Package ‘portalr’

February 20, 2019

Title Create Useful Summaries of the Portal Data

Version 0.2.2

Description Download and generate summaries for the rodent, plant, ant, and weather data from the Portal Project. Portal is a long-term (and ongoing) experimental monitoring site in the Chihuahua desert. The raw data files can be found at <https://github.com/weecology/portaldata>.

License MIT + file LICENSE

URL `https://weecology.github.io/portalr/`, `https://github.com/weecology/portalr`

BugReports `https://github.com/weecology/portalr/issues`

LazyData true

Depends R (>= 3.2.3)

Imports dplyr, ggplot2, tidyr, zoo, lubridate, magrittr, httr, rlang, forecast, lunar, jsonlite, tibble

Suggests httptest, digest, tidyverse, cowplot, knitr, rmarkdown, pkgdown, covr

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Glenda M. Yenni [aut, cre] (<https://orcid.org/0000-0001-6969-1848>),
Hao Ye [aut] (<https://orcid.org/0000-0002-8630-1458>),
Erica M. Christensen [aut] (<https://orcid.org/0000-0002-5635-2502>),
Juniper L. Simonis [aut] (<https://orcid.org/0000-0001-9798-0460>),
Ellen K. Bledsoe [aut] (<https://orcid.org/0000-0002-3629-7235>),
Renata M. Diaz [aut] (<https://orcid.org/0000-0003-0803-4734>),
Shawn D. Taylor [aut] (<https://orcid.org/0000-0002-6178-6903>),
Ethan P, White [aut] (<https://orcid.org/0000-0001-6728-7745>),
S.K. Morgan Ernest [aut] (<https://orcid.org/0000-0002-6026-8530>)

Maintainer Glenda M. Yenni <`glenda@weecology.org`>

Repository CRAN

Date/Publication 2019-02-20 19:40:04 UTC

R topics documented:

add_seasons 2
bait_presence_absence 3
check_for_newer_data 4
clean_plant_data 4
clean_rodent_data 5
colony_presence_absence 6
download_observations 7
fcast_ndvi 7
fill_missing_ndvi 8
find_incomplete_censuses 9
get_data_versions 9
get_future_moons 10
load_datafile 10
load_rodent_data 11
make_timeseries 13
ndvi 13
portalr 14
shrub_cover 14
summarize_individual_rodents 15
summarize_plant_data 16
summarize_rodent_data 17
weather 19

Index **20**

---

add_seasons *Add Seasons*

Description

Higher-order data summaries, by 6-month seasons, 3-month seasons, or year. Also applies specified functions to the specified summary level.

`yearly` generates a table of yearly means

Usage

```
add_seasons(data, level = "site", season_level = 2,
  date_column = "yearmon", summarize = NA, path = "~",
  download_if_missing = TRUE, clean = TRUE)

yearly(...)
```

Arguments

`data`	data frame containing columns: date, period, newmoonnumber, or year and month
`level`	"plot, "treatment" or "site"
`season_level`	either year, 2: winter = Oct-March summer = April-Sept 4: winter = Dec-Feb spring = March-May summer = Jun-Aug fall = Sep-Nov
`date_column`	either "date" (must be in format "y-m-d"), "period", "newmoonnumber", or "yearmon" (data must contain "year" and "month")
`summarize`	A function or list of functions specified by their name (e.g. "mean"). Default is NA (returned with seasons added but not summarized).
`path`	path to location of downloaded Portal data; or "repo" to retrieve data from github repo
`download_if_missing`	if the specified file path doesn't have the PortalData folder, then download it
`clean`	logical, load only QA/QC rodent data (TRUE) or all data (FALSE)
`...`	arguments passed to `add_seasons`

Value

a data.frame with additional "season" and "year" column, and other columns summarized as specified

Examples

```
yearly(abundance(path = "repo", time = "newmoon"),
       date_column = "newmoonnumber", path = "repo")
```

`bait_presence_absence` *Ant Bait Presence Absence*

Description

Get ant species presence/absence by year/plot/stake from bait census data

Bait census data is more consistent over time than the colony census data. This function assumes that all species present in at least one census were censused in all years.

Usage

```
bait_presence_absence(path = "~", level = "Site",
  download_if_missing = TRUE)
```

Arguments

`path`	path to location of downloaded Portal data; or "repo" to retrieve data from github repo
`level`	level at which to summarize data: ’Site’, ’Plot’, or ’Stake’
`download_if_missing`	if the specified file path doesn’t have the PortalData folder, then download it

Value

data frame with year, species, (plot if applicable), and presence [1, 0]

`check_for_newer_data` *Check for latest version of data files*

Description

Check the latest version against the data that exists on the GitHub repo

Usage

```
check_for_newer_data(base_folder = "~")
```

Arguments

`base_folder`	Folder in which data will be checked

Value

bool TRUE if there is a newer version of the data online

`clean_plant_data` *Do basic cleaning of Portal plant data*

Description

This function does basic quality control of the Portal plant data. It is mainly called from `summarize_plant_data`, with several arguments passed along.

The specific steps it does are, in order: (1) correct species names according to recent vouchers, if requested (2) restrict species to annuals or non-woody (3) remove records for unidentified species (5) exclude the plots that aren’t long-term treatments

Usage

```
clean_plant_data(data_tables, type = "All", unknowns = FALSE,
  correct_sp = TRUE)
```

Arguments

`data_tables`	the list of data_tables, returned from calling `load_plant_data`
`type`	specify subset of species; If type=Annuals, removes all non-annual species. If type=Non-woody, removes shrub and subshrub species If type=Perennials, returns all perennial species (includes shrubs and subshrubs) If type=Shrubs, returns only shrubs and subshrubs If type=Winter-annual, returns all annuals found in winter IF type=Summer-annual, returns all annuals found in summer
`unknowns`	either removes all individuals not identified to species (unknowns = FALSE) or sums them in an additional column (unknowns = TRUE)
`correct_sp`	T/F whether or not to use likely corrected plant IDs, passed to `rename_species_plants`

`clean_rodent_data` *Do basic cleaning of Portal rodent data*

Description

This function does basic quality control of the Portal rodent data. It is mainly called from `summarize_rodent_data`, with several arguments passed along.

The specific steps it does are, in order: (1) add in missing weight data (2) remove records with "bad" period codes or plot numbers (3) remove records for unidentified species (4) exclude non-granivores (5) exclude incomplete trapping sessions (6) exclude the plots that aren't long-term treatments

Usage

```
clean_rodent_data(data_tables, fillweight = FALSE, type = "Rodents",
  unknowns = FALSE)
```

Arguments

`data_tables`	the list of data_tables, returned from calling `load_rodent_data`
`fillweight`	specify whether to fill in unknown weights with other records from that individual or species, where possible
`type`	specify subset of species; either all "Rodents" or only "Granivores"
`unknowns`	either removes all individuals not identified to species (unknowns = FALSE) or sums them in an additional column (unknowns = TRUE)

colony_presence_absence *Ant Colony Presence Absence*

Description

Get ant species presence/absence by year/plot/stake from colony census data

Anomalies in ant colony census protocol over the years means that it can be difficult to discern true absences of all species in all years. This function uses information from Portal_ant_species.csv and Portal_ant_dataflags.csv to predict true presence/absence of species per plot per year. If a more conservative estimate is desired, setting the argument 'rare_sp = T' will only include species we are confident were censused regularly. Setting 'rare_sp = F' may include some false absences, since it is unknown if some rare species were censused in all years. Unknowns may also be excluded from output if desired.

Usage

```
colony_presence_absence(path = "~", level = "Site", rare_sp = F,
  unknowns = F, download_if_missing = TRUE)
```

Arguments

`path`	path to location of downloaded Portal data; or "repo" to retrieve data from github repo
`level`	level at which to summarize data: 'Site', 'Plot', or 'Stake'
`rare_sp`	include rare species (T) or not (F). Rare species may or may not have been censused in all years. Setting rare_sp=F gives a more conservative estimate of presence/absence
`unknowns`	include unknown species (T) or not (F). Unknowns include those only identified to genus.
`download_if_missing`	if the specified file path doesn't have the PortalData folder, then download it

Value

data frame with year, species, (plot if applicable), and presence [1, 0, NA]

`download_observations` *Download the PortalData repo*

Description

This downloads the latest portal data regardless if they are actually updated or not. TODO: incorporate data retriever into this when it's pointed at the github repo

Usage

```
download_observations(base_folder = "~", version = "latest",
  from_zenodo = FALSE)
```

Arguments

`base_folder`	Folder into which data will be downloaded
`version`	Version of the data to download (default = "latest")
`from_zenodo`	logical; if 'TRUE', get info from Zenodo, otherwise GitHub

Value

None

Examples

```
download_observations()
download_observations("~/old-data", version = "1.50.0")
```

`fcast_ndvi` *Forecast ndvi using a seasonal auto ARIMA*

Description

Forecast ndvi using a seasonal auto ARIMA

Usage

```
fcast_ndvi(hist_ndvi, level, lead, moons = NULL)
```

Arguments

`hist_ndvi`	historic ndvi data
`level`	specify "monthly" or "newmoon"
`lead`	number of steps forward to forecast
`moons`	moon data (required if level = "newmoon")

Details

ndvi values are forecast using auto.arima with seasonality (using a Fourier transform)

Value

a data.frame with time and ndvi values

`fill_missing_ndvi` *Fill in historic ndvi data to the complete timeseries being fit*

Description

Fill in historic ndvi data to the complete timeseries being fit

Usage

```
fill_missing_ndvi(ndvi, level, last_time, moons = NULL)
```

Arguments

`ndvi`	ndvi data
`level`	specify "monthly" or "newmoon"
`last_time`	the last time step to have been completed
`moons`	moon data (required if level = "newmoons" and forecasts are needed)

Details

missing values during the time series are replaced using na.interp, missing values at the end of the time series are forecast using auto.arima with seasonality (using Fourier transform)

Value

a data.frame with time and ndvi values

find_incomplete_censuses — *Period code for incomplete censuses*

Description

Determines incomplete censuses by finding dates when some plots were trapped, but others were not.

Usage

```
find_incomplete_censuses(trapping_table, min_plots, min_traps)
```

Arguments

`trapping_table`	Data table that contains sampled column (1 for sampled, 0 for unsampled)
`min_plots`	minimum number of plots in a census for a census to count as sampled
`min_traps`	minimum number of traps on a plot for a census to count as sampled

Value

Data.table of period codes when not all plots were trapped.

get_data_versions — *get version and download info for PortalData*

Description

Check either Zenodo or GitHub for the version and download link for PortalData.

Usage

```
get_data_versions(from_zenodo = FALSE, halt_on_error = FALSE)
```

Arguments

`from_zenodo`	logical; if ‘TRUE‘, get info from Zenodo, otherwise GitHub
`halt_on_error`	logical; if ‘FALSE‘, return NULL on errors, otherwise whatever got returned (could be an error or warning)

Value

A data.frame with two columns, ‘version‘ (string with the version #) and ‘zipball_url‘ (download URLs for the corresponding zipped release).

`get_future_moons` *Get future moon dates*

Description

Get next 12 new moon dates and assign newmoon numbers for forecasting

Usage

```
get_future_moons(moons, num_future_moons = 12)
```

Arguments

`moons`	current newmoonnumber table
`num_future_moons`	number of future moons to get

Value

expected moons table for 12 future new moons

`load_datafile` *read in a raw datafile from the downloaded data or the GitHub repo*

Description

does checking for whether a particular datafile exists and then reads it in, using na_strings to determine what gets converted to NA. It can also download the dataset if it's missing locally.

Usage

```
load_datafile(datafile, na.strings = "", path = "~",
  download_if_missing = TRUE, version_message = FALSE)
```

Arguments

`datafile`	the path to the datafile within the folder for Portal data
`na.strings`	a character vector of strings which are to be interpreted as `NA` values. Blank fields are also considered to be missing values in logical, integer, numeric and complex fields. Note that the test happens *after* white space is stripped from the input, so `na.strings` values may need their own white space stripped in advance.
`path`	either the file path that contains the PortalData folder or "repo", which then pulls data from the PortalData GitHub repository

`download_if_missing`	if the specified file path doesn't have the PortalData folder, then download it
`version_message`	whether to display a message about the data version being loaded

Examples

```
rodent_species <- load_datafile("Rodents/Portal_rodent_species.csv")
```

`load_rodent_data` *Read in the Portal data files*

Description

Loads Portal data files from either a user-defined path or the online Github repository. If the user-defined path is un- available, the default option is to download to that location.

`load_rodent_data` loads the rodent data files

`load_plant_data` loads the plant data files

`load_ant_data` loads the ant data files

`load_trapping_data` loads just the rodent trapping files

Usage

```
load_rodent_data(path = "~", download_if_missing = TRUE,
  clean = TRUE)

load_plant_data(path = "~", download_if_missing = TRUE)

load_ant_data(path = "~", download_if_missing = TRUE)

load_trapping_data(path = "~", download_if_missing = TRUE,
  clean = TRUE)
```

Arguments

`path`	either the file path that contains the PortalData folder or "repo", which then pulls data from the PortalData GitHub repository
`download_if_missing`	if the specified file path doesn't have the PortalData folder, then download it
`clean`	logical, load only QA/QC rodent data (TRUE) or all data (FALSE)

Value

`load_rodent_data` returns a list of 5 dataframes:

`rodent_data`	raw data on rodent captures
`species_table`	species code, names, types
`trapping_table`	when each plot was trapped
`newmoons_table`	pairs census periods with newmoons
`plots_table`	rodent treatment assignments for each plot

`load_plant_data` returns a list of 7 dataframes:

`quadrat_data`	raw plant quadrat data
`species_table`	species code, names, types
`census_table`	indicates whether each quadrat was counted in each census; area of each quadrat
`date_table`	start and end date of each plant census
`plots_table`	rodent treatment assignments for each plot
`transect_data`	raw plant transect data with length and height (2015-present)
`oldtransect_data`	raw plant transect data as point counts (1989-2009)

`load_ant_data` returns a list of 4 dataframes:

`bait_data`	raw ant bait data
`colony_data`	raw ant colony data
`species_table`	species code, names, types
`plots_table`	treatment assignments for each plot

`load_trapping_data` returns a list of 2 dataframes:

`trapping_table`	when each plot was trapped
`newmoons_table`	pairs census periods with newmoons

Examples

```
portal_data <- load_rodent_data("repo")

portal_plant_data <- load_plant_data("repo")

portal_ant_data <- load_ant_data("repo")

trapping_data <- load_trapping_data("repo")
```

make_timeseries — *Makes a standardized monthly time series for Portal rodents*

Description

The Portal rodent data is collected roughly monthly. However, some time series methods require regular monthly data with no gaps. This function deals with the issue that some monthly censuses occur slightly before or after their intended sample month. The function takes a univariate time series and pairs samples up with their intended month, averages out double censuses that occur in the same month and interpolates gaps in the data. To work properly the data should be a dataframe containing a single time series where each date contains a single value.

Usage

```
make_timeseries(data, date_format = "%Y-%m-%d")
```

Arguments

`data`	Dataframe with columns date (date of the period (e.g. 2016-01-01)), period (period code for the census) and value (numeric value to be analyzed)
`date_format`	format for the dattes in date column (e.g. "%Y-%m-%d")

Value

dataframe containing 2 columns: newdate and value

ndvi — *NDVI by calendar month or lunar month*

Description

Summarize NDVI data to monthly or lunar monthly level

Usage

```
ndvi(level = "monthly", fill = FALSE, path = "~")
```

Arguments

`level`	specify "monthly" or "newmoon"
`fill`	specify if missing data should be filled, passed to `fill_missing_ndvi`
`path`	specify where to locate Portal data

`portalr`	*Creates summaries of the Portal data*

Description

This package is designed to be an interface to the Portal data, which resides online at `https://github.com/weecology/portalData`. Its contains a set of functions to download, clean, and summarize the data.

`shrub_cover`	*Generate percent cover from Portal plant transect data*

Description

This function calculates percent cover from transect data. It handles the pre-2015 data differently from the current transects, becase they are collected differently. But it returns a single time-series with all years of transect data available. It also returns mean height beginning in 2015.

Usage

```
shrub_cover(path = "~", type = "Shrubs", plots = "all",
  unknowns = FALSE, correct_sp = TRUE)
```

Arguments

`path`	path to location of downloaded Portal data; or "repo" to retrieve data from github repo
`type`	specify subset of species; If type=Annuals, removes all non-annual species. If type=Summer Annuals, returns all annual species that can be found in the summer If type=Winter Annuals, returns all annual species that can be found in the winter If type=Non-woody, removes shrub and subshrub species If type=Perennials, returns all perennial species (includes shrubs and subshrubs) If type=Shrubs, returns only shrubs and subshrubs
`plots`	specify subset of plots; can be a vector of plots, or specific sets: "all" plots or "Longterm" plots (plots that have had the same treatment for the entire time series)
`unknowns`	either removes all individuals not identified to species (unknowns = FALSE) or sums them in an additional column (unknowns = TRUE)
`correct_sp`	correct species names suspected to be incorrect in early data (T/F)

Value

a data.frame of percent cover and mean height

summarize_individual_rodents
Return cleaned Portal rodent individual data

Description

This function cleans and subsets the data based on a number of arguments. It returns stake number and individual level data.

Usage

```
summarize_individual_rodents(path = "~", clean = TRUE,
  type = "Rodents", length = "all", unknowns = FALSE,
  time = "period", fillweight = FALSE, min_plots = 1,
  min_traps = 1, download_if_missing = TRUE)

summarise_individual_rodents(path = "~", clean = TRUE,
  type = "Rodents", length = "all", unknowns = FALSE,
  time = "period", fillweight = FALSE, min_plots = 1,
  min_traps = 1, download_if_missing = TRUE)
```

Arguments

`path`	path to location of downloaded Portal data; or "repo" to retrieve data from github repo
`clean`	logical, load only QA/QC rodent data (TRUE) or all data (FALSE)
`type`	specify subset of species; either all "Rodents" or only "Granivores"
`length`	specify subset of plots; use "All" plots or only "Longterm" plots (plots that have had same treatment for entire time series)
`unknowns`	either removes all individuals not identified to species (unknowns = FALSE) or sums them in an additional column (unknowns = TRUE)
`time`	specify the format of the time index in the output, either "period" (sequential Portal surveys), "newmoon" (lunar cycle numbering), "date" (calendar date)
`fillweight`	specify whether to fill in unknown weights with other records from that individual or species, where possible
`min_plots`	minimum number of plots within a period for an observation to be included
`min_traps`	minimum number of plots within a period for an observation to be included
`download_if_missing`	if the specified file path doesn't have the PortalData folder, then download it

Value

a data.frame

summarize_plant_data *Generate summaries of Portal plant data*

Description

This function is a generic interface into creating summaries of the Portal plant species data. It contains a number of arguments to specify both the kind of data to summarize, at what level of aggregation, various choices for dealing with data quality, and output format.

`plant_abundance` generates a table of plant abundance

Usage

```
summarize_plant_data(path = "~", level = "Site", type = "All",
  length = "all", plots = length, unknowns = FALSE,
  correct_sp = TRUE, shape = "flat", output = "abundance",
  na_drop = switch(tolower(level), plot = FALSE, treatment = TRUE, site =
  TRUE), zero_drop = switch(tolower(level), plot = FALSE, treatment =
  TRUE, site = TRUE), min_quads = 1, effort = TRUE,
  download_if_missing = TRUE)

plant_abundance(..., shape = "flat")

summarise_plant_data(path = "~", level = "Site", type = "All",
  length = "all", plots = length, unknowns = FALSE,
  correct_sp = TRUE, shape = "flat", output = "abundance",
  na_drop = switch(tolower(level), plot = FALSE, treatment = TRUE, site =
  TRUE), zero_drop = switch(tolower(level), plot = FALSE, treatment =
  TRUE, site = TRUE), min_quads = 1, effort = TRUE,
  download_if_missing = TRUE)
```

Arguments

`path`	path to location of downloaded Portal data; or "repo" to retrieve data from github repo
`level`	summarize by "Plot", "Treatment", or "Site"
`type`	specify subset of species; If type=Annuals, removes all non-annual species. If type=Summer Annuals, returns all annual species that can be found in the summer If type=Winter Annuals, returns all annual species that can be found in the winter If type=Non-woody, removes shrub and subshrub species If type=Perennials, returns all perennial species (includes shrubs and subshrubs) If type=Shrubs, returns only shrubs and subshrubs
`length`	specify subset of plots; use "All" plots or only "Longterm" plots (to be deprecated)
`plots`	specify subset of plots; can be a vector of plots, or specific sets: "all" plots or "Longterm" plots (plots that have had the same treatment for the entire time series)

`unknowns`	either removes all individuals not identified to species (unknowns = FALSE) or sums them in an additional column (unknowns = TRUE)
`correct_sp`	correct species names suspected to be incorrect in early data (T/F)
`shape`	return data as a "flat" list or "crosstab"
`output`	specify whether to return "abundance", or "cover" [cover data starts in summer 2015]
`na_drop`	logical, drop NA values (representing insufficient sampling)
`zero_drop`	logical, drop 0s (representing sufficient sampling, but no detections)
`min_quads`	numeric [1:16], minimum number of quadrats (out of 16) for a plot to be included
`effort`	logical as to whether or not the effort columns should be included in the output
`download_if_missing`	if the specified file path doesn't have the PortalData folder, then download it
`...`	arguments passed to `summarize_plant_data`

Value

a data.frame in either "long" or "wide" format, depending on the value of ‘shape‘

Examples

```
plant_abundance("repo")
```

`summarize_rodent_data` *Generate summaries of Portal rodent data*

Description

This function is a generic interface into creating summaries of the Portal rodent species data. It contains a number of arguments to specify the kind of data to summarize (at what level of aggregation) and various choices for dealing with data quality, and output format.

abundance generates a table of rodent abundance

* `biomass()` generates a table of rodent biomass

* `energy()` generates a table of rodent energy (computed as 5.69 * (biomass ^ 0.75) after White et al 2004)

Usage

```
summarize_rodent_data(path = "~", clean = TRUE, level = "Site",
  type = "Rodents", length = "all", plots = length,
  unknowns = FALSE, shape = "crosstab", time = "period",
  output = "abundance", fillweight = (output != "abundance"),
  na_drop = switch(tolower(level), plot = FALSE, treatment = TRUE, site =
  TRUE), zero_drop = switch(tolower(level), plot = FALSE, treatment =
  TRUE, site = TRUE), min_traps = 1, min_plots = 24, effort = FALSE,
  download_if_missing = TRUE)

abundance(...)

biomass(...)

energy(...)

summarise_rodent_data(path = "~", clean = TRUE, level = "Site",
  type = "Rodents", length = "all", plots = length,
  unknowns = FALSE, shape = "crosstab", time = "period",
  output = "abundance", fillweight = (output != "abundance"),
  na_drop = switch(tolower(level), plot = FALSE, treatment = TRUE, site =
  TRUE), zero_drop = switch(tolower(level), plot = FALSE, treatment =
  TRUE, site = TRUE), min_traps = 1, min_plots = 24, effort = FALSE,
  download_if_missing = TRUE)
```

Arguments

`path`	path to location of downloaded Portal data; or "repo" to retrieve data from github repo
`clean`	logical, load only QA/QC rodent data (TRUE) or all data (FALSE)
`level`	summarize by "Plot", "Treatment", or "Site"
`type`	specify subset of species; either all "Rodents" or only "Granivores"
`length`	specify subset of plots; use "All" plots or only "Longterm" plots (to be deprecated)
`plots`	specify subset of plots; can be a vector of plots, or specific sets: "all" plots or "Longterm" plots (plots that have had the same treatment for the entire time series)
`unknowns`	either removes all individuals not identified to species (unknowns = FALSE) or sums them in an additional column (unknowns = TRUE)
`shape`	return data as a "crosstab" or "flat" list
`time`	specify the format of the time index in the output, either "period" (sequential Portal surveys), "newmoon" (lunar cycle numbering), "date" (calendar date)
`output`	specify whether to return "abundance", or "biomass", or "energy"
`fillweight`	specify whether to fill in unknown weights with other records from that individual or species, where possible

`na_drop`	logical, drop NA values (representing insufficient sampling)
`zero_drop`	logical, drop 0s (representing sufficient sampling, but no detections)
`min_traps`	minimum number of traps for a plot to be included
`min_plots`	minimum number of plots within a period for an observation to be included
`effort`	logical as to whether or not the effort columns should be included in the output
`download_if_missing`	if the specified file path doesn't have the PortalData folder, then download it
`...`	arguments passed to `summarize_rodent_data`

Value

a data.frame in either "long" or "wide" format, depending on the value of ‘shape‘

Examples

```
abundance("repo")


biomass("repo")


energy("repo")
```

`weather` *Weather by day, calendar month, or lunar month*

Description

Summarize hourly weather data to either daily, monthly, or lunar monthly level.

Usage

```
weather(level = "daily", fill = FALSE, path = "~")
```

Arguments

`level`	specify 'monthly', 'daily', or 'newmoon'
`fill`	specify if missing data should be filled, passed to `fill_missing_weather`
`path`	specify where to locate Portal data

Index

∗Topic **package**
    portalr, 14

abundance *(summarize_rodent_data)*, 17
add_seasons, 2, *3*

bait_presence_absence, 3
biomass *(summarize_rodent_data)*, 17

check_for_newer_data, 4
clean_plant_data, 4
clean_rodent_data, 5
colony_presence_absence, 6

download_observations, 7

energy *(summarize_rodent_data)*, 17

fcast_ndvi, 7
fill_missing_ndvi, 8
find_incomplete_censuses, 9

get_data_versions, 9
get_future_moons, 10

load_ant_data, *11, 12*
load_ant_data *(load_rodent_data)*, 11
load_datafile, 10
load_plant_data, *5, 11, 12*
load_plant_data *(load_rodent_data)*, 11
load_rodent_data, *5*, *11*, 11
load_trapping_data, *11, 12*
load_trapping_data *(load_rodent_data)*, 11

make_timeseries, 13

NA, *10*
ndvi, 13

plant_abundance *(summarize_plant_data)*, 16

portalr, 14
portalr-package *(portalr)*, 14

shrub_cover, 14
summarise_individual_rodents *(summarize_individual_rodents)*, 15
summarise_plant_data *(summarize_plant_data)*, 16
summarise_rodent_data *(summarize_rodent_data)*, 17
summarize_individual_rodents, 15
summarize_plant_data, *4*, 16, *17*
summarize_rodent_data, *5*, 17, *19*

weather, 19

yearly *(add_seasons)*, 2