Package: ATO (via r-universe)

November 14, 2026

Title Animal Tracking Object Class

Version 0.0.0.9006

Description Class and methods to create, edit, and expand an ATO.

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Language en-GB

Depends R (>= 4.0.0)

Imports methods, stats, utils

Suggests data.table, tibble, testthat, knitr, quarto, rmarkdown

URL https://github.com/trackyverse/ATO, http://ato.trackyverse.org, https://trackyverse.github.io/ATO/

BugReports https://github.com/trackyverse/ATO/issues

VignetteBuilder quarto

Config/Needs/website quarto

Repository https://trackyverse.r-universe.dev

Date/Publication 2026-11-14 18:10:43 UTC

RemoteUrl https://github.com/trackyverse/ATO

RemoteRef HEAD

RemoteSha cf53b350eafca457b9c3ff2374ac8d31bd4fd3e3

Contents

$,ATO-method 3
ATO-class 3
ATO_ani 4

ato_debug . . . 5
ato_debug_header . . . 5
ATO_dep . . . 6
ATO_det . . . 7
ATO_log . . . 7
ato_match_immediate . . . 8
ATO_obs . . . 9
ato_table_type_global . . . 10
ATO_tag . . . 10
get_ani . . . 11
get_dep . . . 12
get_det . . . 13
get_det_dep . . . 14
get_det_tag . . . 15
get_log . . . 16
get_obs . . . 17
get_tag . . . 18
has . . . 19
init_ato . . . 20
is_ato . . . 21
log_comment . . . 21
log_event . . . 22
make_ani . . . 23
make_dep . . . 24
make_det . . . 26
make_obs . . . 28
make_tag . . . 29
match_update . . . 31
show,ATO-method . . . 32
summary,ATO_ani-method . . . 32
summary,ATO_dep-method . . . 33
summary,ATO_det-method . . . 33
summary,ATO_log-method . . . 34
summary,ATO_obs-method . . . 34
summary,ATO_tag-method . . . 35
table_type . . . 36
table_type<- . . . 37
tzone . . . 38
tzone<- . . . 39

Index **41**

$,ATO-method *$ Method to extract ATO slots directly*

Description

$ Method to extract ATO slots directly

Usage

```
## S4 method for signature 'ATO'
x$name
```

Arguments

x	an ATO object
name	the name of the desired ATO slot

Value

The slot as defined in name, along with a warning to directly extract slots in the future.

See Also

Extract

Examples

```
# access a slot using $ instead of @
dep <- example_ato$dep
summary(dep)

# clean up
rm(dep)
```

ATO-class *S4 class: ATO*

Description

The Animal Tracking Object (ATO) class contains slots that allow researchers to package their study data in a standard format, to be used by the R packages that collectively make the trackyverse.

Slots

`ani` The animals slot. See `ATO_ani`.

`dep` The deployments slot. See `ATO_dep`.

`det` The detections slot. See `ATO_det`.

`obs` The observations slot. See `ATO_obs`.

`tag` The tags slot. See `ATO_tag`.

`log` The log slot. See `ATO_log`.

`pkg` A list slot reserved for package outputs.

`ATO_ani` *S4 class: ATO_ani*

Description

An S4 class for the `ATO` animals (@ani). Does not contain internal slots. It is a table of either data.frame, data.table, or tibble format.

Usage

```
.ATO_ani
```

Format

See `make_ani`.

An object of class `ATO_ani` (inherits from `data.frame`) with 0 rows and 10 columns.

Details

The prototype @ani slot contains the standard columns of the @ani slot. Other columns are allowed, but these are necessary, and must be of the designated types.

The ani slot contains information on the animals tagged.

Can be of type data.frame, data.table, or tibble, depending on the @tbl slot. See `table_type` for more details on `ATO` table types.

See Also

`make_ani`

ato_debug *Helper function to turn debug mode on or off*

Description

Helper function to turn debug mode on or off

Usage

```
ato_debug(value = c(TRUE, FALSE))
```

Arguments

value	logical. Should matching be done on the fly?

Value

the current value for ATO_match_immediate if no new value is provided

Examples

```
# Use without arguments to verify the current state, or
# with TRUE of FALSE to activate/deactivate debug mode
ato_debug()
```

ato_debug_header *Helper function send a message at the start of each function*

Description

This function is intended for developers. If you want to benefit from the same debug properties as the ATO functions, include a call to this function at the head of your own.

Usage

```
ato_debug_header()
```

Value

nothing, called for side effects

Examples

```
# include this at the top of your function. e.g.
x <- function(arg) {
  ato_debug_header()
  return(arg * 2)
}

# and then use ato_debug to get a message when this function starts
old_ato_debug <- ato_debug()
ato_debug(TRUE)
x(arg = 2)
ato_debug(old_ato_debug)
```

ATO_dep *S4 class: ATO_dep*

Description

An S4 class for the ATO deployments (@dep). Does not contain internal slots. It is a table of either data.frame, data.table, or tibble format.

Usage

```
.ATO_dep
```

Format

See make_dep.

An object of class ATO_dep (inherits from data.frame) with 0 rows and 18 columns.

Details

The prototype @dep slot contains the standard columns of the @dep slot. Other columns are allowed, but these are necessary, and must be of the designated types.

Deployments include both the deployments of receivers (where the receiver information is filled but the transmitter information isn't); transceivers (where both the receiver and the transmitter information are filled in; i.e. a receiver with a beacon tag); and reference tags (where there is no receiver information, but there is transmitter information).

Can be of type data.frame, data.table, or tibble, depending on the @tbl slot. See table_type for more details on ATO table types.

See Also

make_dep

ATO_det *S4 class: ATO_det*

Description

An S4 class for the ATO detections (@det). Does not contain internal slots. It is a table of either data.frame, data.table, or tibble format.

Usage

```
.ATO_det
```

Format

See make_det.

An object of class ATO_det (inherits from data.frame) with 0 rows and 6 columns.

Details

The prototype @det slot contains the standard columns of the @det slot. Other columns are allowed, but these are necessary, and must be of the designated types.

Can be of type data.frame, data.table, or tibble, depending on the @tbl slot. See table_type for more details on ATO table types.

See Also

make_det

ATO_log *S4 class: ATO_log*

Description

An S4 class for the ATO log (@log). Does not contain internal slots. It is a table of either data.frame, data.table, or tibble format.

Usage

```
.ATO_log
```

Format

A data frame with 0 rows and 4 variables:

datetime date and time of the log entry.

type type of log entry.

pkg name of the package that hosts the function that triggered the log entry.

fun name of the function that triggered the log entry.

call full function call, for debugging.

log log entry.

An object of class `ATO_log` (inherits from `data.frame`) with 0 rows and 6 columns.

Details

The prototype @log slot contains the standard columns of the @log slot. Other columns are not allowed.

Contains summary information of the actions performed throughout the life of the ATO.

Can be of type data.frame, data.table, or tibble, depending on the @tbl slot. See `table_type` for more details on `ATO` table types.

`ato_match_immediate` *Helper function to turn automatic matching on or off*

Description

Matching of the slots' contents is essential to keep the ATO operational. Do not turn automatic matching off unless you are working with extremely large datasets that would cause heavy computation delays.

Usage

```
ato_match_immediate(value = TRUE, silent = FALSE)
```

Arguments

`value`	logical. Should matching be done on the fly?
`silent`	Supresses messages

Value

the current value for ATO_match_immediate if no new value is provided

Examples

```
old_match_immediate <- ato_match_immediate()
ato_match_immediate(FALSE)
x <- init_ato(ani = ani(example_ato),
              tag = tag(example_ato),
              det = det(example_ato),
              dep = dep(example_ato))
x <- match_update(x)

# clean up
ato_match_immediate(old_match_immediate)
rm(x, old_match_immediate)
```

`ATO_obs` *S4 class: ATO_obs*

Description

An S4 class for the `ATO` observations (@obs). Does not contain internal slots. It is a table of either data.frame, data.table, or tibble format.

Usage

```
.ATO_obs
```

Format

See `make_obs`.

An object of class `ATO_obs` (inherits from `data.frame`) with 0 rows and 9 columns.

Details

The prototype @obs slot contains the standard columns of the @obs slot. Other columns are allowed, but these are necessary, and must be of the designated types.

The observations slot contains information about locations where an animal was seen or a tag was heard. E.g. fin observations, or detections through manual tracking.

Can be of type data.frame, data.table, or tibble, depending on the @tbl slot. See `table_type` for more details on `ATO` table types.

See Also

`make_obs`

ato_table_type_global *Helper function to set the global type of ATO tables*

Description

Helper function to set the global type of ATO tables

Usage

```
ato_table_type_global(type = c("data.frame", "data.table", "tibble"))
```

Arguments

type	the desired type of ATO tables. One of data.frame (default), data.table (requires the package data.table installed), or tibble (requires the package tibble installed).

Value

the current global table type if 'type' is missing.

Examples

```
old_table_type_global <- ato_table_type_global()
ato_table_type_global("data.frame")

# clean up
ato_table_type_global(old_table_type_global)
rm(old_table_type_global)
```

ATO_tag *S4 class: ATO_tag*

Description

An S4 class for the ATO tags (@tag). Does not contain internal slots. It is a table of either data.frame, data.table, or tibble format.

Usage

```
.ATO_tag
```

Format

See make_tag.

An object of class ATO_tag (inherits from data.frame) with 0 rows and 13 columns.

Details

The prototype @tag slot contains the standard columns of the @tag slot. Other columns are allowed, but these are necessary, and must be of the designated types.

The tag slot contains information on the different transmitters being tracked. Tags with multiple transmitter codes are coded as multiple rows associated with a single serial number and a single animal.

Can be of type data.frame, data.table, or tibble, depending on the @tbl slot. See table_type for more details on ATO table types.

See Also

make_tag

`get_ani` *Generic to extract the ani slot as a table*

Description

Generic to extract the ani slot as a table

Usage

```
get_ani(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
get_ani(x, type = c("all", "valid", "invalid"))

ani(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
ani(x, type = c("all", "valid", "invalid"))
```

Arguments

x	an ATO object
type	the type of rows to return. One of: 'all' - returns both valid and invalid rows (default); 'valid' - returns only valid rows; 'invalid' - returns only invalid rows

Value

The ani slot as a table

Examples

```
# extract all the animals from an ATO in table format
x <- get_ani(example_ato)
summary(x)

# short form:
x <- ani(example_ato)
summary(x)

# clean up
rm(x)
```

---

get_dep *Generic to extract the dep slot as a table*

Description

Generic to extract the dep slot as a table

Usage

```
get_dep(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
get_dep(x, type = c("all", "valid", "invalid"))

dep(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
dep(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
det(x, receivers, transmitters, type = c("all", "valid", "invalid"))
```

Arguments

x	an ATO object
type	the type of rows to return. One of: 'all' - returns both valid and invalid rows (default); 'valid' - returns only valid rows; 'invalid' - returns only invalid rows
receivers	An optional vector of receiver serial numbers from which to extract detections.
transmitters	An optional vector of transmitters for which to extract detections.

Value

The dep slot as a table

Examples

```
# extract all the deployments from an ATO in table format
x <- get_dep(example_ato)
summary(x)

# short form:
x <- dep(example_ato)
summary(x)

# clean up
rm(x)
```

`get_det` *Generic to extract detections as a table*

Description

Generic to extract detections as a table

Usage

```
get_det(x, receivers, transmitters, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
get_det(x, receivers, transmitters, type = c("all", "valid", "invalid"))

det(x, receivers, transmitters, type = c("all", "valid", "invalid"))
```

Arguments

`x`	an `ATO` object
`receivers`	An optional vector of receiver serial numbers from which to extract detections.
`transmitters`	An optional vector of transmitters for which to extract detections.
`type`	the type of rows to return. One of: 'all' - returns both valid and invalid rows (default); 'valid' - returns only valid rows; 'invalid' - returns only invalid rows

Value

a table of detections

Examples

```
# extract all the detections from an ATO in table format
x <- get_det(example_ato)
summary(x)

# short form:
x <- det(example_ato)
summary(x)

# extract only detections from one or more specific receivers
x <- get_det(example_ato, receivers = "132908")
summary(x)

# or matching one or more specific transmitters
x <- get_det(example_ato, transmitters = "R64K-4529")
summary(x)

# or both!
x <- get_det(example_ato,
             receivers = "132908",
             transmitters = "R64K-4529")
summary(x)

# clean up
rm(x)
```

---

get_det_dep *A wrapper of* get_det *to extract detections for transmitters listed in the dep slot*

Description

A wrapper of get_det to extract detections for transmitters listed in the dep slot

Usage

```
get_det_dep(x, receivers, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
get_det_dep(x, receivers, type = c("all", "valid", "invalid"))
```

Arguments

x	an ATO object
receivers	An optional vector of receiver serial numbers from which to extract detections.
type	the type of rows to return. One of: 'all' - returns both valid and invalid rows (default); 'valid' - returns only valid rows; 'invalid' - returns only invalid rows

Value

a table of detections

Examples

```
# wrapper to extract all detections that match transmitters
# listed in the @dep slot (beacon tags)
x <- get_det_dep(example_ato)
summary(x)
# note: The example_ato does not have beacon detections,
# so this returns a table with 0 rows

# extract only detections from one or more specific receivers
x <- get_det_dep(example_ato, receivers = "132908")
summary(x)
# note: The example_ato does not have beacon detections,
# so this returns a table with 0 rows

# clean up
rm(x)
```

---

get_det_tag *A wrapper of* get_det *to extract detections for transmitters listed in the tag slot*

Description

A wrapper of get_det to extract detections for transmitters listed in the tag slot

Usage

```
get_det_tag(x, receivers, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
get_det_tag(x, receivers, type = c("all", "valid", "invalid"))
```

Arguments

x	an ATO object
receivers	An optional vector of receiver serial numbers from which to extract detections.
type	the type of rows to return. One of: 'all' - returns both valid and invalid rows (default); 'valid' - returns only valid rows; 'invalid' - returns only invalid rows

Value

a table of detections

Examples

```
# wrapper to extract all detections that match transmitters
# listed in the @tag slot
x <- get_det_tag(example_ato)

# extract only detections from one or more specific receivers
x <- get_det_tag(example_ato, receivers = "132908")
summary(x)

# clean up
rm(x)
```

---

get_log *Generic to extract the log slot as a table*

Description

Generic to extract the log slot as a table

Usage

```
get_log(x, debug = FALSE)

## S4 method for signature 'ATO'
get_log(x, debug = FALSE)
```

Arguments

x	an ATO object
debug	should debug entries and the call column be displayed?

Value

The log slot as a table

Examples

```
# extract all the log lines from an ATO in table format
log <- get_log(example_ato)
head(log)

# clean up
rm(log)
```

get_obs *Generic to extract the obs slot as a table*

Description

Generic to extract the obs slot as a table

Usage

```
get_obs(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
get_obs(x, type = c("all", "valid", "invalid"))

obs(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
obs(x, type = c("all", "valid", "invalid"))
```

Arguments

x	an ATO object
type	the type of rows to return. One of: 'all' - returns both valid and invalid rows (default); 'valid' - returns only valid rows; 'invalid' - returns only invalid rows

Value

The obs slot as a table

Examples

```
# extract all the observations from an ATO in table format
x <- get_obs(example_ato)
summary(x)

# short form:
x <- obs(example_ato)
summary(x)

# clean up
rm(x)
```

get_tag *Generic to extract the tag slot as a table*

Description

Generic to extract the tag slot as a table

Usage

```
get_tag(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
get_tag(x, type = c("all", "valid", "invalid"))

tag(x, type = c("all", "valid", "invalid"))

## S4 method for signature 'ATO'
tag(x, type = c("all", "valid", "invalid"))
```

Arguments

x	an ATO object
type	the type of rows to return. One of: 'all' - returns both valid and invalid rows (default); 'valid' - returns only valid rows; 'invalid' - returns only invalid rows

Value

The tag slot as a table

Examples

```
# extract all the tags from an ATO in table format
x <- get_tag(example_ato)
summary(x)

# short form:
x <- tag(example_ato)
summary(x)

# clean up
rm(x)
```

has *Generic to check if an ATO has data in a given slot*

Description

Generic to check if an ATO has data in a given slot

Usage

```
has(object, value, column, allow_NA = TRUE, error = FALSE)

## S4 method for signature 'ATO'
has(object, value, column, allow_NA = TRUE, error = FALSE)
```

Arguments

object	an ATO
value	a vector with the names of the requested slots
column	a vector of column names to find within the requested slots
allow_NA	Should NAs be allowed in the inspected columns?
error	Should the code execution stop if the requested slots are empty?

Value

TRUE if the slots have data, FALSE if they're empty and error = FALSE. Throws an error if the slots are empty and error = TRUE.

Examples

```
# check if ATO object has detections
has(example_ato, "det")

# check if ATO has observations
has(example_ato, "obs")

# alternatively, this would error
# has(example_ato, "obs", error = TRUE)

# check specific columns in slot
has(example_ato, "tag", c("release_datetime", "power_level"))
has(example_ato, "tag", c("test", "power_level"))
# has(example_ato, "tag", c("test", "power_level"), error = TRUE)

# check that columns don't have NAs
# has(example_ato, "tag",
#    c("release_datetime", "power_level"),
#    allow_NA = FALSE, error = TRUE)
```

`init_ato` *Initiate an* `ATO`

Description

This is a wrapper around the usual way of creating S4 objects through new(). The main purpose of this function is to ensure the ATO is created with the right table type. See `table_type` for more details.

Usage

```
init_ato(ani, dep, det, obs, tag, silent = FALSE)
```

Arguments

`ani`	an object of class ATO_ani. See `make_ani`.
`dep`	an object of class ATO_dep. See `make_dep`.
`det`	an object of class ATO_det. See `make_det`.
`obs`	an object of class ATO_obs. See `make_obs`.
`tag`	an object of class ATO_tag. See `make_tag`.
`silent`	Supresses summary messages

Value

an `ATO`

Examples

```
# split apart the example ATO
ani <- get_ani(example_ato)
dep <- get_dep(example_ato)
det <- get_det(example_ato)
tag <- get_tag(example_ato)

# and now use the parts to build a new ato
x <- init_ato(ani = ani,
              dep = dep,
              det = det,
              tag = tag)

# clean up
rm(ani, dep, det, tag, x)
```

is_ato *Wrapper for is(x, "ATO")*

Description

This function allows an easy check if the input is an ATO and throw an error if that is not the case.

Usage

```
is_ato(x, error = TRUE)
```

Arguments

x	the object to test
error	Should the code error if x is not an ATO?

Value

TRUE if x is an ATO, FALSE if it is not an ATO and error = FALSE. Throws an error if x is not an ATO and error = TRUE.

Examples

```
# check if object is an ato
is_ato(example_ato)

# by default, errors if not an ATO
# is_ato(1:5)

# But can be set to return FALSE
# is_ato(1:5, FALSE)
```

log_comment *Log a new comment*

Description

Logs a user-written comment into the log.

Usage

```
log_comment(ato, ...)
```

Arguments

ato	the ATO object to which to log the comment
...	The text fragments that compose the comment.

Value

the updated ATO.

Examples

```
# add an event to the ato
x <- log_comment(example_ato, "M: just a comment")
get_log(x)
```

---

log_event *Log a new event*

Description

Logs useful information as the analysis progresses. This function is meant to be used by other functions within the trackyverse. If you are looking for a way to add comments to the analysis, look at log_comment.

Usage

```
log_event(
  ato,
  type = c("log", "message", "msg", "warning", "debug", "comment"),
  ...,
  blame
)
```

Arguments

ato	the ATO object to which to log the entry.
type	One of several event flags: • 'log' saves to the log without displaying. • 'message' (or 'msg') displays a message on screen. • 'warning' displays a warning on screen. • 'debug' flags the entry as debug (not shown by default). • 'comment' user comment entry.
...	The text fragments that compose the event message.
blame	Not used yet; functionality to come in the future.

Value

The updated ATO.

Examples

```
# add an event to the ato
x <- log_event(example_ato, type = "msg", "M: just a test")
get_log(x)
```

`make_ani` *Make an ATO animals object*

Description

Formats the input data into the ATO format and appends the ATO_ani class.

Usage

```
make_ani(
  animal,
  release_datetime,
  release_location = NA_character_,
  release_lat = NA_real_,
  release_lon = NA_real_,
  capture_datetime = as.POSIXct(NA_real_),
  capture_location = NA_character_,
  capture_lat = NA_real_,
  capture_lon = NA_real_,
  tz,
  tbl,
  ...
)
```

Arguments

`animal`	Name of the animal being tracked (character). Required.
`release_datetime`	date and time of the release (POSIXct). Required.
`release_location`	Name of the location where the animal was released character).
`release_lat`	Latitude of the release. Preferably in WGS84 (numeric).
`release_lon`	Longitude of the release. Preferably in WGS84 (numeric).
`capture_datetime`	Date and time of the capture (POSIXct).

capture_location	Name of the location where the animal was captured (character).
capture_lat	Latitude of the capture. Preferably in WGS84 (numeric).
capture_lon	Longitude of the capture. Preferably in WGS84 (numeric).
tz	the timezone of the datetime data.
tbl	The type of table to be generated. One of "data.frame", "data.table", or "tibble". If omitted, the output of ato_table_type_global is used.
...	Non-standard columns to be added to the table.

Value

an ATO_ani object, ready to be used by one of the set functions or init_ato.

See Also

ATO_ani

Examples

```
# only two columns are required to start an ani table.
# Note that posix objects created on-the-fly do not necessarily
# have an associated timezone - try running the code below
# without the tz argument and it will fail.
x <- make_ani(animal = letters[1:10],
              release_datetime = rep(Sys.time(), 10),
              tz = "America/Halifax")
head(x)

# additionally, you can change the type of table being created by using
# the tbl argument - this requires having either data.table or tibble
# installed on your machine.
```

---

make_dep *Make an ATO deployments object*

Description

Formats the input data into the ATO format and appends the ATO_dep class.

Usage

```
make_dep(
  deploy_datetime,
  recover_datetime,
  deploy_location = NA_character_,
  deploy_lat = NA_real_,
```

```
  deploy_lon = NA_real_,
  deploy_z = NA_real_,
  recover_lat = NA_real_,
  recover_lon = NA_real_,
  receiver_manufacturer = NA_character_,
  receiver_model = NA_character_,
  receiver_serial = NA_character_,
  receiver_codeset = NA_character_,
  transmitter = NA_character_,
  transmitter_manufacturer = NA_character_,
  transmitter_model = NA_character_,
  transmitter_serial = NA_character_,
  transmitter_ping_rate = NA_real_,
  tz,
  tbl,
  ...
)
```

Arguments

deploy_datetime	date and time of the deployment (POSIXct). Required.
recover_datetime	date and time of the recovery (POSIXct). Required.
deploy_location	Name of the location where the receiver was deployed (character).
deploy_lat	latitude of the deployment. Preferably in WGS84 (numeric).
deploy_lon	longitude of the deployment. Preferably in WGS84 (numeric).
deploy_z	depth of the deployment, as measured from the reference surface of the water body (numeric).
recover_lat	latitude of the recovery point. Preferably in WGS84 (numeric).
recover_lon	longitude of the recovery point. Preferably in WGS84 (numeric).
receiver_manufacturer	Maker of the receiver (character).
receiver_model	Model of the receiver (character).
receiver_serial	Receiver serial number (character).
receiver_codeset	Codeset of the receiver (character).
transmitter	Transmitter code for a beacon/reference tag (character).
transmitter_manufacturer	Manufacturer of the transmitter (character).
transmitter_model	Model of the transmitter (character).
transmitter_serial	Serial number of the transmitter (integer).

transmitter_ping_rate
: Expected ping rate of the transmitter, in seconds (numeric).

tz
: the timezone of the datetime data.

tbl
: The type of table to be generated. One of "data.frame", "data.table", or "tibble". If omitted, the output of ato_table_type_global is used.

...
: Non-standard columns to be added to the table.

Value

an ATO_dep object, ready to be used by one of the set functions or init_ato.

See Also

ATO_dep

Examples

```
# only two columns are required to start a dep table.
# Note that posix objects created on-the-fly do not necessarily
# have an associated timezone - try running the code below
# without the tz argument and it will fail.
x <- make_dep(deploy_datetime = rep(Sys.time(), 10),
              recover_datetime = rep(Sys.time(), 10),
              tz = "America/Halifax")
head(x)

# additionally, you can change the type of table being created by using
# the tbl argument - this requires having either data.table or tibble
# installed on your machine.

# while you can create deployments with only deployment and recovery times,
# they will likely not be very useful unless you also include a receiver
# serial number or an associated transmitter code.
```

---

make_det *Make an ATO detections object*

Description

Formats the input data into the ATO format and appends the ATO_det class.

Usage

```
make_det(
  datetime,
  receiver_serial,
  transmitter,
```

```
  frac_second = NA_real_,
  sensor_value = NA_real_,
  tz,
  tbl,
  ...
)
```

Arguments

datetime	date and time of the detection (POSIXct). Required.
receiver_serial	Receiver serial (character). Required.
transmitter	Transmitter code (character). Required.
frac_second	fractional second of the detection (numeric).
sensor_value	reported sensor value (numeric).
tz	the timezone of the datetime data.
tbl	The type of table to be generated. One of "data.frame", "data.table", or "tibble". If omitted, the output of ato_table_type_global is used.
...	Non-standard columns to be added to the table.

Value

an ATO_det object, ready to be used by one of the set functions or init_ato.

See Also

ATO_det

Examples

```
# only three columns are required to start a det table.
# Note that posix objects created on-the-fly do not necessarily
# have an associated timezone - try running the code below
# without the tz argument and it will fail.
x <- make_det(datetime = rep(round(Sys.time()), 10),
              receiver_serial = letters[1:10],
              transmitter = paste0(LETTERS[1:10], "-", 1:10),
              tz = "America/Halifax")
head(x)

# additionally, you can change the type of table being created by using
# the tbl argument - this requires having either data.table or tibble
# installed on your machine.

# Note that make_det will complain if the datetime argument contains
# milliseconds. Try removing the round() call above to see it
```

make_obs *Make an ATO observations object*

Description

Formats the input data into the ATO format and appends the ATO_obs class.

Usage

```
make_obs(
  datetime,
  animal = NA_character_,
  transmitter = NA_character_,
  terminal,
  location = NA_character_,
  type = NA_character_,
  lat = NA_real_,
  lon = NA_real_,
  tz,
  tbl,
  ...
)
```

Arguments

datetime	date and time of the observation (POSIXct). Required.
animal	Name of the animal that was observed (character). Each observation must have animal or transmitter information (or both).
transmitter	Transmitter code that was observed (character). Each observation must have animal or transmitter information (or both).
terminal	Was the animal/transmitter permanently captured at the moment of observation (logical)? Required.
location	Name of the place where the observation occurred (character).
type	Type of observation (e.g. directly seen, manual tracking) (character).
lat	latitude of the observation. Preferably in WGS84 (numeric).
lon	longitude of the observation. Preferably in WGS84 (numeric).
tz	the timezone of the datetime data.
tbl	The type of table to be generated. One of "data.frame", "data.table", or "tibble". If omitted, the output of ato_table_type_global is used.
...	Non-standard columns to be added to the table.

Value

an ATO_obs object, ready to be used by one of the set functions or init_ato.

See Also

ATO_obs

Examples

```
# To start an obs table, you need datetime, either the name
# of the animal or the code of the transmitter observed, and to
# specify if that observation is terminal (i.e. there will be no
# more detections/observations for this animal or transmitter).
# Note that posix objects created on-the-fly do not necessarily
# have an associated timezone - try running the code below
# without the tz argument and it will fail.
x <- make_obs(datetime = rep(Sys.time(), 10),
              transmitter = c(paste0(LETTERS[1:5], "-", 1:5), rep(NA, 5)),
              animal = c(rep(NA, 5), letters[6:10]),
              terminal = rep(FALSE, 10),
              tz = "America/Halifax")
head(x)

# additionally, you can change the type of table being created by using
# the tbl argument - this requires having either data.table or tibble
# installed on your machine.
```

---

make_tag *Make an ATO tags object*

Description

Formats the input data into the ATO format and appends the ATO_tag class.

Usage

```
make_tag(
  transmitter,
  manufacturer = NA_character_,
  model = NA_character_,
  serial = NA_character_,
  power_level = NA_real_,
  ping_rate = NA_real_,
  ping_variation = NA_real_,
  activation_datetime = as.POSIXct(NA_real_),
  battery_life = NA_real_,
  sensor_type = NA_character_,
  sensor_unit = NA_character_,
  animal = NA_character_,
  tz,
  tbl,
```

```
  ...
)
```

Arguments

transmitter	Transmitter code (character). Required.
manufacturer	Manufacturer of the transmitter (character).
model	Model of the transmitter (character).
serial	Serial number of the tag (integer).
power_level	Power level of the transmitter (real).
ping_rate, ping_variation	Expected average ping rate of the transmitter and respective variation around the average, in seconds (numeric). E.g. if a tag's ping interval may vary between 60 and 120s, then ping_rate = 90, and ping_variation = 30.
activation_datetime	Date and time of the tag activation (POSIXct).
battery_life	Expected battery duration of the tag, in days (numeric).
sensor_type	Type of sensor data associated with the transmitter (character).
sensor_unit	Unit of the data associated with the transmitter (character).
animal	Name of the animal that received the tag (character).
tz	the timezone of the datetime data.
tbl	The type of table to be generated. One of "data.frame", "data.table", or "tibble". If omitted, the output of ato_table_type_global is used.
...	Non-standard columns to be added to the table.

Value

an ATO_tag object, ready to be used by one of the set functions or init_ato.

See Also

ATO_tag

Examples

```
# All you need to make a tag table is the respective transmitter
# code.
x <- make_tag(transmitter = LETTERS[1:5])
head(x)

# additionally, you can change the type of table being created by using
# the tbl argument - this requires having either data.table or tibble
# installed on your machine.
```

match_update *Match the various slots of the ATO object*

Description

Automatically called by the set functions. You should not need to call this function directly unless you have deactivated the ATO's automatic matching feature (see ato_match_immediate).

Usage

```
match_update(x, silent = FALSE)
```

Arguments

x	an ATO
silent	Supresses summary messages

Details

For developers: You may deactivate ato_match_immediate() to be able to to perform multiple actions in the ATO in quick sequence without incurring the computational cost of repeated, needless matches. You can then run match_update() on the ATO to match everything up in the end. Remember to leave ato_match_immediate() in the same state that the user set it to. See the code inside init_ato for an example.

Value

the updated ATO

Examples

```
old_match_immediate <- ato_match_immediate()
ato_match_immediate(FALSE)
x <- init_ato(ani = ani(example_ato),
              tag = tag(example_ato),
              det = det(example_ato),
              dep = dep(example_ato))
x <- match_update(x)

# clean up
ato_match_immediate(old_match_immediate)
rm(x, old_match_immediate)
```

show,ATO-method *Method to show an ATO*

Description

Method to show an ATO

Usage

```
## S4 method for signature 'ATO'
show(object)
```

Arguments

object	an ATO

Value

Nothing. Called for side-effects

Examples

```
show(example_ato)
```

summary,ATO_ani-method *Summary method for an ATO_ani slot object*

Description

Summary method for an ATO_ani slot object

Usage

```
## S4 method for signature 'ATO_ani'
summary(object)
```

Arguments

object	an ATO_ani slot object

Value

Nothing. Prints a summary.

Examples

```
summary(get_ani(example_ato))
```

---

summary,ATO_dep-method

Summary method for an ATO_dep object

Description

Summary method for an ATO_dep object

Usage

```
## S4 method for signature 'ATO_dep'
summary(object)
```

Arguments

object	an ATO_dep object

Value

Nothing. Prints a summary.

Examples

```
summary(get_dep(example_ato))
```

---

summary,ATO_det-method

Summary method for an ATO_det slot object

Description

Summary method for an ATO_det slot object

Usage

```
## S4 method for signature 'ATO_det'
summary(object)
```

Arguments

object	an ATO_det slot object

Value

Nothing. Prints a summary.

Examples

```
summary(get_det(example_ato))
```

---

summary,ATO_log-method

Summary method for an ATO_log slot object

Description

Summary method for an ATO_log slot object

Usage

```
## S4 method for signature 'ATO_log'
summary(object)
```

Arguments

object	an ATO_log slot object

Value

Nothing. Prints a summary.

Examples

```
summary(get_log(example_ato))
```

---

summary,ATO_obs-method

Summary method for an ATO_obs slot object

Description

Summary method for an ATO_obs slot object

Usage

```
## S4 method for signature 'ATO_obs'
summary(object)
```

Arguments

object	an ATO_obs slot object

Value

Nothing. Prints a summary.

Examples

```
summary(get_obs(example_ato))
```

---

summary,ATO_tag-method

Summary method for an ATO_tag slot object

Description

Summary method for an ATO_tag slot object

Usage

```
## S4 method for signature 'ATO_tag'
summary(object)
```

Arguments

object	an ATO_tag slot object

Value

Nothing. Prints a summary.

Examples

```
summary(get_tag(example_ato))
```

table_type *Table type of ATO data*

Description

The tables within an ATO may be of one of three main groups: 'data.frame', 'data.table', or 'tibble'. By default, the make functions (e.g. `make_ani`) create data.frame tables, which in turn create a data.frame ATO. This can be modified either by using the tbl argument within the make functions, or globally by calling `ato_table_type_global`.

Usage

```
table_type(x, expect)

## S4 method for signature 'ATO'
table_type(x, expect)

## S4 method for signature 'ATO_ani'
table_type(x, expect)

## S4 method for signature 'ATO_dep'
table_type(x, expect)

## S4 method for signature 'ATO_det'
table_type(x, expect)

## S4 method for signature 'ATO_obs'
table_type(x, expect)

## S4 method for signature 'ATO_tag'
table_type(x, expect)
```

Arguments

x	an `ATO`
expect	An expected type of table. Optional. Errors if x is not of this type.

Details

table_type may be used on an ATO object to check the type of table within its slots, or directly on a slot object. table_type may also be used to change the type of tables of a slot object or of an ATO object.

Value

if expect is missing, returns the type of table used by the ATO. If expect is provided, either errors or returns nothing.

Examples

```
# check the table type of an ato
table_type(example_ato)

# check the table type of an ATO slot
table_type(.ATO_ani)
```

---

table_type<- *Assign a new table type to an ATO or an ATO slot.*

Description

Assign a new table type to an ATO or an ATO slot.

Usage

```
table_type(x) <- value

## S4 replacement method for signature 'ATO'
table_type(x) <- value

## S4 replacement method for signature 'ATO_ani'
table_type(x) <- value

## S4 replacement method for signature 'ATO_dep'
table_type(x) <- value

## S4 replacement method for signature 'ATO_det'
table_type(x) <- value

## S4 replacement method for signature 'ATO_obs'
table_type(x) <- value

## S4 replacement method for signature 'ATO_tag'
table_type(x) <- value
```

Arguments

x	an ATO
value	The new table type.

Value

Acts directly on x

Examples

```
# runnning examples for this function would
# require having either data.table or tibble installed.
# if you have them installed, you can try e.g.
# table_type(example_ato) <- "data.table"
# table_type(example_ato)
```

---

tzone *What is the timezone of this ATO object?*

Description

You may use tzone to check and display the timezone of the datetime data of an ATO object or of any ATO slot object.

Usage

```
tzone(x)

## S4 method for signature 'ATO'
tzone(x)

## S4 method for signature 'ATO_ani'
tzone(x)

## S4 method for signature 'ATO_dep'
tzone(x)

## S4 method for signature 'ATO_det'
tzone(x)

## S4 method for signature 'ATO_obs'
tzone(x)

## S4 method for signature 'ATO_tag'
tzone(x)
```

Arguments

x	an ATO or ATO slot object.

Value

Returns the timezone of the ATO object.

Examples

```
# check the timezone of the example ATO
tzone(example_ato)
```

---

tzone<- *Modify the time zone of an ATO object*

Description

Modify the time zone of an ATO object

Usage

```
tzone(x) <- value

## S4 replacement method for signature 'ATO'
tzone(x) <- value

## S4 replacement method for signature 'ATO_ani'
tzone(x) <- value

## S4 replacement method for signature 'ATO_det'
tzone(x) <- value

## S4 replacement method for signature 'ATO_dep'
tzone(x) <- value

## S4 replacement method for signature 'ATO_obs'
tzone(x) <- value

## S4 replacement method for signature 'ATO_tag'
tzone(x) <- value
```

Arguments

x	an ATO or ATO slot object.
value	the desired time zone

Value

Nothing, acts directly on x

Examples

```
# check the timezone of the example ATO
tzone(example_ato)
summary(get_det(example_ato))

# change the tz to UTC
tzone(example_ato) <- "UTC"
tzone(example_ato)
summary(get_det(example_ato))

# clean_up
tzone(example_ato) <- "Europe/Copenhagen"
```

Index

∗ **classes**
ATO_ani, 4
ATO_dep, 6
ATO_det, 7
ATO_log, 7
ATO_obs, 9
ATO_tag, 10
∗ **datasets**
ATO_ani, 4
ATO_dep, 6
ATO_det, 7
ATO_log, 7
ATO_obs, 9
ATO_tag, 10
.ATO_ani *(*ATO_ani*)*, 4
.ATO_dep *(*ATO_dep*)*, 6
.ATO_det *(*ATO_det*)*, 7
.ATO_log *(*ATO_log*)*, 7
.ATO_obs *(*ATO_obs*)*, 9
.ATO_tag *(*ATO_tag*)*, 10
$,ATO-method, 3

ani *(*get_ani*)*, 11
ani,ATO-method *(*get_ani*)*, 11
ATO, *3*, *4*, *6–21*, *31*, *32*, *36–39*
ATO-class, 3
ATO_ani, 4, *4*, *24*
ato_debug, 5
ato_debug_header, 5
ATO_dep, *4*, 6, *26*
ATO_det, *4*, 7, *27*
ATO_log, *4*, 7
ato_match_immediate, 8, *31*
ATO_obs, *4*, 9, *29*
ato_table_type_global, 10, *24*, *26–28*, *30*, *36*
ATO_tag, *4*, 10, *30*

dep *(*get_dep*)*, 12
dep,ATO-method *(*get_dep*)*, 12
det *(*get_det*)*, 13
det,ATO-method *(*get_dep*)*, 12

Extract, *3*

get_ani, 11
get_ani,ATO-method *(*get_ani*)*, 11
get_dep, 12
get_dep,ATO-method *(*get_dep*)*, 12
get_det, 13, *14*, *15*
get_det,ATO-method *(*get_det*)*, 13
get_det_dep, 14
get_det_dep,ATO-method *(*get_det_dep*)*, 14
get_det_tag, 15
get_det_tag,ATO-method *(*get_det_tag*)*, 15
get_log, 16
get_log,ATO-method *(*get_log*)*, 16
get_obs, 17
get_obs,ATO-method *(*get_obs*)*, 17
get_tag, 18
get_tag,ATO-method *(*get_tag*)*, 18

has, 19
has,ATO-method *(*has*)*, 19

init_ato, 20, *24*, *26–28*, *30*, *31*
is_ato, 21

log_comment, 21, *22*
log_event, 22

make_ani, *4*, *20*, 23, *36*
make_dep, *6*, *20*, 24
make_det, *7*, *20*, 26
make_obs, *9*, *20*, 28
make_tag, *10*, *11*, *20*, 29
match_update, 31

obs *(*get_obs*)*, 17
obs,ATO-method *(*get_obs*)*, 17

set, *24*, *26–28*, *30*, *31*
show,ATO-method, 32
summary,ATO_ani-method, 32
summary,ATO_dep-method, 33
summary,ATO_det-method, 33
summary,ATO_log-method, 34
summary,ATO_obs-method, 34
summary,ATO_tag-method, 35

table_type, *4*, *6–9*, *11*, *20*, 36
table_type,ATO-method *(*table_type*)*, 36
table_type,ATO_ani-method *(*table_type*)*, 36
table_type,ATO_dep-method *(*table_type*)*, 36
table_type,ATO_det-method *(*table_type*)*, 36
table_type,ATO_obs-method *(*table_type*)*, 36
table_type,ATO_tag-method *(*table_type*)*, 36
table_type<-, 37
table_type<-,ATO-method *(*table_type<-*)*, 37
table_type<-,ATO_ani-method *(*table_type<-*)*, 37
table_type<-,ATO_dep-method *(*table_type<-*)*, 37
table_type<-,ATO_det-method *(*table_type<-*)*, 37
table_type<-,ATO_obs-method *(*table_type<-*)*, 37
table_type<-,ATO_tag-method *(*table_type<-*)*, 37
tag *(*get_tag*)*, 18
tag,ATO-method *(*get_tag*)*, 18
tzone, 38
tzone,ATO-method *(*tzone*)*, 38
tzone,ATO_ani-method *(*tzone*)*, 38
tzone,ATO_dep-method *(*tzone*)*, 38
tzone,ATO_det-method *(*tzone*)*, 38
tzone,ATO_obs-method *(*tzone*)*, 38
tzone,ATO_tag-method *(*tzone*)*, 38
tzone<-, 39
tzone<-,ATO-method *(*tzone<-*)*, 39
tzone<-,ATO_ani-method *(*tzone<-*)*, 39
tzone<-,ATO_dep-method *(*tzone<-*)*, 39
tzone<-,ATO_det-method *(*tzone<-*)*, 39
tzone<-,ATO_obs-method *(*tzone<-*)*, 39
tzone<-,ATO_tag-method *(*tzone<-*)*, 39