add_duration

Description

add_duration() approximates the time spent on a visit based on the difference between two consecutive timestamps, replacing differences exceeding cutoff with the value defined in replace_by.
add_duration

Usage

add_duration(
  wt,
  cutoff = 300,
  replace_by = NA,
  last_replace_by = NA,
  device_switch_na = FALSE,
  device_var = NULL
)

Arguments

wt webtrack data object.
cutoff numeric (seconds). If duration is greater than this value, it is reset to the value defined by replace_by. Defaults to 300 seconds.
replace_by numeric. Determines whether differences greater than the cutoff are set to NA, or some value. Defaults to NA.
last_replace_by numeric. Determines whether the last visit for an individual is set to NA, or some value. Defaults to NA.
device_switch_na boolean. Relevant only when data was collected from multiple devices. When visits are ordered by timestamp sequence, two consecutive visits can come from different devices, which makes the timestamp difference less likely to be the true duration. It may be preferable to set the duration of the visit to NA (TRUE) rather than the difference to the next timestamp (FALSE). Defaults to FALSE.
device_var character. Column indicating device. Required if 'device_switch_na' set to TRUE. Defaults to NULL.

Value

webtrack data.table (ordered by panelist_id and timestamp) with the same columns as wt and a new column called duration.

Examples

```r
## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
wtdiff <- add_duration(wt)

# Defining cutoff at 10 minutes, replacing those exceeding cutoff to 5 minutes, 
# and setting duration before device switch to 'NA':
wtdiff <- add_duration(wt, 
  cutoff = 600, replace_by = 300, 
  device_switch_na = TRUE, device_var = "device"
)
```

## End(Not run)
add_next_visit

Add the next visit as a new column

Description

add_next_visit() adds the subsequent visit, as determined by order of timestamps as a new column. The next visit can be added as either the full URL, the extracted host or the extracted domain, depending on level.

Usage

add_next_visit(wt, level = "url")

Arguments

wt webtrack data object.
level character. Either "url", "host" or "domain". Defaults to "url".

Value

webtrack data.table with the same columns as wt and a new column called url_next, host_next or domain_next.

Examples

## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# Adding next full URL as new column
wt <- add_next_visit(wt, level = "url")
# Adding next host as new column
wt <- add_next_visit(wt, level = "host")
# Adding next domain as new column
wt <- add_next_visit(wt, level = "domain")

## End(Not run)

add_panelist_data

Add panelist features to tracking data

Description

add_panelist_data() adds information about panelists (e.g., from a survey) to the tracking data.

Usage

add_panelist_data(wt, data, cols = NULL, join_on = "panelist_id")
**add_previous_visit**

Arguments

- **wt**: webtrack data object.
- **data**: a data.table (or object that can be converted to data.table) which contains columns about panelists.
- **cols**: character vector of columns to add. If NULL, all columns are added. Defaults to NULL.
- **join_on**: which columns to join on. Defaults to "panelist_id".

Value

webtrack object with the same columns and the columns from data specified in cols.

Examples

```r
## Not run:
data("testdt_tracking")
data("testdt_survey_w")
w t <- as.wt_dt(testdt_tracking)
# add survey test data
add_panelist_data(wt, testdt_survey_w)

## End(Not run)
```

**add_previous_visit**

Add the previous visit as a new column

Description

add_previous_visit() adds the previous visit, as determined by order of timestamps as a new column. The previous visit can be added as either the full URL, the extracted host or the extracted domain, depending on level.

Usage

```r
add_previous_visit(wt, level = "url")
```

Arguments

- **wt**: webtrack data object.
- **level**: character. Either "url", "host" or "domain". Defaults to "url".

Value

webtrack data.table with the same columns as wt and a new column called `url_previous`, `host_previous` or `domain_previous`.
Examples

```r
## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# Adding previous full URL as new column
wt <- add_previous_visit(wt, level = "url")
# Adding previous host as new column
wt <- add_previous_visit(wt, level = "host")
# Adding previous domain as new column
wt <- add_previous_visit(wt, level = "domain")

## End(Not run)
```

---

**add_referral**  
*Add social media referrals as a new column*

**Description**

Identifies whether a visit was referred to from social media and adds it as a new column. See details for method.

**Usage**

```
add_referral(wt, platform_domains, patterns)
```

**Arguments**

- `wt`: webtrack data object.
- `platform_domains`: character. A vector of platform domains for which referrers should be identified. Order and length must correspond to `patterns` argument.
- `patterns`: character. A vector of patterns for which referrers should be identified. Order and length must correspond to `platform_domains` vector.

**Details**

To identify referrals, we rely on the method described as most valid in Schmidt et al.: When the domain preceding a visit was to the platform in question, and the query string of the visit’s URL contains a certain pattern, we count it as a referred visit. For Facebook, the pattern has been identified by Schmidt et al. as `fbclid=`, although this can change in future.

**Value**

webtrack data.table with the same columns as `wt` and a new column called `referral`, which takes on NA if no referral has been identified, or the name specified `platform_domains` if a referral from that platform has been identified.
add_session

References

Examples
```r
## Not run:
data("testdt_tracking")
wtk <- as.wt_dt(testdt_tracking)
wtk <- add_referral(wtk, platform_domains = "facebook.com", patterns = "fbclid=")
plot(wtk)
wtk <- add_referral(wtk,
  platform_domains = c("facebook.com", "twitter.com"),
  patterns = c("fbclid="), "utm_source=twitter")
## End(Not run)
```

add_session

Add a session variable

Description
add_session() groups visits into "sessions", defining a session to end when the difference between two consecutive timestamps exceeds a cutoff.

Usage
```r
add_session(wt, cutoff)
```

Arguments
- `wt`: webtrack data object.
- `cutoff`: numeric (seconds). If the difference between two consecutive timestamps exceeds this value, a new browsing session is defined.

Value
webtrack data.table (ordered by panelist_id and timestamp) with the same columns as wt and a new column called session.

Examples
```r
## Not run:
data("testdt_tracking")
wtk <- as.wt_dt(testdt_tracking)
# Setting cutoff to 30 minutes
wtk <- add_session(wtk, cutoff = 1800)
## End(Not run)
```
add_title  

Download and add the "title" of a URL

**Description**

`add_title()` gets the title of a URL by accessing the web address online and adds the title as a new column. See details for the meaning of "title". You need an internet connection to run this function.

**Usage**

```r
add_title(wt, lang = "en-US,en-GB,en")
```

**Arguments**

- `wt`  
  webtrack data object.

- `lang`  
  character (a language tag). Language accepted by the request. Defaults to "en-US,en-GB,en". Note that you are likely to still obtain titles different from the ones seen originally by the user, because the language also depend on the user's IP and device settings.

**Details**

The title of a website (the text within the `<title>` tag of a web site's `<head>`) is the text that is shown on the "tab" when looking at the website in a browser. It can contain useful information about a URL's content and can be used, for example, for classification purposes. Note that it may take a while to run this function for a large number of URLs.

**Value**

webtrack data.table with the same columns as `wt` and a new column called "title", which will be NA if the title cannot be retrieved.

**Examples**

```r
## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)[1:2]
# Get titles with 'lang' set to default English
wt_titles <- add_title(wt)
# Get titles with 'lang' set to German
wt_titles <- add_title(wt, lang = "de")
## End(Not run)
```
audience_incidence

Create incidence matrix for two-mode networks including audiences

Description

audience_incidence() created an incidence matrix, which is a matrix $A$ with entries $A[i,j]=1$ if panelist $i$ visited web site $j$ at least once. Web site can be defined, for example, by the URL's domain, or its host.

Usage

audience_incidence(wt, mode2 = "domain", cutoff = 3)

Arguments

wt webtrack data object.
mode2 character. Name of column that includes the second mode (e.g., domain or host)
cutoff visits below this cutoff will not be considered as a visit.

Value

Incidence matrix of a two-mode network

See Also

To create audience networks see audience_network.

Examples

```r
## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
wt <- add_duration(wt)
wt <- suppressWarnings(extract_domain(wt))
# create incidence matrix using domains as second mode
incidence <- audience_incidence(wt)
# create incidence matrix using hosts as second mode
wt <- suppressWarnings(extract_host(wt))
incidence <- audience_incidence(wt, mode2 = "host")
## End(Not run)
```
audience_network Create audience networks

Description

audience network

Usage

audience_network(wt, mode2 = "domain", cutoff = 3, type = "pmi", alpha = 0.05)

Arguments

wt webtrack data object
mode2 character. name of column that includes the second mode (e.g. 'domain' or 'host')
cutoff visits below this cutoff will not be considered as a visit
type one of "pmi", "phi", "disparity", "sdsm", "or "fdsm".
alpha significance level

Value

audience network as igraph object

Examples

## Not run:
data("testdt_tracking")
w <- as.wt_dt(testdt_tracking)
w <- add_duration(wt)
w <- suppressWarnings(extract_domain(wt))
network <- audience_network(wt, type = "pmi", cutoff = 120)
## End(Not run)

bakshy Bakshy Top500 Ideological alignment of 500 domains based on facebook data

Description

Bakshy Top500 Ideological alignment of 500 domains based on facebook data

Usage

bakshy
classify_visits

Format

An object of class data.table (inherits from data.frame) with 500 rows and 7 columns.

References


classify_visits (classify visits by matching to a list of classes)

Description

classify_visits() categorizes visits by either extracting the visit URL’s domain or host and matching them to a list of domains or hosts; or by matching a list of regular expressions against the visit URL.

Usage

classify_visits(
  wt, 
  classes, 
  match_by = "domain", 
  regex_on = NULL, 
  return_rows_by = NULL, 
  return_rows_val = NULL
)

Arguments

wt 
webtrack data object.

classes 
a data.table containing classes that can be matched to visits.

match_by character. Whether to match list entries from classes to the domain of a visit ("domain") or the host ("host") with an exact match; or with a regular expression against the whole URL of a visit ("regex"). If set to "domain" or "host", both wt and classes need to have a column called accordingly. If set to "regex", the url column of wt will be used, and you need to set regex_on to the column in classes for which to do the pattern matching. Defaults to "domain".

regex_on 
character. Column in classes which to use for pattern matching. Defaults to NULL.

return_rows_by 
character. A column in classes on which to subset the returning data. Defaults to NULL.
return_rows_val

character. The value of the columns specified in return_rows_by, for which data should be returned. For example, if your classes data contains a column type, which has a value called "shopping", setting return_rows_by to "type" and return_rows_val to "shopping" will only return visits classified as "shopping".

Value

webtrack data.table with the same columns as wt and any column in classes except the column specified by match_by.

Examples

```r
## Not run:
data("testdt_tracking")
data("domain_list")
wt <- as.wt_dt(testdt_tracking)
# classify visits via domain
wt_domains <- extract_domain(wt, drop_na = FALSE)
wt_classes <- classify_visits(wt_domains, classes = domain_list, match_by = "domain")
# classify visits via domain
# for the example, just renaming "domain" column
domain_list$host <- domain_list$domain
wt_hosts <- extract_host(wt, drop_na = FALSE)
wt_classes <- classify_visits(wt_hosts, classes = domain_list, match_by = "host")
# classify visits with pattern matching
# for the example, any value in "domain" treated as pattern
data("domain_list")
regex_list <- domain_list[type == "facebook"]
wt_classes <- classify_visits(wt[1:5000],
  classes = regex_list,
  match_by = "regex", regex_on = "domain")
# classify visits via domain and only return class "search"
data("domain_list")
wt_classes <- classify_visits(wt_domains,
  classes = domain_list,
  match_by = "domain", return_rows_by = "type",
  return_rows_val = "search")
```

## End(Not run)
Usage

create_urldummy(wt, dummy, name)

Arguments

wt webtrack data object
dummy a vector of urls that should be dummy coded
name name of dummy variable to create.

Value

webtrack object with the same columns and a new column called "name" including the dummy variable

Examples

## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
wt <- extract_domain(wt)
code_urls <- "https://dkr1.ssisurveys.com/tzktsxmta"
create_urldummy(wt, dummy = code_urls, name = "test_dummy")
## End(Not run)
Arguments

wt webtrack data object.

method character. One of "aggregate", "flag" or "drop". If set to "aggregate", consecutive visits (no matter the time difference) to the same URL are combined and their duration aggregated. In this case, a duration column must be specified via "duration_var". If set to "flag", duplicates within a certain time frame are flagged in a new column called duplicate. In this case, within argument must be specified. If set to "drop", duplicates are dropped. Again, within argument must be specified. Defaults to "aggregate".

within numeric (seconds). If method set to "flag" or "drop", a subsequent visit is only defined as a duplicate when happening within this time difference. Defaults to 1 second.

duration_var character. Name of duration variable. Defaults to "duration".

keep_nvisits boolean. If method set to "aggregate", this determines whether number of aggregated visits should be kept as variable. Defaults to FALSE.

same_day boolean. If method set to "aggregate", determines whether to count visits as consecutive only when on the same day. Defaults to TRUE.

Value

webtrack data.table with the same columns as wt with updated duration

Examples

## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
wt <- add_duration(wt, cutoff = 300, replace_by = 300)
# Dropping duplicates with one-second default
wt_dedup <- deduplicate(wt, method = "drop")
# Flagging duplicates with one-second default
wt_dedup <- deduplicate(wt, method = "flag")
# Aggregating duplicates
wt_dedup <- deduplicate(wt[1:1000], method = "aggregate")
# Aggregating duplicates and keeping number of visits for aggregated visits
wt_dedup <- deduplicate(wt[1:1000], method = "aggregate", keep_nvisits = TRUE)

## End(Not run)

domain_list Domain list classification of domains into news, portals, search, and social media

Description

Domain list classification of domains into news, portals, search, and social media
drop_query

Usage

domain_list

Format

An object of class data.table (inherits from data.frame) with 663 rows and 2 columns.

References


drop_query

Drop the query and fragment from URL

Description

drop_query() adds the URL without query and fragment as a new column. The query is defined as the part following a "?" after the path. The fragment is anything following a "#" after the query.

Usage

drop_query(wt, varname = "url")

Arguments

wt webtrack data object.
varname character. name of the column from which to extract the host. Defaults to "url".

Value

webtrack data.table with the same columns as wt and a new column called '<varname>_noquery'

Examples

## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# Extract URL without query/fragment
wt <- drop_query(wt)

## End(Not run)
extract_domain

Extract the domain from URL

Description

extract_domain() adds the domain of a URL as a new column. By "domain", we mean the "top private domain", i.e., the domain under the public suffix (e.g., "com") as defined by the Public Suffix List. See details.

Extracts the domain from urls.

Usage

extract_domain(wt, varname = "url", drop_na = TRUE)

Arguments

wt webtrack data object.

varname character. Name of the column from which to extract the host. Defaults to "url".

drop_na boolean. Determines whether rows for which no host can be extracted should be dropped from the data. Defaults to TRUE.

Details

We define a "web domain" in the common colloquial meaning, that is, the part of an web address that identifies the person or organization in control. is google.com. More technically, what we mean by "domain" is the "top private domain", i.e., the domain under the public suffix, as defined by the Public Suffix List. Note that this definition sometimes leads to counterintuitive results because not all public suffixes are "registry suffixes". That is, they are not controlled by a domain name registrar, but allow users to directly register a domain. One example of such a public, non-registry suffix is blogspot.com. For a URL like www.mysite.blogspot.com, our function, and indeed the packages we are aware of, would extract the domain as mysite.blogspot.com, although you might think of blogspot.com as the domain. For details, see here

Value

webtrack data table with the same columns as wt and a new column called 'domain' (or, if varname not equal to 'url', '<varname>_domain')

Examples

## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# Extract domain and drop rows without domain
wt <- extract_domain(wt)
# Extract domain and keep rows without domain
extract_host

wt <- extract_domain(wt, drop_na = FALSE)
## End(Not run)

---

**extract_host**

*Extract the host from URL*

**Description**

extract_host() adds the host of a URL as a new column. The host is defined as the part following the scheme (e.g., "https://") and preceding the subdirectory (anything following the next "/"). Note that for URL entries like chrome-extension://soomething or http://192.168.0.1/something, result will be set to NA.

**Usage**

extract_host(wt, varname = "url", drop_na = TRUE)

**Arguments**

- `wt`  
  webtrack data object.
- `varname`  
  character. Name of the column from which to extract the host. Defaults to "url".
- `drop_na`  
  boolean. Determines whether rows for which no host can be extracted should be dropped from the data. Defaults to TRUE.

**Value**

webtrack data.table with the same columns as wt and a new column called 'host' (or, if varname not equal to 'url', '<varname>_host')

**Examples**

```r
## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# Extract host and drop rows without host
wt <- extract_host(wt)
# Extract host and keep rows without host
wt <- extract_host(wt, drop_na = FALSE)
## End(Not run)
```
**extract_path**  
*Extract the path from URL*

**Description**

`extract_path()` adds the path of a URL as a new column. The path is defined as the part following the host but not including a query (anything after a "?") or a fragment (anything after a "#").

**Usage**

`extract_path(wt, varname = "url")`

**Arguments**

- **wt**: webtrack data object
- **varname**: character. name of the column from which to extract the host. Defaults to "url".

**Value**

webtrack data.table with the same columns as `wt` and a new column called 'path' (or, if `varname` not equal to 'url', '<varname>_path')

**Examples**

```r
## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# Extract path
wt <- extract_path(wt)
## End(Not run)
```

---

**isolation_index**  
*Isolation Index*

**Description**

Given two groups (A and B) of individuals, the isolation index captures the extent to which group A disproportionately visit websites whose other visitors are also members of group A.

**Usage**

`isolation_index(grp_A, grp_B)`
Arguments

grp_A  vector (usually corresponds to a column in a webtrack data.table) indicating the number of individuals of group A using a website

grp_B  vector (usually corresponds to a column in a webtrack data.table) indicating the number of individuals of group B using a website

Details

A value of 1 indicates that the websites visited by group A and group B do not overlap. A value of 0 means both visit exactly the same websites.

Value

numeric value between 0 and 1. 0 indicates no isolation and 1 perfect isolation

References


Examples

# perfect isolation
left <- c(5, 5, 0, 0)
right <- c(0, 0, 5, 5)
isolation_index(left, right)

# perfect overlap
left <- c(5, 5, 5, 5)
right <- c(5, 5, 5, 5)
isolation_index(left, right)

news_types

<table>
<thead>
<tr>
<th>news_types</th>
<th>News Types</th>
</tr>
</thead>
</table>

Description

Classification of domains into different news types

Usage

news_types

Format

An object of class data.table (inherits from data.frame) with 690 rows and 2 columns.
References

print.wt_dt  
Print web tracking data

Description
Print web tracking data

Usage
## S3 method for class 'wt_dt'
print(x, ...)

Arguments
x object of class wt_dt
... additional parameters for print

Value
No return value, called for side effects

summary.wt_dt  
Summary function for web tracking data

Description
Summary function for web tracking data

Usage
## S3 method for class 'wt_dt'
summary(object, ...)

Arguments
object object of class wt_dt
... additional parameters for summary

Value
No return value, called for side effects
sum_activity

Summarize activity per person

Description

sum_activity() counts the number of active time periods (i.e., days, weeks, months, years, or waves) by panelist_id. A period counts as "active" if the panelist provided at least one visit for that period.

Usage

sum_activity(wt, timeframe = "date")

Arguments

wt webtrack data object.

timeframe character. Indicates for what time frame to aggregate visits. Possible values are "date", "week", "month", "year" or "wave". If set to "wave", wt must contain a column call wave. Defaults to "date".

Value

a data.table with columns panelist_id, column indicating the number of active time units.

Examples

## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# summarize activity by day
wt_sum <- sum_activity(wt, timeframe = "date")
## End(Not run)

sum_durations

Summarize visit duration by person

Description

sum_durations() summarizes the duration of visits by person within a timeframe, and optionally by visit_class of visit. Note:

- If for a time frame all rows are NA on the duration column, the summarized duration for that time frame will be NA.
- If only some of the rows of a time frame are NA on the duration column, the function will ignore those NA rows.
If there were no visits to a class (i.e., a value of the 'visit_class' column) for a time frame, the summarized duration for that time frame will be zero; if there were visits, but NA on duration, the summarized duration will be NA.

Usage

```r
sum_durations(wt, var_duration = NULL, timeframe = NULL, visit_class = NULL)
```

Arguments

- `wt`: webtrack data object.
- `var_duration`: character. Name of the duration variable if already present. Defaults to NULL, in which case duration will be approximated with `add_duration(wt, cutoff = 300, replace_by = "na", replace_val = NULL)`.
- `timeframe`: character. Indicates for what time frame to aggregate visit durations. Possible values are "date", "week", "month", "year", "wave" or NULL. If set to "wave", `wt` must contain a column called `wave`. Defaults to NULL, in which case the output contains duration of visits for the entire time.
- `visit_class`: character. Column that contains a classification of visits. For each value in this column, the output will have a column indicating the number of visits belonging to that value. Defaults to NULL.

Value

a data.table with columns `panelist_id`, column indicating the time unit (unless `timeframe` set to NULL), `duration_visits` indicating the duration of visits (in seconds, or whatever the unit of the variable specified by `var_duration` parameter), and a column for each value of `visit_class`, if specified.

Examples

```r
## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# summarize for whole period
wt_summ <- sum_durations(wt)
# summarize by week
wt_summ <- sum_durations(wt, timeframe = "week")
# create a class variable to summarize by class
wt <- suppressWarnings(extract_domain(wt, drop_na = TRUE))
wt[, google := ifelse(domain == "google.com", 1, 0)]
wt_summ <- sum_durations(wt, timeframe = "week", visit_class = "google")
## End(Not run)
```
### Description

`sum_visits()` summarizes the number of visits by person within a timeframe, and optionally by `visit_class` of visit.

### Usage

```r
sum_visits(wt, timeframe = NULL, visit_class = NULL)
```

### Arguments

- **wt**: webtrack data object.
- **timeframe**: character. Indicates for what time frame to aggregate visits. Possible values are "date", "week", "month", "year", "wave" or `NULL`. If set to "wave", `wt` must contain a column call wave. Defaults to `NULL`, in which case the output contains number of visits for the entire time.
- **visit_class**: character. Column that contains a classification of visits. For each value in this column, the output will have a column indicating the number of visits belonging to that value. Defaults to `NULL`.

### Value

A data.table with columns `panelist_id`, column indicating the time unit (unless `timeframe` set to `NULL`), `n_visits` indicating the number of visits, and a column for each value of `visit_class`, if specified.

### Examples

```r
## Not run:
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
# summarize for whole period
wt_summ <- sum_visits(wt)
# summarize by week
wt_summ <- sum_visits(wt, timeframe = "week")
# create a class variable to summarize by class
wt <- suppressWarnings(extract_domain(wt, drop_na = TRUE))
wt[, google := ifelse(domain == "google.com", 1, 0)]
wt_summ <- sum_visits(wt, timeframe = "week", visit_class = "google")
## End(Not run)
```
<table>
<thead>
<tr>
<th>Data Set</th>
<th>Description</th>
<th>Usage</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>testdt_survey_l</strong></td>
<td>Test survey Same randomly generated survey data, one row per person/wave (long format)</td>
<td>testdt_survey_l</td>
<td>An object of class <code>tbl_df</code> (inherits from <code>tbl.data.frame</code>) with 15 rows and 7 columns.</td>
</tr>
<tr>
<td><strong>testdt_survey_w</strong></td>
<td>Test survey Randomly generated survey data only used for illustrative purposes (wide format)</td>
<td>testdt_survey_w</td>
<td>An object of class <code>data.frame</code> with 5 rows and 8 columns.</td>
</tr>
<tr>
<td><strong>testdt_tracking</strong></td>
<td>Test data Sample of fully anonymized webtrack data from a research project with US participants</td>
<td>testdt_tracking</td>
<td>An object of class <code>data.table</code> (inherits from <code>data.frame</code>) with 49612 rows and 5 columns.</td>
</tr>
</tbody>
</table>
vars_exist

**Check if columns are present**

**Description**

`vars_exist()` checks if columns are present in a webtrack data object. By default, checks whether the data has a `panelist_id`, `url` and a `timestamp` column.

**Usage**

```r
vars_exist(wt, vars = c("panelist_id", "url", "timestamp"))
```

**Arguments**

- `wt`: webtrack data object.

**Value**

A data.table object.

---

**wt_dt**

*An S3 class, based on data.table, to store web tracking data*

**Description**

An S3 class, based on `data.table`, to store web tracking data

Convert a data.frame containing web tracking data to a `wt_dt` object

**Usage**

```r
as.wt_dt(
  x,
  timestamp_format = "%Y-%m-%d %H:%M:%OS",
  varnames = c(panelist_id = "panelist_id", url = "url", timestamp = "timestamp")
)
```

**Arguments**

- `x`: data.frame containing a necessary set of columns, namely panelist’s ID, visit URL and visit timestamp.
- `timestamp_format`: string. Specifies the raw timestamp’s formatting. Defaults to "%Y-%m-%d %H:%M:%OS".
- `varnames`: Named vector of column names, which contain the panelist’s ID (`panelist_id`), the visit’s URL (`url`) and the visit’s timestamp (`timestamp`).
Details
A wt.dt table is a data.table. Therefore, it can be used by any function that would work on a data.frame or a data.table. Most of the operation such as variable creation, subsetting and joins are inherited from the data.table [] operator, following the convention DT[i,j,by] (see data table package for detail).

Value
a webtrack data object with at least columns panelist_id, url and timestamp

logical. TRUE if x is a webtrack data object and FALSE otherwise

See Also
• data.table – on which wt.dt is based

Examples
data("testdt_tracking")
wt <- as.wt_dt(testdt_tracking)
is.wt_dt(wt)
Index

* datasets
  bakshy, 10
  domain_list, 14
  news_types, 19
  testdt_survey_l, 24
  testdt_survey_w, 24
  testdt_tracking, 24
  add_duration, 2
  add_next_visit, 4
  add_panelist_data, 4
  add_previous_visit, 5
  add_referral, 6
  add_session, 7
  add_title, 8
  as.wt_dt(wt_dt), 25
  audience_incidence, 9
  audience_network, 9, 10
  bakshy, 10
  classify_visits, 11
  create_urldummy, 12
  data.frame, 26
  data.table, 25, 26
  deduplicate, 13
  domain_list, 14
  drop_query, 15
  extract_domain, 16
  extract_host, 17
  extract_path, 18
  is.wt_dt(wt_dt), 25
  isolation_index, 18
  news_types, 19
  print.wt_dt, 20
  sum_activity, 21
  sum_durations, 21
  sum_visits, 23
  summary.wt_dt, 20
  testdt_survey_l, 24
  testdt_survey_w, 24
  testdt_tracking, 24
  vars_exist, 25
  wt_dt, 25

27