

# Package ‘mantis’

February 11, 2026

**Type** Package

**Title** Multiple Time Series Scanner

**Version** 1.0.2

## Description

Generate interactive html reports that enable quick visual review of multiple related time series stored in a data frame. For static datasets, this can help to identify any temporal artefacts that may affect the validity of subsequent analyses. For live data feeds, regularly scheduled reports can help to pro-actively identify data feed problems or unexpected trends that may require action. The reports are self-contained and shareable without a web server.

**URL** <https://github.com/ropensci/mantis>,  
<https://ropensci.github.io/mantis/>

**BugReports** <https://github.com/ropensci/mantis/issues>

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 4.1.0)

**Imports** rmarkdown, knitr, reactable, dplyr (>= 1.1.1), tidy,  
dygraphs, xts, ggplot2, scales, purrr, htmltools, lubridate

**Suggests** covr, testthat (>= 3.0.0), codemetar, vdiff, shiny,  
diffviewer, withr

**Config/testthat/edition** 3

**RoxygenNote** 7.3.3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** T. Phuong Quan [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-8566-1817>>),  
University of Oxford [cph],  
National Institute for Health Research (NIHR) [fnd],  
Rodrigo Pires [rev],  
Margaret Siple [rev]

**Maintainer** T. Phuong Quan <phuongquan567@outlook.com>

**Repository** CRAN

**Date/Publication** 2026-02-11 20:40:03 UTC

## Contents

alertspec . . . . .	2
alert_rules . . . . .	3
alert_rule_types . . . . .	4
bespoke_rmd_alert_results . . . . .	7
bespoke_rmd_initialise_widgets . . . . .	9
bespoke_rmd_output . . . . .	10
example_data . . . . .	12
example_prescription_numbers . . . . .	12
inputspec . . . . .	13
mantis_alerts . . . . .	15
mantis_report . . . . .	16
outputspec_interactive . . . . .	19
outputspec_static_heatmap . . . . .	21
outputspec_static_multipanel . . . . .	22

**Index** 24

---

alertspec	<i>Specify alerting rules to be run on the data and displayed in the report</i>
-----------	---

---

## Description

The alert results are displayed in different ways depending on the chosen outputspec. Tabs containing time series which failed at least one alert are highlighted, and a separate tab containing the alert results is created by default.

## Usage

```
alertspec(alert_rules, show_tab_results = c("PASS", "FAIL", "NA"))
```

## Arguments

`alert_rules` `alert_rules()` object specifying conditions to test

`show_tab_results`

only show rows where the alert result is in this vector of values. Alert results can be "PASS", "FAIL", or "NA". If NULL, no separate tab will be created.

## Value

An alertspec() object

**See Also**

[alert\\_rules\(\)](#), [alert\\_rule\\_types\(\)](#)

**Examples**

```
# define some alerting rules
ars <- alert_rules(
  alert_missing(extent_type = "any", extent_value = 1),
  alert_equals(extent_type = "all", rule_value = 0)
)

# specify that all results should be included in the Alerts tab (the default)
alsp <- alertspec(
  alert_rules = ars
)

# specify that only results which fail or are incalculable should be included
# in the Alerts tab
alsp <- alertspec(
  alert_rules = ars,
  show_tab_results = c("FAIL", "NA")
)
```

---

alert_rules	<i>Create set of alert rules</i>
-------------	----------------------------------

---

**Description**

Specify which alert rules should be run on the time series

**Usage**

```
alert_rules(...)
```

**Arguments**

...                    alerts to apply to the time series

**Value**

An alert\_rules object

**See Also**

[alert\\_rule\\_types\(\)](#)

**Examples**

```

# alert if any values are NA
# or if all values are zero
ars <- alert_rules(
  alert_missing(extent_type = "any", extent_value = 1),
  alert_equals(extent_type = "all", rule_value = 0)
)

# alert if any values are over 100, but only for certain antibiotics
ars <- alert_rules(
  alert_above(
    extent_type = "any", extent_value = 1, rule_value = 100,
    items = list("Antibiotic" = c("Coamoxiclav", "Gentamicin"))
  )
)

# alert if any values are over 100, but only for SITE1,
# and only for certain antibiotics
ars <- alert_rules(
  alert_above(
    extent_type = "any", extent_value = 1, rule_value = 100,
    items = list(
      "Location" = "SITE1",
      "Antibiotic" = c("Coamoxiclav", "Gentamicin")
    )
  )
)

```

---

alert_rule_types	<i>Built-in alert rules</i>
------------------	-----------------------------

---

**Description**

A range of built-in rules can be run on the time series to test for particular conditions.

**Usage**

```

alert_missing(extent_type = "all", extent_value = 1, items = NULL)

alert_equals(extent_type = "all", extent_value = 1, rule_value, items = NULL)

alert_above(extent_type = "all", extent_value = 1, rule_value, items = NULL)

alert_below(extent_type = "all", extent_value = 1, rule_value, items = NULL)

alert_difference_above_perc(
  current_period,
  previous_period,

```

```

    rule_value,
    items = NULL
)

alert_difference_below_perc(
    current_period,
    previous_period,
    rule_value,
    items = NULL
)

alert_custom(short_name, description, expression, items = NULL)

```

### Arguments

extent_type	Type of subset of the time series values that must satisfy the condition for the rule to return "FAIL". One of "all", "any", "last", "consecutive". See Details.
extent_value	Numeric lower limit of the extent type. See Details.
items	Named list with names corresponding to members of item_cols. List members are character vectors of values contained in the named column that the rule should be applied to. If items = NULL the rule will be applied to all items. See Details.
rule_value	Numeric value to test against. See Details.
current_period	Numeric vector containing positions from end of time series to use for comparison
previous_period	Numeric vector containing positions from end of time series to use for comparison. Can overlap with current_period if desired.
short_name	Short name to uniquely identify the rule. Only include alphanumeric, '-', and '_' characters.
description	Short description of what the rule checks for
expression	Quoted expression containing the call to be evaluated per item, that returns either TRUE or FALSE. Return value of TRUE means alert result is "FAIL". See Details.

### Value

An alert\_rule object

### Details

Tolerance can be adjusted using the extent\_type and extent\_value parameters, e.g. extent\_type="all" means alert if all values satisfy the condition, extent\_type="any" in combination with extent\_value=5 means alert if there are 5 or more values that satisfy the condition, in any position. Also see Examples.

Use items to restrict the rule to be applied only to specified items. items can either be NULL or a named list of character vectors. If NULL, the rule will be applied to all items. If a named list, the names must match members of the item\_cols parameter in the inputspec, (as well as column

names in the df), though can be a subset. If an `item_col` is not named in the list, the rule will apply to all its members. If an `item_col` is named in the list, the rule will only be applied when the `item_col`'s value is contained in the corresponding character vector. When multiple `item_cols` are specified, the rule will be applied only to items that satisfy all the conditions. See Examples in [alert\\_rules\(\)](#)

`alert_missing()` - Test for the presence of NA values.

`alert_equals()` - Test for the presence of values equal to `rule_value`.

`alert_above()` - Test for the presence of values strictly greater than `rule_value`.

`alert_below()` - Test for the presence of values strictly less than `rule_value`.

`alert_difference_above_perc()` - Test if latest values are greater than in a previous period, increasing strictly more than the percentage stipulated in `rule_value`. Based on the mean of values in the two periods. Ranges should be contiguous, and denote positions from the end of the time series.

`alert_difference_below_perc()` - Test if latest values are lower than in a previous period, dropping strictly more than the percentage stipulated in `rule_value`. Based on the mean of values in the two periods. Ranges should be contiguous, and denote positions from the end of the time series.

`alert_custom()` - Specify a custom rule. The supplied expression is passed to `eval()` within a `dplyr::summarise()` after grouping by the `item_cols` and ordering by the `timepoint_col`. Column names that can be used explicitly in the expression are `value` and `timepoint`, and which refer to the values in the `value_col` and `timepoint_col` columns of the data respectively. See Examples.

### See Also

[alert\\_rules\(\)](#), [alertspect\(\)](#)

### Examples

```
# alert if all values are NA
ars <- alert_rules(alert_missing(extent_type = "all"))

# alert if there are 10 or more missing values in total
# or if the last 3 or more values are missing
# or if 5 or more values in a row are missing
ars <- alert_rules(
  alert_missing(extent_type = "any", extent_value = 10),
  alert_missing(extent_type = "last", extent_value = 3),
  alert_missing(extent_type = "consecutive", extent_value = 5)
)

# alert if any values are zero
ars <- alert_rules(alert_equals(extent_type = "any", rule_value = 0))

# alert if all values are greater than 50
ars <- alert_rules(alert_above(extent_type = "all", rule_value = 50))

# alert if all values are less than 2
ars <- alert_rules(alert_below(extent_type = "all", rule_value = 2))
```

```

# alert if mean of last 3 values is over 20% greater
# than mean of the previous 12 values
ars <- alert_rules(
  alert_difference_above_perc(
    current_period = 1:3,
    previous_period = 4:15,
    rule_value = 20)
)

# alert if mean of last 3 values is over 20% lower than mean of
# the previous 12 values
ars <- alert_rules(
  alert_difference_below_perc(
    current_period = 1:3,
    previous_period = 4:15,
    rule_value = 20)
)

# Create two custom rules
ars <- alert_rules(
  alert_custom(
    short_name = "my_rule_combo",
    description = "Over 3 missing values and max value is > 10",
    expression = quote(
      sum(is.na(value)) > 3 && max(value, na.rm = TRUE) > 10
    )
  ),
  alert_custom(
    short_name = "my_rule_doubled",
    description = "Last value is over double the first value",
    expression = quote(rev(value)[1] > 2 * value[1])
  )
)

```

---

bespoke\_rmd\_alert\_results

*Dynamically generate a table containing alert results for an rmd chunk*

---

## Description

Add mantis alert results to an existing rmarkdown report. The function writes directly to the chunk using side-effects, so chunk options must contain `results = 'asis'`.

## Usage

```

bespoke_rmd_alert_results(
  df,

```

```

    inputspec,
    alert_rules,
    filter_results = c("PASS", "FAIL", "NA"),
    timepoint_limits = c(NA, NA),
    fill_with_zero = FALSE,
    tab_name = NULL,
    tab_level = 1
  )

```

## Arguments

<code>df</code>	A data frame containing multiple time series in long format. See Details.
<code>inputspec</code>	<code>inputspec()</code> object specifying which columns in the supplied <code>df</code> represent the "timepoint", "item", and "value" for the time series. Any "tab" column specification will be ignored.
<code>alert_rules</code>	<code>alert_rules()</code> object specifying conditions to test
<code>filter_results</code>	only return rows where the alert result is in this vector of values. Alert results can be "PASS", "FAIL", or "NA".
<code>timepoint_limits</code>	Set start and end dates for time period to include. Defaults to min/max of <code>timepoint_col</code>
<code>fill_with_zero</code>	Logical. Replace any missing or NA values with 0? Useful when <code>value_col</code> is a record count
<code>tab_name</code>	Character string to appear on the tab label. If omitted or NULL, only the content (and not the parent tab) will be created.
<code>tab_level</code>	integer specifying the nesting level of the tab. If <code>tab_name</code> is specified, a value of 1 generates a tab at rmd level "###". If <code>tab_name</code> is not specified, this is ignored.

## Value

(invisibly) the supplied `df`

## Examples

```

## Not run:

# put this inside a chunk in the rmd file,
# with chunk option `results = 'asis'`
mantis::bespoke_rmd_alert_results(
  df = mantis::example_prescription_numbers,
  inputspec = mantis::inputspec(
    timepoint_col = "PrescriptionDate",
    item_cols = c("Location", "Antibiotic"),
    value_col = "NumberOfPrescriptions",
    tab_col = "Location"
  ),
  alert_rules = alert_rules(

```

```

      alert_missing(extent_type = "any", extent_value = 1),
      alert_equals(extent_type = "all", rule_value = 0)
    ),
    filter_results = c("FAIL", "NA"),
    fill_with_zero = FALSE,
    tab_name = "Failed alerts",
    tab_level = 1
  )

## End(Not run)

```

---

```

bespoke_rmd_initialise_widgets
  Initialise HTML widgets

```

---

## Description

Since the output is being constructed in `results='asis'` chunks, there must also be at least one standard chunk that contains the relevant widgets, otherwise they will fail to render. The dygraph also needs to be initialised with the appropriate `plot_type`. This is only needed when creating interactive reports. Make sure you read the vignette: `vignette("bespoke-reports", package = "mantis")` as it contains further important information. Note: The chunk currently appears like a line break when rendered. See <https://github.com/rstudio/rmarkdown/issues/1877> for more info.

## Usage

```
bespoke_rmd_initialise_widgets(plot_type)
```

## Arguments

`plot_type` "bar" or "line", depending on what will be used in real tables. Or "none" if just want a reactable widget without dygraphs e.g. for alerts

## Value

A (mostly) invisible html widget

## Examples

```

## Not run:
# put this inside its own chunk in the rmd file
# it ensures that the dygraphs render when built using `cat()`
# set the plot_type to the same plot_type as the real output
mantis::bespoke_rmd_initialise_widgets(plot_type = "bar")

## End(Not run)

```

---

bespoke\_rmd\_output      *Dynamically generate mantis output for an rmd chunk*

---

### Description

Add mantis tabs and visualisations to an existing rmarkdown report. The function writes directly to the chunk using side-effects, so chunk options must contain `results = 'asis'`. Make sure you read the vignette: `vignette("bespoke-reports", package = "mantis")` as it contains further important information.

### Usage

```
bespoke_rmd_output(
  df,
  inputspec,
  outputspec,
  alertspec = NULL,
  timepoint_limits = c(NA, NA),
  fill_with_zero = FALSE,
  tab_name = NULL,
  tab_level = 1
)
```

### Arguments

<code>df</code>	A data frame containing multiple time series in long format. See Details.
<code>inputspec</code>	<a href="#">inputspec()</a> object specifying which columns in the supplied <code>df</code> represent the "timepoint", "item", and "value" for the time series. A separate tab will be created for each distinct value in the "tab" column.
<code>outputspec</code>	<code>outputspec</code> object specifying the desired format of the html table(s)/plot(s). If not supplied, default values will be used.
<code>alertspec</code>	<a href="#">alertspec()</a> object specifying conditions to test and display
<code>timepoint_limits</code>	Set start and end dates for time period to include. Defaults to min/max of <code>timepoint_col</code> . If the <code>timepoint_unit</code> of <code>inputspec()</code> is a "day" or longer, this must be a Date type, otherwise it should be a POSIXt type.
<code>fill_with_zero</code>	Logical. Replace any missing or NA values with 0? Useful when <code>value_col</code> is a record count
<code>tab_name</code>	Character string to appear on the tab label. If omitted or NULL, only the content/child tabs (and not the parent tab) will be created.
<code>tab_level</code>	integer specifying the nesting level of the tab. If <code>tab_name</code> is specified, a value of 1 generates a tab at rmd level "##", and any <code>tab_col</code> tabs at a level down. If <code>tab_name</code> is not specified, any <code>tab_col</code> tabs will be created at rmd level "##".

## Details

You can:

- add a single visualisation, with or without creating the container tab, or
- add a set of tabs, each based on the same output specification, with or without creating the parent tab.

The supplied data frame should contain multiple time series in long format, i.e.:

- one "timepoint" (date/posixt) column which will be used for the x-axes. Values should follow a regular pattern, e.g. daily or monthly, but do not have to be consecutive.
- one or more "item" (character) columns containing categorical values identifying distinct time series.
- one "value" (numeric) column containing the time series values which will be used for the y-axes.

The `inputspec` parameter maps the data frame columns to the above. Optionally, if there are multiple columns specified in `item_cols`, one of them can be used to group the time series into different child tabs, by using the `tab_col` parameter.

## Value

(invisibly) the supplied df

## See Also

[bespoke\\_rmd\\_initialise\\_widgets\(\)](#)

## Examples

```
## Not run:

# put this inside a chunk in the rmd file,
# with chunk option `results = 'asis'`
mantis::bespoke_rmd_output(
  df = mantis::example_prescription_numbers,
  inputspec = mantis::inputspec(
    timepoint_col = "PrescriptionDate",
    item_cols = c("Location", "Antibiotic"),
    value_col = "NumberOfPrescriptions",
    tab_col = "Location"
  ),
  outputspec = mantis::outputspec_interactive(
    plot_value_type = "value",
    plot_type = "bar",
    item_labels = c("Antibiotic" = "Antibiotic name"),
    plot_label = "Prescriptions over time",
    sync_axis_range = FALSE,
    item_order = list("Location" = c("SITE3", "SITE2", "SITE1"))
  ),
  timepoint_limits = c(NA, Sys.Date()),
```

```
fill_with_zero = FALSE,  
tab_name = "Group of child tabs",  
tab_level = 1  
)  
  
## End(Not run)
```

---

example\_data

*Example data frame containing multiple time series in long format*

---

### **Description**

Simulated data to cover a range of different behaviours of time series

### **Usage**

```
example_data
```

### **Format**

example\_data:

A data frame with 3,903 rows and 4 columns:

- timepoint - Dates for the time series
- item - Labels to identify the different time series
- value - Values for the time series
- tab - Labels to group related time series into tabs

---

example\_prescription\_numbers

*Example data frame containing numbers of antibiotic prescriptions in long format*

---

### **Description**

Simulated data to demonstrate package usage

### **Usage**

```
example_prescription_numbers
```

**Format**

example\_prescription\_numbers:

A data frame with 6,570 rows and 4 columns:

- PrescriptionDate - The date the prescriptions were written
- Antibiotic - The name of the antibiotic prescribed
- Spectrum - The spectrum of activity of the antibiotic. This value is always the same for a particular antibiotic
- NumberOfPrescriptions - The number of prescriptions written for this antibiotic on this day
- Location - The hospital site where the prescription was written

---

inputspec

*Specify relevant columns in the source data frame*

---

**Description**

Specify relevant columns in the source data frame

**Usage**

```
inputspec(
  timepoint_col,
  item_cols,
  value_col,
  tab_col = NULL,
  timepoint_unit = "day"
)
```

**Arguments**

timepoint_col	String denoting the (date/posixt) column which will be used for the x-axes.
item_cols	String denoting the (character) column containing categorical values identifying distinct time series. Multiple columns that together identify a time series can be provided as a vector
value_col	String denoting the (numeric) column containing the time series values which will be used for the y-axes.
tab_col	Optional. String denoting the (character) column containing categorical values which will be used to group the time series into different tabs on the report.
timepoint_unit	expected pattern of the timepoint_col values. "sec"/"min"/"hour"/"day"/"month"/"quarter"/"year". This will be used to fill in any gaps in the time series.

**Value**

A inputspec() object

**Examples**

```
# create a flat report, and include the "Location" and "Antibiotic" fields
# in the content
inspec_flat <- inputspec(
  timepoint_col = "PrescriptionDate",
  item_cols = c("Location", "Antibiotic"),
  value_col = "NumberOfPrescriptions",
  timepoint_unit = "day"
)

# create a flat report, and include the "Location", "Spectrum",
# and "Antibiotic" fields in the content
inspec_flat2 <- inputspec(
  timepoint_col = "PrescriptionDate",
  item_cols = c("Location", "Spectrum", "Antibiotic"),
  value_col = "NumberOfPrescriptions",
  timepoint_unit = "day"
)

# create a tabbed report, with a separate tab for each unique value of
# "Location", and include just the "Antibiotic" field in the content of
# each tab
inspec_tabbed <- inputspec(
  timepoint_col = "PrescriptionDate",
  item_cols = c("Antibiotic", "Location"),
  value_col = "NumberOfPrescriptions",
  tab_col = "Location",
  timepoint_unit = "day"
)

# create a tabbed report, with a separate tab for each unique value of
# "Location", and include the "Antibiotic" and "Spectrum" fields in the
# content of each tab
inspec_tabbed2 <- inputspec(
  timepoint_col = "PrescriptionDate",
  item_cols = c("Antibiotic", "Spectrum", "Location"),
  value_col = "NumberOfPrescriptions",
  tab_col = "Location",
  timepoint_unit = "day"
)

# create a tabbed report, with a separate tab for each unique value of
# "Antibiotic", and include just the "Location" field in the content of
# each tab
inspec_tabbed3 <- inputspec(
  timepoint_col = "PrescriptionDate",
  item_cols = c("Antibiotic", "Location"),
  value_col = "NumberOfPrescriptions",
  tab_col = "Antibiotic",
  timepoint_unit = "day"
)
```

---

mantis_alerts	<i>Generate a data frame containing alert results</i>
---------------	---

---

### Description

Test the time series for a set of conditions without generating an html report. This can be useful for incorporation into a pipeline.

### Usage

```
mantis_alerts(
  df,
  inputspec,
  alert_rules,
  filter_results = c("PASS", "FAIL", "NA"),
  timepoint_limits = c(NA, NA),
  fill_with_zero = FALSE
)
```

### Arguments

df	A data frame containing multiple time series in long format. See Details.
inputspec	<a href="#">inputspec()</a> object specifying which columns in the supplied df represent the "timepoint", "item", and "value" for the time series.
alert_rules	<a href="#">alert_rules()</a> object specifying conditions to test
filter_results	Only return rows where the alert result is in this vector of values. Alert results can be "PASS", "FAIL", or "NA".
timepoint_limits	Set start and end dates for time period to include. Defaults to min/max of timepoint_col. Can be either Date values or NAs.
fill_with_zero	Logical. Replace any missing or NA values with 0? Useful when value_col is a record count.

### Details

The supplied data frame should contain multiple time series in long format, i.e.:

- one "timepoint" (date/posixt) column which will be used for the x-axes. Values should follow a regular pattern, e.g. daily or monthly, but do not have to be consecutive.
- one or more "item" (character) columns containing categorical values identifying distinct time series.
- one "value" (numeric) column containing the time series values which will be used for the y-axes.

The inputspec parameter maps the data frame columns to the above.

**Value**

tibble

**See Also**

[alert\\_rules\(\)](#), [inputspec\(\)](#), [alert\\_rule\\_types\(\)](#)

**Examples**

```
alert_results <- mantis_alerts(  
  example_prescription_numbers,  
  inputspec = inputspec(  
    timepoint_col = "PrescriptionDate",  
    item_cols = c("Antibiotic", "Location"),  
    value_col = "NumberOfPrescriptions"  
  ),  
  alert_rules = alert_rules(  
    alert_missing(extent_type = "any", extent_value = 1),  
    alert_equals(extent_type = "all", rule_value = 0)  
  )  
)
```

---

mantis\_report

*Create an interactive time series report from a data frame*

---

**Description**

Accepts a data frame containing multiple time series in long format, generates a collection of interactive time series plots for visual inspection, and saves the report to disk.

**Usage**

```
mantis_report(  
  df,  
  file,  
  inputspec,  
  outputspec = NULL,  
  alertspec = NULL,  
  report_title = "mantis report",  
  dataset_description = "",  
  add_timestamp = FALSE,  
  show_progress = TRUE,  
  ...  
)
```

**Arguments**

<code>df</code>	A data frame containing multiple time series in long format. See Details.
<code>file</code>	String specifying the desired file name (and path) to save the report to. The file name should include the extension ".html". If only a file name is supplied, the report will be saved in the current working directory. If a path is supplied, the directory should already exist. Any existing file of the same name will be overwritten unless <code>add_timestamp</code> is set to <code>TRUE</code> .
<code>inputspec</code>	<code>inputspec()</code> object specifying which columns in the supplied <code>df</code> represent the "timepoint", "item", "value" and (optionally) "tab" for the time series. If a "tab" column is specified, a separate tab will be created for each distinct value in the column.
<code>outputspec</code>	<code>outputspec</code> object specifying the desired format of the html table(s). If not supplied, default values will be used.
<code>alertspec</code>	<code>alertspec()</code> object specifying conditions to test and display.
<code>report_title</code>	Title to appear on the report.
<code>dataset_description</code>	Short description of the dataset being shown. This will appear on the report.
<code>add_timestamp</code>	Append a timestamp to the end of the filename with format <code>_YYMMDD_HHMMSS</code> . This can be used to keep multiple versions of the same report. Default = <code>FALSE</code> .
<code>show_progress</code>	Print progress to console. Default = <code>TRUE</code> .
<code>...</code>	Further parameters to be passed to <code>rmarkdown::render()</code> . Cannot include any of <code>input</code> , <code>output_dir</code> , <code>output_file</code> , <code>params</code> , <code>quiet</code> .

**Details**

The supplied data frame should contain multiple time series in long format, i.e.:

- one "timepoint" (date/posixt) column which will be used for the x-axes. Values should follow a regular pattern, e.g. daily or monthly, but do not have to be consecutive.
- one or more "item" (character) columns containing categorical values identifying distinct time series.
- one "value" (numeric) column containing the time series values which will be used for the y-axes.
- Optionally, a "tab" (character) column containing categorical values which will be used to group the time series into different tabs on the report.

The `inputspec` parameter maps the data frame columns to the above.

**Value**

A string containing the name and full path of the saved report.

**See Also**

`inputspec()`, `outputspec_interactive()`, `outputspec_static_heatmap()`, `outputspec_static_multipanel()`, `alertspec()`

**Examples**

```

# create an interactive report in the temp directory,
# with one tab per Location
filename <- mantis_report(
  df = example_prescription_numbers,
  file = file.path(tempdir(), "example_prescription_numbers_interactive.html"),
  inputspec = inputspec(
    timepoint_col = "PrescriptionDate",
    item_cols = c("Location", "Antibiotic", "Spectrum"),
    value_col = "NumberOfPrescriptions",
    tab_col = "Location",
    timepoint_unit = "day"
  ),
  outputspec = outputspec_interactive(),
  report_title = "Daily antibiotic prescribing",
  dataset_description = "Antibiotic prescriptions by site",
  show_progress = TRUE
)

filename

```

```

# create an interactive report in the temp directory, with alerting rules
filename <- mantis_report(
  df = example_prescription_numbers,
  file = file.path(tempdir(), "example_prescription_numbers_interactive.html"),
  inputspec = inputspec(
    timepoint_col = "PrescriptionDate",
    item_cols = c("Location", "Antibiotic", "Spectrum"),
    value_col = "NumberOfPrescriptions",
    tab_col = "Location",
    timepoint_unit = "day"
  ),
  outputspec = outputspec_interactive(),
  alertspec = alertspec(
    alert_rules = alert_rules(
      alert_missing(extent_type = "any", extent_value = 1),
      alert_equals(extent_type = "all", rule_value = 0)
    ),
    show_tab_results = c("FAIL", "NA")
  ),
  report_title = "Daily antibiotic prescribing",
  dataset_description = "Antibiotic prescriptions by site",
  show_progress = TRUE
)

filename

```

---

 outputspec\_interactive

*Specify output options for an interactive report*


---

## Description

Each tab contains a single table with one row per time series, and sortable/filterable columns based on the `item_cols` parameter of the `inputspec()`. The time series plots have tooltips and can be zoomed in by selecting an area of the plot.

## Usage

```
outputspec_interactive(
  plot_value_type = "value",
  plot_type = "bar",
  item_labels = NULL,
  plot_label = NULL,
  summary_cols = c("max_value"),
  sync_axis_range = FALSE,
  item_order = NULL,
  sort_by = NULL
)
```

## Arguments

<code>plot_value_type</code>	Display the raw "value" for the time series or display the calculated "delta" between consecutive values.
<code>plot_type</code>	Display the time series as a "bar" or "line" chart.
<code>item_labels</code>	Named vector containing string label(s) to use for the "item" column(s) in the report. The names should correspond to the <code>item_cols</code> , and the values should contain the desired labels. If <code>NULL</code> , the original columns name(s) will be used.
<code>plot_label</code>	String label to use for the time series column in the report. If <code>NULL</code> , the original <code>value_col</code> name will be used.
<code>summary_cols</code>	Summary data to include as columns in the report. Options are <code>c("max_value", "last_value", "last_value_nonmissing", "last_timepoint", "mean_value")</code> .
<code>sync_axis_range</code>	Set the y-axis to be the same range for all time series in a table. X-axes are always synced. Logical.
<code>item_order</code>	named list corresponding to <code>item_cols</code> columns for ordering the items in the output. List values are either <code>TRUE</code> for ascending order, or a character vector of values contained in the named column for explicit ordering. If <code>item_order = NULL</code> , the original order will be kept. See Details.
<code>sort_by</code>	column in output table to sort by. Can be one of <code>alert_overall</code> , or one of the summary columns. Append a minus sign to sort in descending order e.g. <code>-max_value</code> . Secondary ordering will be based on <code>item_order</code> .

**Value**

An outputspec() object

**Details**

For `item_order`, the names of the list members should correspond to the column names in the `df`. Any names that don't match will be ignored. When multiple columns are specified, they are sorted together, in the same priority order as the list. If a list item is `TRUE` then that column is sorted in ascending order. If a list item is a character vector then that column is sorted in the order of the vector first, with any remaining values included alphabetically at the end. If you want to order the tabs, it is recommended to put the `tab_col` as the first item in the list.

**See Also**

[outputspec\\_static\\_heatmap\(\)](#), [outputspec\\_static\\_multipanel\(\)](#)

**Examples**

```
# Set explicit labels for the column headings
outspec <- outputspec_interactive(
  item_labels = c("Antibiotic" = "ABX", "Location" = "Which site?"),
  plot_label = "Daily records"
)

## Change the sort order that the items appear in the table

# Sort alphabetically by Antibiotic
outspec <- outputspec_interactive(
  item_order = list("Antibiotic" = TRUE)
)

# Sort alphabetically by Location first,
# then put "Vancomycin" and "Linezolid" before other antibiotics
outspec <- outputspec_interactive(
  item_order = list("Location" = TRUE,
    "Antibiotic" = c("Vancomycin", "Linezolid"))
)

# Put the time series with the largest values first
outspec <- outputspec_interactive(
  sort_by = "-max_value"
)

# Put the time series with failed alerts first
outspec <- outputspec_interactive(
  sort_by = "alert_overall"
)

# Put the time series with failed alerts first,
# then sort alphabetically by Antibiotic
outspec <- outputspec_interactive(
```

```
  item_order = list("Antibiotic" = TRUE),
  sort_by = "alert_overall"
)
```

---

outputspec\_static\_heatmap

*Specify output options for a static report containing heatmaps*

---

## Description

Each tab contains a heatmap with one row per time series.

## Usage

```
outputspec_static_heatmap(
  fill_colour = "blue",
  y_label = NULL,
  item_order = NULL
)
```

## Arguments

<code>fill_colour</code>	colour to use for the tiles. Passed to <code>high</code> parameter of <code>ggplot2::scale_fill_gradient()</code>
<code>y_label</code>	string for y-axis label. Optional. If <code>NULL</code> , the label will be constructed from the <code>inputspec()</code>
<code>item_order</code>	named list corresponding to <code>item_cols</code> columns for ordering the items in the output. List values are either <code>TRUE</code> for ascending order, or a character vector of values contained in the named column for explicit ordering. If <code>item_order = NULL</code> , the original order will be kept. See Details.

## Value

An `outputspec()` object

## Details

For `item_order`, the names of the list members should correspond to the column names in the `df`. Any names that don't match will be ignored. When multiple columns are specified, they are sorted together, in the same priority order as the list. If a list item is `TRUE` then that column is sorted in ascending order. If a list item is a character vector then that column is sorted in the order of the vector first, with any remaining values included alphabetically at the end. If you want to order the tabs, it is recommended to put the `tab_col` as the first item in the list.

## See Also

[outputspec\\_interactive\(\)](#), [outputspec\\_static\\_multipanel\(\)](#)

**Examples**

```

# Customise the plot
outspec <- outputspec_static_heatmap(
  fill_colour = "#56B1F7",
  y_label = "Daily records"
)

# Sort alphabetically by Antibiotic
outspec <- outputspec_static_heatmap(
  item_order = list("Antibiotic" = TRUE)
)

# Sort alphabetically by Location first,
# then put "Vancomycin" and "Linezolid" before other antibiotics
outspec <- outputspec_static_heatmap(
  item_order = list("Location" = TRUE,
    "Antibiotic" = c("Vancomycin", "Linezolid"))
)

```

---

outputspec\_static\_multipanel

*Specify output options for a static report containing a panel of plots.*

---

**Description**

Each tab contains a single column of scatter plots with one row per time series.

**Usage**

```

outputspec_static_multipanel(
  sync_axis_range = FALSE,
  y_label = NULL,
  item_order = NULL
)

```

**Arguments**

sync_axis_range	Set the y-axis to be the same range for all the plots. X-axes are always synced.
y_label	string for y-axis label. Optional. If NULL, the label will be constructed from the inputspec()
item_order	named list corresponding to item_cols columns for ordering the items in the output. List values are either TRUE for ascending order, or a character vector of values contained in the named column for explicit ordering. If item_order = NULL, the original order will be kept. See Details.

**Value**

An outputspec() object

## Details

For `item_order`, the names of the list members should correspond to the column names in the `df`. Any names that don't match will be ignored. When multiple columns are specified, they are sorted together, in the same priority order as the list. If a list item is `TRUE` then that column is sorted in ascending order. If a list item is a character vector then that column is sorted in the order of the vector first, with any remaining values included alphabetically at the end. If you want to order the tabs, it is recommended to put the `tab_col` as the first item in the list.

## See Also

[outputspec\\_interactive\(\)](#), [outputspec\\_static\\_heatmap\(\)](#)

## Examples

```
# Plot all panels to same scale
outputspec <- outputspec_static_multipanel(
  sync_axis_range = TRUE,
  y_label = "Daily records"
)

# Sort panels alphabetically by Antibiotic
outputspec <- outputspec_static_multipanel(
  item_order = list("Antibiotic" = TRUE)
)

# Sort alphabetically by Location first,
# then put "Vancomycin" and "Linezolid" before other antibiotics
outputspec <- outputspec_static_multipanel(
  item_order = list("Location" = TRUE,
    "Antibiotic" = c("Vancomycin", "Linezolid"))
)
```

# Index

## \* datasets

example\_data, [12](#)  
example\_prescription\_numbers, [12](#)

alert\_above(alert\_rule\_types), [4](#)  
alert\_below(alert\_rule\_types), [4](#)  
alert\_custom(alert\_rule\_types), [4](#)  
alert\_difference\_above\_perc  
    (alert\_rule\_types), [4](#)  
alert\_difference\_below\_perc  
    (alert\_rule\_types), [4](#)  
alert\_equals(alert\_rule\_types), [4](#)  
alert\_missing(alert\_rule\_types), [4](#)  
alert\_rule\_types, [4](#)  
alert\_rule\_types(), [3](#), [16](#)  
alert\_rules, [3](#)  
alert\_rules(), [2](#), [3](#), [6](#), [8](#), [15](#), [16](#)  
alertspect, [2](#)  
alertspect(), [6](#), [10](#), [17](#)

bespoke\_rmd\_alert\_results, [7](#)  
bespoke\_rmd\_initialise\_widgets, [9](#)  
bespoke\_rmd\_initialise\_widgets(), [11](#)  
bespoke\_rmd\_output, [10](#)

example\_data, [12](#)  
example\_prescription\_numbers, [12](#)

inputspec, [13](#)  
inputspec(), [8](#), [10](#), [15–17](#)

mantis\_alerts, [15](#)  
mantis\_report, [16](#)

outputspec\_interactive, [19](#)  
outputspec\_interactive(), [17](#), [21](#), [23](#)  
outputspec\_static\_heatmap, [21](#)  
outputspec\_static\_heatmap(), [17](#), [20](#), [23](#)  
outputspec\_static\_multipanel, [22](#)  
outputspec\_static\_multipanel(), [17](#), [20](#),  
[21](#)