author Marcelo Ponce [aut, cre]

Maintainer Marcelo Ponce <mponce@scinet.utoronto.ca>

Description Load and analyze updated time series worldwide data of reported cases for the Novel CoronaVirus Disease (CoViD-19) from the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) data repository <https://github.com/CSSEGISandData/COVID-19>. The datasets are available in two main modalities, as a time series sequences and aggregated for the last day with greater spatial resolution. Several analysis, visualization and modelling functions are available in the package that will allow the user to compute and visualize total number of cases, total number of changes and growth rate globally or for an specific geographical location, while at the same time generating models using these trends; generate interactive visualizations and generate Susceptible-Infected-Recovered (SIR) model for the disease spread.

Imports ape, plotly, htmlwidgets, deSolve, gplots, pheatmap

Suggests knitr, devtools, roxygen2

License GPL (>= 2)

URL https://mponce0.github.io/covid19.analytics/

BugReports https://github.com/mponce0/covid19.analytics/issues

RoxygenNote 7.0.2

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-13 06:50:08 UTC

R topics documented:

`covid19.data` *function to read "live" data as reported by JHU's CCSE repository*

Description

function to read "live" data as reported by JHU's CCSE repository

Usage

```
covid19.data(case = "aggregated", local.data = FALSE, debrief = FALSE)
```

Arguments

`case`	a string indicating the category of the data, possible values are: "aggregated" : latest number of cases *aggregated* by country, "ts-confirmed" : time data of confirmed cases, "ts-deaths" : time series data of fatal cases, "ts-recovered" : time series data of recovered cases, "ts-ALL" : all time series data combined, "ts-dep-confirmed" : time series data of confirmed cases as originally reported (depricated), "ts-dep-deaths" : time series data of deaths as originally reported (depricated), "ts-dep-recovered" : time series data of recovered cases as originally reported (depricated), "ALL": all of the above
`local.data`	boolean flag to indicate whether the data will be read from the local repo, in case of connectivity issues or data integrity
`debrief`	boolean specifying whether information about the read data is going to be displayed in screen

Value

a dataframe (or a list in the case of "ALL") with the daily worlwide indicated type of data per country/region/city

Examples

```
# reads all possible datastest, returnin a list
covid19.all.datasets <- covid19.data("ALL")

# reads the latest aggregated data
covid19.ALL.agg.cases <- covid19.data("aggregated")
# reads time series data for casualities
covid19.TS.deaths <- covid19.data("ts-deaths")
```

---

covid19.genomic.data *function to obtain sequencing data grom NCBI Reference: https://www.ncbi.nlm.nih.gov/nuccore/NC_045512.2*

Description

function to obtain sequencing data grom NCBI Reference: https://www.ncbi.nlm.nih.gov/nuccore/NC_045512.2

Usage

```
covid19.genomic.data(graphics.ON = TRUE)
```

Arguments

graphics.ON	flag to activate/deactivate graphical output

Examples

```
# obtain covid19's genomic data
covid19.gen.seq <- covid19.genomic.data()
# display the actual RNA seq
covid19.gen.seq$NC_045512.2
```

---

generate.SIR.model *function to generate a simple SIR (Susceptible-Infected-Recovered) model based on the actual data of the coivd19 cases*

Description

function to generate a simple SIR (Susceptible-Infected-Recovered) model based on the actual data of the coivd19 cases

Usage

```
generate.SIR.model(
  data = NULL,
  geo.loc = "Hubei",
  t0 = NULL,
  t1 = NULL,
  deltaT = NULL,
  tfinal = 90,
  fatality.rate = 0.02,
  tot.population = 1.4e+09,
  staticPlt = TRUE,
  interactiveFig = FALSE
)
```

Arguments

data	time series dataset to consider
geo.loc	country/region to analyze
t0	initial period of time for data consideration
t1	final period of time for data consideration
deltaT	interval period of time from t0, ie. number of days to consider since t0
tfinal	total number of days
fatality.rate	rate of causality, deafault value of 2 percent
tot.population	total population of the country/region
staticPlt	optional flag to activate/deactive plotting of the data and the SIR model generated
interactiveFig	optional flag to activate/deactive the generation of an interactive plot of the data and the SIR model generated

Examples

```
data <- covid19.data("ts-confirmed")
generate.SIR.model(data,"Hubei", t0=1,t1=15)
generate.SIR.model(data,"Germany",tot.population=83149300)
generate.SIR.model(data,"Uruguay", tot.population=3500000)
generate.SIR.model(data,"Canada", tot.population=37590000)
```

`growth.rate` *function to compute daily changes and "Growth Rates" per location; "Growth Rates" defined as the ratio between changes in consecutive days*

Description

function to compute daily changes and "Growth Rates" per location; "Growth Rates" defined as the ratio between changes in consecutive days

Usage

```
growth.rate(data0, geo.loc = NULL, stride = 1, info = "")
```

Arguments

data0	data.frame with *time series* data from covid19
geo.loc	list of locations
stride	how frequently to compute the growth rate in units of days
info	additional information to include in plots' title

Value

a list containing two dataframes: one reporting changes on daily baisis and a second one reporting growth rates, for the indicated regions

Examples

```
###\donttest{
# read data for confirmed cases
data <- covid19.data("ts-confirmed")
# compute changes and growth rates per location for all the countries
growth.rate(data)
# compute changes and growth rates per location for 'Italy'
growth.rate(data,geo.loc="Italy")
# compute changes and growth rates per location for 'Italy' and 'Germany'
growth.rate(data,geo.loc=c("Italy","Germany"))
###}
```

---

live.map *function to map cases in an interactive map*

Description

function to map cases in an interactive map

Usage

```
live.map(
  data = covid19.data(),
  projctn = "orthographic",
  title = "",
  szRef = 0.2,
  fileName = NULL
)
```

Arguments

data	data to be used
projctn	initial type of map-projection to use, possible values are: "equirectangular" | "mercator" | "orthographic" | "natural earth" | "kavrayskiy7" | "miller" | "robinson" | "eckert4" | "azimuthal equal area" | "azimuthal equidistant" | "conic equal area" | "conic conformal" | "conic equidistant" | "gnomonic" | "stereographic" | "mollweide" | "hammer" | "transverse mercator" | "albers usa" | "winkel tripel" | "aitoff" | "sinusoidal"
title	a string with a title to add to the plot
szRef	numerical value to use as reference, to scale up the size of the bubbles in the map, from 0 to 1 (smmaller value –> larger bubbles)
fileName	file where to save the HTML version of the interactive figure

Examples

```
# retrieve aggregated data
data <- covid19.data("aggregated")
# interactive map of aggregated cases -- with more spatial resolution
live.map(data)

# interactive map of the time series data of the confirmed cases
# with less spatial resolution, ie. aggregated by country
live.map(covid19.data("ts-confirmed"))
```

---

movingFn *generic fn that computes the "fn" on a moving window*

Description

generic fn that computes the "fn" on a moving window

Usage

```
movingFn(x, fn = mean, period = length(x), direction = "forward")
```

Arguments

x	a numeric vector
fn	a function to be applied/computed, default is set to mean()
period	size of the "moving window", default set to the lenght of the vector
direction	type of moving avergage to consider: "forward", "centered", "backward"; ie. whether the window computation is ("centered" / "forward" / "backward") wrt the data series

Value

a vector with the 'moving operation' applied to the x vector

`plt.SIR.model` *function to plot the results from the SIR model fn*

Description

function to plot the results from the SIR model fn

Usage

```
plt.SIR.model(SIR.model, geo.loc = "", interactiveFig = FALSE, fileName = NULL)
```

Arguments

`SIR.model`	model resulting from the generate.SIR.model() fn
`geo.loc`	optional string to specify geographical location
`interactiveFig`	optional flag to activate interactive plot
`fileName`	file where to save the HTML version of the interactive figure

---

`report.summary` *function to summarize the current situation, will download the latest data and summarize the top provinces/cities per case*

Description

function to summarize the current situation, will download the latest data and summarize the top provinces/cities per case

Usage

```
report.summary(
  cases.to.process = "ALL",
  Nentries = 10,
  graphical.output = TRUE,
  saveReport = FALSE
)
```

Arguments

cases.to.process	which data to process: "TS" –time series–, "AGG" –aggregated– or "ALL" –time series and aggregated–
Nentries	number of top cases to display
graphical.output	flag to deactivate graphical output
saveReport	flag to indicate whether the report should be saved in a file

Examples

```
# displaying top 10s
report.summary()

# get the top 20
report.summary(Nentries=20)
```

---

totals.plt *function to plot total number of cases per day for different groups*

Description

function to plot total number of cases per day for different groups

Usage

```
totals.plt(
  data0 = NULL,
  geo.loc0 = NULL,
  one.plt.per.page = FALSE,
  log.plt = TRUE,
  with.totals = FALSE,
  interactive.fig = TRUE,
  fileName = NULL
)
```

Arguments

data0	time series dataset to process, default all the possible cases: 'confirmed' and 'deaths' for all countries/regions
geo.loc0	geographical location, country/region or province/state to restrict the analysis to
one.plt.per.page	boolean flag to have one plot per figure
log.plt	include a log scale plot in the static plot

with.totals	a boolean flag to indicate whether the totals should be displayed with the records for the specific location
interactive.fig	swith to turn off/on an interactive plot
fileName	file where to save the HTML version of the interactive figure

Examples

```
# retrieve time series data
TS.data <- covid19.data("ts-ALL")

# static and interactive plot
totals.plt(TS.data)
```

---

tots.per.location *function to compute totals per location*

Description

function to compute totals per location

Usage

```
tots.per.location(
  data,
  geo.loc = NULL,
  confBnd = FALSE,
  nbr.plts = 1,
  info = ""
)
```

Arguments

data	data.frame with *time series* data from covid19
geo.loc	list of locations
confBnd	flag to activate/deactivate drawing of confidence bands base on a moving average window
nbr.plts	parameter to control the number of plots to display per figure
info	additional info to display in plots' titles

Value

a list or dataframe with totals per specified locations and type of case

Examples

```
# read data for confirmed cases
data <- covid19.data("ts-confirmed")
# compute totals per location for all the countries

tots.per.location(data)

# compute totals per location for 'Italy'
tots.per.location(data,geo.loc="Italy")
# compute totals per location for 'Italy' and 'Germany'
tots.per.location(data,geo.loc=c("Italy","Germany"))
```

Index