

# Package ‘incidence2’

November 7, 2021

**Type** Package

**Title** Compute, Handle and Plot Incidence of Dated Events

**Version** 1.2.3

**Description** Provides functions and classes to compute, handle and visualise incidence from dated events for a defined time interval. Dates can be provided in various standard formats. The class 'incidence2' is used to store computed incidence and can be easily manipulated, subsetted, and plotted. This package is part of the RECON (<<https://www.repidemicsconsortium.org/>>) toolkit for outbreak analysis (<<https://www.reconverse.org/>>).

**Encoding** UTF-8

**License** MIT + file LICENSE

**URL** <https://github.com/reconverse/incidence2>

**BugReports** <https://github.com/reconverse/incidence2/issues>

**RoxygenNote** 7.1.2

**LazyData** true

**Imports** tibble, ellipsis, vctrs, pillar, data.table, stats, rlang (>= 0.1.2), tidyselect, grates (>= 0.3.0), dplyr, clock

**Suggests** outbreaks, knitr, rmarkdown, testthat, scales, roxygen2, ggplot2, magrittr, covr, zoo

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Depends** R (>= 2.10)

**NeedsCompilation** no

**Author** Tim Taylor [aut, cre] (<<https://orcid.org/0000-0002-8587-7113>>),  
Thibaut Jombart [ctb],  
Zhian N. Kamvar [ctb] (<<https://orcid.org/0000-0003-1458-7108>>)

**Maintainer** Tim Taylor <[tim.taylor@hiddenelephants.co.uk](mailto:tim.taylor@hiddenelephants.co.uk)>

**Repository** CRAN

**Date/Publication** 2021-11-07 22:00:02 UTC

**R topics documented:**

accessors	2
as.data.frame.incidence_df	5
as_tibble	5
build_incidence	6
complete_counts	7
covidregionaldataUK	8
cumulate	9
incidence	10
keep	13
new_incidence	14
plot.incidence2	15
print_incidence	18
regroup	19
summary.incidence_df	19
vibrant	20
<b>Index</b>	<b>21</b>

---

accessors	<i>Access various elements of an incidence object</i>
-----------	---

---

**Description**

Access various elements of an incidence object  
 NULL if none are present.

**Usage**

```
get_counts(x, ...)

## Default S3 method:
get_counts(x, ...)

## S3 method for class 'incidence_df'
get_counts(x, ...)

get_count_names(x, ...)

## Default S3 method:
get_count_names(x, ...)

## S3 method for class 'incidence_df'
get_count_names(x, ...)

get_date_index(x, ...)
```

```
## Default S3 method:
get_date_index(x, ...)

## S3 method for class 'incidence_df'
get_date_index(x, ...)

get_dates(x, ...)

get_dates_name(x, ...)

## Default S3 method:
get_dates_name(x, ...)

## S3 method for class 'incidence_df'
get_dates_name(x, ...)

get_group_names(x, ...)

## Default S3 method:
get_group_names(x, ...)

## S3 method for class 'incidence_df'
get_group_names(x, ...)

get_timespan(x, ...)

## Default S3 method:
get_timespan(x, ...)

## S3 method for class 'incidence2'
get_timespan(x, ...)

get_n(x)

## Default S3 method:
get_n(x)

## S3 method for class 'incidence_df'
get_n(x)

get_interval(x, ...)

## Default S3 method:
get_interval(x, ...)

## S3 method for class 'incidence2'
get_interval(x, ...)
```

## Arguments

x	An <code>incidence()</code> object.
...	Not used.

## Value

- `get_counts`: The count vector from x.
- `get_count_names()`: The name of the count variable of x.
- `get_date_index()`: The `date_index` vector from x.
- `get_dates()`: Same as `get_date_index()`.
- `get_dates_name()`: The name of the `date_index` variable of x.
- `get_group_names()`: a character vector of the group variables of x or
- `get_timespan()`: an integer denoting the timespan in days represented by the incidence object.
- `get_n()` The total number of cases stored in the object
- `get_interval()`: if `integer = TRUE`, an integer vector, otherwise the character value of the interval

## Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
  withAutoprint({
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist
    i <- incidence(dat,
                  date_index = date_of_onset,
                  groups = c(gender, hospital))

    get_counts(i)
    get_count_names(i)

    get_group_names(i)

    get_date_index(i)
    get_dates_name(i)

    get_interval(i)

    get_n(i)

    get_timespan(i)
  })
}
```

---

```
as.data.frame.incidence_df
  Convert incident object to dataframe
```

---

**Description**

Convert incident object to dataframe

**Usage**

```
## S3 method for class 'incidence_df'
as.data.frame(x, ...)
```

**Arguments**

x	An <a href="#">incidence()</a> object.
...	Not used.

**Examples**

```
dat <- data.frame(dates = Sys.Date() + 1:100,
                 names = rep(c("Jo", "John"), 5))

dat <- incidence(dat, date_index = dates, groups = names)
as.data.frame(dat)
```

---

```
as_tibble          Convert incident object to a tibble
```

---

**Description**

Convert incident object to a tibble

**Usage**

```
## S3 method for class 'incidence_df'
as_tibble(x, ...)
```

**Arguments**

x	An <a href="#">incidence()</a> object.
...	Not used.

**Examples**

```
dat <- data.frame(dates = Sys.Date() + 1:100,
                 names = rep(c("Jo", "John"), 5))

dat <- incidence(dat, date_index = dates, groups = names)
as_tibble(dat)
```

---

build_incidence	<i>Coerce to incidence</i>
-----------------	----------------------------

---

**Description**

build\_incidence() coerces an object to an incidence of events.

**Usage**

```
build_incidence(
  x,
  date_index,
  groups = NULL,
  counts = NULL,
  na_as_group = TRUE,
  FUN = identity,
  args = list()
)
```

**Arguments**

x	A data frame representing a linelist (or potentially a pre-aggregated dataset).
date_index	The time index(es) of the given data. Multiple inputs only make sense when x is a linelist, and in this situation, to avoid ambiguity, the vector must be named. These names will be used for the resultant count columns.
groups	An optional vector giving the names of the groups of observations for which incidence should be grouped.
counts	The count variables of the given data. If NULL (default) the data is taken to be a linelist of individual observations.
na_as_group	A logical value indicating if missing group values (NA) should be treated as a separate category (TRUE) or removed from consideration (FALSE). Defaults to TRUE.
FUN	Function applied to the dates_index vectors before grouping. The first argument of FUN must work with a dates_index vector. Defaults to the identity function.
args	List of additional arguments passed to FUN.

## Value

An `incidence_df`` object. This is a subclass of `[tibble][tibble::tbl-df`]` represents an aggregated count of observations. It will contain the following columns:

**date\_index**: If the default interval of 1 day is used then this will be the dates of the given observations and given the name "date", otherwise, this will be values obtained from the specified date grouping with column name "date\_index" (See Interval specification below).

- **count** (or name of count variables): The aggregated observation counts.
- **groups** (if specified): column(s) containing the categories of the given groups.

---

complete_counts	<i>Complete counts for all date and group combinations</i>
-----------------	--

---

## Description

This function ensures that an incidence object has the same range of dates for each grouping. By default missing counts will be filled with 0L.

## Usage

```
complete_counts(x, fill = 0L)
```

## Arguments

x	An <code>incidence()</code> object.
fill	The value to replace missing counts by. Defaults to 0L.

## Examples

```
dat <- data.frame(  
  dates = Sys.Date() + 1:4,  
  groups = rep(c("grp1", "grp2"), 2),  
  counts = 1:4  
)  
  
i <- incidence(dat, date_index = dates, groups = groups, counts = counts)  
complete_counts(i)
```

---

covidregionaldataUK *Regional data for COVID-19 cases in the UK*

---

### Description

A dataset containing the daily time-series of cases, tests, hospitalisations, and deaths for UK.

### Usage

```
covidregionaldataUK
```

### Format

A data frame with 6370 rows and 26 variables:

**date** the date that the counts were reported (YYYY-MM-DD)

**region** the region name

**region\_code** the region code

**cases\_new** new reported cases for that day

**cases\_total** total reported cases up to and including that day

**deaths\_new** new reported deaths for that day

**deaths\_total** total reported deaths up to and including that day

**recovered\_new** new reported recoveries for that day

**recovered\_total** total reported recoveries up to and including that day

**hosp\_new** new reported hospitalisations for that day

**hosp\_total** total reported hospitalisations up to and including that day (note this is cumulative total of new reported, not total currently in hospital).

**tested\_new** tests for that day

**tested\_total** total tests completed up to and including that day

### Note

Extracted using the `covidregionaldata` package on 2021-06-03.

### Source

<https://CRAN.R-project.org/package=covidregionaldata>



---

cumulate	<i>Compute cumulative 'incidence'</i>
----------	---------------------------------------

---

## Description

cumulate is an S3 generic to compute cumulative numbers, with methods for different types of objects:

- default method is a wrapper for cumsum
- incidence objects: computes cumulative incidence over time

## Usage

```
cumulate(x, ...)  
  
## Default S3 method:  
cumulate(x, ...)  
  
## S3 method for class 'incidence_df'  
cumulate(x, fill = 0L, ...)
```

## Arguments

x	An incidence object.
...	Not currently used
fill	Value to complete missing date-grouping combinations with. If NULL, no completion is performed. Default: 0L.

## Examples

```
dat <- data.frame(  
  dates = as.integer(c(0,1,2,2,3,5,7)),  
  groups = factor(c(1, 2, 3, 3, 3, 3, 1))  
)  
  
i <- incidence(dat, date_index = dates, groups = groups)  
i  
  
cumulative_i <- cumulate(i)  
cumulative_i
```

---

incidence                      *Compute the incidence of events*

---

### Description

Compute the incidence of events

### Usage

```
incidence(
  x,
  date_index,
  groups = NULL,
  interval = 1L,
  na_as_group = TRUE,
  counts = NULL,
  firstdate = NULL
)
```

### Arguments

x	A data frame representing a linelist (or potentially a pre-aggregated dataset).
date_index	The time index(es) of the given data. This should be the name(s) corresponding to the desired date column(s) in x of class: integer, numeric, Date, POSIXct, POSIXlt, and character. (See Note about numeric and character formats). Multiple inputs only make sense when x is a linelist, and in this situation, to avoid ambiguity, the vector must be named. These names will be used for the resultant count columns.
groups	An optional vector giving the names of the groups of observations for which incidence should be grouped.
interval	An integer or character indicating the (fixed) size of the time interval used for computing the incidence; defaults to 1 day. This can also be a text string that corresponds to a valid date interval, e.g. <ul style="list-style-type: none"> <li>* (x) day(s)</li> <li>* (x) weeks(s)</li> <li>* (x) epiweeks(s)</li> <li>* (x) isoweeks(s)</li> <li>* (x) months(s)</li> <li>* (x) quarter(s)</li> <li>* (x) years(s)</li> </ul> More details can be found in the "Interval specification" and "Week intervals" sections below.
na_as_group	A logical value indicating if missing group values (NA) should be treated as a separate category (TRUE) or removed from consideration (FALSE). Defaults to TRUE.

counts	The count variables of the given data. If NULL (default) the data is taken to be a linelist of individual observations.
firstdate	When the interval is numeric or in days/months and has a numeric prefix greater than 1, then you can optionally specify the date that you wish to anchor your intervals to begin from. If NULL (default) then the intervals will start at the minimum value contained in the date_index column. Note that the class of firstdate must be Date if the date_index column is Date, POSIXct, POSIXlt, or character and integer otherwise.

### Value

An incidence2 object. This is a subclass of `incidence_df` and aggregated count of observations grouped according to the specified interval and, optionally, the given groups. By default it will contain the following columns:

- **date / date\_index**: If the default interval of 1 day is used then this will be the dates of the given observations and given the name "date", otherwise, this will be values obtained from the specified date grouping with column name "date\_index" (See Interval specification below).
- **groups** (if specified): Column(s) containing the categories of the given groups.
- **count** (or name of count variables): The aggregated observation counts.

### Note

#### Input data (date\_index):

- **Decimal (numeric) dates**: will be truncated.
- **Character dates** should be in the unambiguous yyyy-mm-dd (ISO 8601) format. Any other format will trigger an error.

**Interval specification (interval):** `incidence()` uses the `grates` package to generate date groupings. The grouping used depends on the value of `interval`. This can be specified as either an integer value or a more standard specification such as "day", "week", "month", "quarter" or "year". The format in this situation is similar to that used by `seq.Date()` where these values can optionally be preceded by a (positive or negative) integer and a space, or followed by "s". When no prefix is given:

- "week" : uses the "grates\_yearweek" class (see `grates::as_yearweek()`).
- "month" : uses the "grates\_month" class (see `grates::as_month()`).
- "quarter" : uses the "grates\_quarter" class (see `grates::as_quarter()`).
- "year" : uses the "grates\_year" class (see `grates::as_year()`).

When a prefix is provided (e.g. 2 weeks) the output is an object of class "period" (see `as_period()`). Note that for the values "month", "quarter" and "year" intervals are always chosen to start at the beginning of the calendar equivalent. If the input is an integer value the input is treated as if it was specified in days (i.e. 2 and 2 days) produce the same output.

The only interval values that do not produce these grouped classes are 1, 1L, "day" or "days" (both without prefix) are used. In this situation the returned object is of the standard "Date" class.

#### Week intervals:

It is possible to construct incidence objects standardized to any day of the week. The default state is to use ISO 8601 definition of weeks, which start on Monday. You can specify the day of the week an incidence object should be standardised to by using the pattern "n W weeks" where "W" represents the weekday in an English or current locale and "n" represents the duration, but this can be omitted. Below are examples of specifying weeks starting on different days assuming we had data that started on 2016-09-05, which is ISO week 36 of 2016:

- interval = "2 monday weeks" (Monday 2016-09-05)
- interval = "1 tue week" (Tuesday 2016-08-30)
- interval = "1 Wed week" (Wednesday 2016-08-31)
- interval = "1 Thursday week" (Thursday 2016-09-01)
- interval = "1 F week" (Friday 2016-09-02)
- interval = "1 Saturday week" (Saturday 2016-09-03)
- interval = "Sunday week" (Sunday 2016-09-04)

It's also possible to use something like "3 weeks: Saturday"; In addition, there are keywords reserved for specific days of the week:

- interval = "week", (Default, Monday)
- interval = "ISOweek" (Monday)
- interval = "EPIweek" (Sunday)
- interval = "MMWRweek" (Sunday)

## Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {
  withAutoprint({
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist

    # daily incidence
    incidence(dat, date_of_onset)

    # weekly incidence
    incidence(dat, date_of_onset, interval = "week")

    # starting on a Monday
    incidence(dat, date_of_onset, interval = "isoweek")

    # starting on a Sunday
    incidence(dat, date_of_onset, interval = "epiweek")

    # group by gender
    incidence(dat, date_of_onset, interval = 7, groups = gender)

    # group by gender and hospital
    incidence(dat, date_of_onset, interval = "2 weeks", groups = c(gender, hospital))
  })
}

# use of first_date
dat <- data.frame(dates = Sys.Date() + sample(-3:10, 10, replace = TRUE))
```

```
incidence(dat, dates, interval = "week", firstdate = Sys.Date() + 1)
```

---

keep *Keep first and last occurrences*

---

### Description

keep\_first() (keep\_last) keeps the first (last) n entries to occur by date ordering.

### Usage

```
keep_first(x, n, ...)

## Default S3 method:
keep_first(x, n, ...)

## S3 method for class 'incidence_df'
keep_first(x, n, ...)

## S3 method for class 'grates_yearweek'
keep_first(x, n, ...)

## S3 method for class 'grates_month'
keep_first(x, n, ...)

## S3 method for class 'grates_quarter'
keep_first(x, n, ...)

## S3 method for class 'grates_year'
keep_first(x, n, ...)

## S3 method for class 'grates_period'
keep_first(x, n, ...)

keep_last(x, n, ...)

## Default S3 method:
keep_last(x, n, ...)

## S3 method for class 'incidence_df'
keep_last(x, n, ...)

## S3 method for class 'grates_yearweek'
keep_last(x, n, ...)

## S3 method for class 'grates_month'
```

```
keep_last(x, n, ...)  
  
## S3 method for class 'grates_quarter'  
keep_last(x, n, ...)  
  
## S3 method for class 'grates_year'  
keep_last(x, n, ...)  
  
## S3 method for class 'grates_period'  
keep_last(x, n, ...)
```

### Arguments

x	Object to filter.
n	Number of entries to keep.
...	Not currently used.

### Value

The objected with the chosen entries.

---

new_incidence	<i>Incidence constructor and validator</i>
---------------	--

---

### Description

Creates or validates an incidence object. Mainly of use to those developing packages to work with incidence objects.

### Usage

```
new_incidence(  
  x,  
  date,  
  groups = NULL,  
  counts,  
  measurements = NULL,  
  validate = TRUE  
)  
  
validate_incidence(x)
```

**Arguments**

x	An incidence-like object
date	The time index of 'x'.
groups	An optional vector giving the names of the groups in x.
counts	The count variables of x
measurements	An optional vector giving the names of measurement variables in x.
validate	A logical value indicating whether to validate the input. If FALSE, only minimal checks are made which can give a performance advantage if so desired.

**Details**

`new_incidence()` creates a new incidence object which is a subclass of a tibble (i.e. class `incidence`, `tbl_df`, `tbl` and `data.frame`).

`validate_incidence()` checks the object for internal consistency. For an object to be considered an incidence object it must: \* inherit the `incidence` and `data.frame` class; \* have a single column representing the `date_index` with the name of this variable being stored in the `date` attribute; \* have one or more columns representing the counts with the name of these variables being stored in the `counts` attribute; \* have zero or more columns representing groups with, if and only if present, the names of these being stored in the `groups` attribute; \* have zero or more columns representing measurement with, if and only if present, the names of these being stored in the `measurements` attribute; \* not have duplicated rows with regards to the date and group variables.

**Value**

An incidence object (invisibly for `validate_incidence()`)

---

plot.incidence2      *Plotting functions*

---

**Description**

`incidence2` includes two plotting functions to simplify graph creation.

**Usage**

```
## S3 method for class 'incidence2'
plot(
  x,
  count = NULL,
  fill = NULL,
  centre_dates = TRUE,
  date_format = "%Y-%m-%d",
  stack = TRUE,
  title = NULL,
  col_pal = vibrant,
```

```

    alpha = 0.7,
    color = NA,
    xlab = "",
    ylab = NULL,
    n_breaks = 6,
    width = 1,
    show_cases = FALSE,
    border = "white",
    na_color = "grey",
    legend = c("right", "left", "bottom", "top", "none"),
    angle = 0,
    size = NULL,
    ...
)

facet_plot(x, ...)

## S3 method for class 'incidence2'
facet_plot(
  x,
  count = NULL,
  facets = NULL,
  centre_dates = TRUE,
  date_format = "%Y-%m-%d",
  stack = TRUE,
  fill = NULL,
  title = NULL,
  col_pal = vibrant,
  alpha = 0.7,
  color = NA,
  xlab = "",
  ylab = NULL,
  n_breaks = 3,
  width = 1,
  show_cases = FALSE,
  border = "white",
  na_color = "grey",
  legend = c("bottom", "top", "left", "right", "none"),
  angle = 0,
  size = NULL,
  nrow = NULL,
  ...
)

```

### Arguments

x	An <code>incidence()</code> object.
count	Which count variable to have on the y-axis. If NULL (default) the first entry



	returned from <code>get_count_names(x)</code> is used.
<code>fill</code>	Which variable to color plots by. If NULL no distinction is made for plot colors.
<code>centre_dates</code>	If the interval is one of a single week, month, quarter or year the <code>x_axis</code> labels are centred with custom category labels. Set this option to <code>FALSE</code> to use date labels at the breaks.
<code>date_format</code>	Format to use if "Date" scales are required. The value is used by <code>format.Date()</code> and can be any input acceptable by that function (defaults to <code>"%Y-%m-%d"</code> ).
<code>stack</code>	A logical indicating if bars of multiple groups should be stacked, or displayed side-by-side. Only used if <code>fill</code> is not NULL.
<code>title</code>	Optional title for the graph.
<code>col_pal</code>	<code>col_pal</code> The color palette to be used for the groups; defaults to vibrant (see <code>?palettes</code> ).
<code>alpha</code>	The alpha level for color transparency, with 1 being fully opaque and 0 fully transparent; defaults to 0.7.
<code>color</code>	The color to be used for the borders of the bars; NA for invisible borders; defaults to NA.
<code>xlab</code>	The label to be used for the x-axis; empty by default.
<code>ylab</code>	The label to be used for the y-axis; by default, a label will be generated automatically according to the time interval used in incidence computation.
<code>n_breaks</code>	Approximate number of breaks calculated using <code>scales::breaks_pretty</code> (default 6).
<code>width</code>	Value between 0 and 1 indicating the relative size of the bars to the interval. Default 1.
<code>show_cases</code>	if <code>TRUE</code> (default: <code>FALSE</code> ), then each observation will be colored by a border. The border defaults to a white border unless specified otherwise. This is normally used outbreaks with a small number of cases. Note: this can only be used if <code>stack = TRUE</code>
<code>border</code>	If <code>show_cases</code> is <code>TRUE</code> this represents the color used for the borders of the individual squares plotted (defaults to "white").
<code>na_color</code>	The colour to plot NA values in graphs (default: grey).
<code>legend</code>	Position of legend in plot.
<code>angle</code>	Rotation angle for text.
<code>size</code>	text size in pts.
<code>...</code>	other arguments to pass to <code>ggplot2::scale_x_continuous()</code> .
<code>facets</code>	Which variable to facet plots by. If NULL will use all <code>group_labels</code> of the incidence object.
<code>nrow</code>	Number of rows.

### Details

- `plot` creates a one-pane graph of an incidence object.
- `facet_plot` creates a multi-facet graph of a grouped incidence object. If the object has no groups it returns the same output as a call to `plot()`.
- If the `incidence()` object has a rolling average column then that average will be overlaid on top.

**Value**

- `facet_plot()` and `plot()` generate a `ggplot2::ggplot()` object.

**Examples**

```
if (requireNamespace("outbreaks", quietly = TRUE) && requireNamespace("ggplot2", quietly = TRUE)) {
  withAutoprint({
    data(ebola_sim_clean, package = "outbreaks")
    dat <- ebola_sim_clean$linelist

    inci <- incidence(dat,
                      date_index = date_of_onset,
                      interval = 7,
                      groups = hospital)

    inci2 <- incidence(dat,
                      date_index = date_of_onset,
                      interval = 7,
                      groups = c(hospital, gender))

    plot(inci)
    plot(inci, fill = hospital)
    plot(inci, fill = hospital, stack = FALSE)

    facet_plot(inci)
    facet_plot(inci2)
    facet_plot(inci2, facets = gender)
    facet_plot(inci2, facets = hospital, fill = gender)
  })
}
```

---

`print_incidence`      *Print an incidence object.*

---

**Description**

Print an incidence object.

**Usage**

```
## S3 method for class 'incidence_df'
print(x, ...)
```

```
## S3 method for class 'incidence_df'
format(x, ...)
```

**Arguments**

`x`                    An 'incidence' object.

`...`                Additional arguments passed through to the tibble format method.

---

regroup	<i>Regroup 'incidence' objects</i>
---------	------------------------------------

---

### Description

This function regroups an `incidence()` object across the specified groups. The resulting `incidence()` object will contains counts summed over the groups present in the input.

### Usage

```
regroup(x, groups = NULL)
```

### Arguments

x	An <code>incidence()</code> object.
groups	The groups to sum over. If NULL (default) then the function ignores all groups.

### Examples

```
if (requireNamespace("outbreaks", quietly = TRUE)) {  
  withAutoprint({  
    data(ebola_sim_clean, package = "outbreaks")  
    dat <- ebola_sim_clean$linelist  
    i <- incidence(dat,  
                  date_index = date_of_onset,  
                  groups = c(gender, hospital))  
  
    regroup(i)  
  
    regroup(i, hospital)  
  })  
}
```

---

summary.incidence_df	<i>Summary of an incidence object</i>
----------------------	---------------------------------------

---

### Description

Summary of an incidence object

### Usage

```
## S3 method for class 'incidence_df'  
summary(object, ...)
```

**Arguments**

object	An 'incidence' object.
...	Not used.

**Value**

object (invisibly).

---

vibrant	<i>Color palettes used in incidence</i>
---------	---

---

**Description**

These functions are color palettes used in incidence. The palettes come from <https://personal.sron.nl/~pault/#sec:qualitative> and exclude grey, which is reserved for missing data.

**Usage**

vibrant(n)

muted(n)

**Arguments**

n	a number of colors
---	--------------------

**Examples**

```
vibrant(5)
muted(10)
```

# Index

- \* **datasets**
  - covidregionaldataUK, 8
- accessors, 2
- as.data.frame.incidence\_df, 5
- as\_period(), 11
- as\_tibble, 5
  
- build\_incidence, 6
  
- complete\_counts, 7
- covidregionaldataUK, 8
- cumulate, 9
  
- facet\_plot(plot.incidence2), 15
- format.incidence\_df(print\_incidence), 18
  
  
- get\_count\_names(accessors), 2
- get\_counts(accessors), 2
- get\_date\_index(accessors), 2
- get\_dates(accessors), 2
- get\_dates\_name(accessors), 2
- get\_group\_names(accessors), 2
- get\_interval(accessors), 2
- get\_n(accessors), 2
- get\_timespan(accessors), 2
- ggplot2::ggplot(), 18
- ggplot2::scale\_x\_continuous(), 17
- grates::as\_month(), 11
- grates::as\_quarter(), 11
- grates::as\_year(), 11
- grates::as\_yearweek(), 11
  
- incidence, 10
- incidence(), 4, 5, 7, 16, 17, 19
- incidence\_df, 11
  
- keep, 13
- keep\_first(keep), 13
- keep\_last(keep), 13
  
- muted(vibrant), 20
  
- new\_incidence, 14
  
- palettes(vibrant), 20
- plot(), 17
- plot.incidence2, 15
- print.incidence\_df(print\_incidence), 18
- print\_incidence, 18
  
- regroup, 19
  
- seq.Date(), 11
- summary.incidence\_df, 19
  
- validate\_incidence(new\_incidence), 14
- vibrant, 20