

# Package ‘daqapo’

July 22, 2025

**Type** Package

**Title** Data Quality Assessment for Process-Oriented Data

**Version** 0.3.2

**Date** 2022-07-13

**Description** Provides a variety of methods to identify data quality issues in process-oriented data, which are useful to verify data quality in a process mining context. Builds on the class for activity logs implemented in the package 'bupaR'. Methods to identify data quality issues either consider each activity log entry independently (e.g. missing values, activity duration outliers,...), or focus on the relation amongst several activity log entries (e.g. batch registrations, violations of the expected activity order,...).

**License** MIT + file LICENSE

**URL** <https://github.com/bupaverse/daqapo/>

**BugReports** <https://github.com/bupaverse/daqapo/issues/>

**LazyData** true

**RoxygenNote** 7.2.0

**Encoding** UTF-8

**Depends** R (>= 3.5.0)

**Imports** dplyr, lubridate, stringdist, stringr, tidyr, xesreadR, rlang,  
bupaR (>= 0.5.0), readr, edeaR, magrittr, purrr, glue, miniUI,  
shiny, tibble

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Niels Martin [aut, cre],  
Greg Van Houdt [ctb],  
Gert Janssenswillen [ctb]

**Maintainer** Niels Martin <niels.martin@uhasselt.be>

**Repository** CRAN

**Date/Publication** 2022-07-14 09:00:09 UTC

Contents

daqapo . . . . .	2
detect_activity_frequency_violations . . . . .	3
detect_activity_order_violations . . . . .	4
detect_attribute_dependencies . . . . .	5
detect_case_id_sequence_gaps . . . . .	6
detect_conditional_activity_presence . . . . .	7
detect_duration_outliers . . . . .	8
detect_inactive_periods . . . . .	9
detect_incomplete_cases . . . . .	10
detect_incorrect_activity_names . . . . .	11
detect_missing_values . . . . .	12
detect_multiregistration . . . . .	13
detect_overlaps . . . . .	14
detect_related_activities . . . . .	15
detect_similar_labels . . . . .	16
detect_time_anomalies . . . . .	17
detect_unique_values . . . . .	17
detect_value_range_violations . . . . .	18
domain_categorical . . . . .	19
domain_numeric . . . . .	20
domain_time . . . . .	20
duration_within . . . . .	21
filter_anomalies . . . . .	21
fix . . . . .	22
hospital . . . . .	22
hospital_actlog . . . . .	23
hospital_events . . . . .	24
<b>Index</b>	<b>25</b>

---

daqapo	<i>daqapo - Data Quality Assessment for Process-oriented Data</i>
--------	---

---

Description

This package is designed to perform data quality assessment on process-oriented data.

---

```
detect_activity_frequency_violations
```

*Check activity frequencies*

---

## Description

Function that detects activity frequency anomalies per case

## Usage

```
detect_activity_frequency_violations(  
  activitylog,  
  ...,  
  details,  
  filter_condition  
)
```

## Arguments

activitylog	The activity log
...	Named vectors with name of the activity, and value of the threshold.
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

## Value

tbl\_df providing an overview of cases for which activities are executed too many times

## Examples

```
data("hospital_actlog")  
detect_activity_frequency_violations(activitylog = hospital_actlog,  
  "Registration" = 1, "Clinical exam" = 1)
```

---

detect\_activity\_order\_violations

*Detect activity order violations*


---

## Description

Function detecting violations in activity order. Having additional or less activity types than those specified in activity\_order is no violation, but the activity types present should occur in the specified order, and only once.

## Usage

```
detect_activity_order_violations(
  activitylog,
  activity_order,
  timestamp,
  details,
  filter_condition
)

## S3 method for class 'activitylog'
detect_activity_order_violations(
  activitylog,
  activity_order,
  timestamp = c("both", "start", "complete"),
  details = TRUE,
  filter_condition = NULL
)
```

## Arguments

activitylog	The activity log
activity_order	Vector expressing the activity order that needs to be checked (using activity names)
timestamp	Type of timestamp that needs to be taken into account in the analysis (either "start", "complete" or "both")
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

## Value

tbl\_df providing an overview of detected activity orders which violate the specified activity order

Methods (by class)

- activitylog: Detect activity order\_violations in activity log.

Examples

```
data("hospital_actlog")
detect_activity_order_violations(activitylog = hospital_actlog,
  activity_order = c(
    "Registration",
    "Triage",
    "Clinical exam",
    "Treatment",
    "Treatment evaluation"))
```

---

detect_attribute_dependencies
<i>Detect dependency violations between attributes</i>

---

Description

Function detecting violations of dependencies between attributes (i.e. condition(s) that should hold when (an)other condition(s) hold(s))

Usage

```
detect_attribute_dependencies(
  activitylog,
  antecedent,
  consequent,
  details = TRUE,
  filter_condition = NULL,
  ...
)
```

Arguments

activitylog	The activity log
antecedent	(Vector of) condition(s) which serve as an antecedent (if the condition(s) in antecedent hold, then the condition(s) in consequent should also hold)
consequent	(Vector of) condition(s) which serve as a consequent (if the condition(s) in antecedent hold, then the condition(s) in consequent should also hold)
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function
...	Named vectors with name of the activity, and value of the threshold.

**Value**

activitylog containing the rows of the original activity log for which the dependencies between attributes are violated

**Examples**

```
data("hospital_actlog")
detect_attribute_dependencies(activitylog = hospital_actlog,
  antecedent = activity == "Registration",
  consequent = startsWith(originator, "Clerk"))
```

---

```
detect_case_id_sequence_gaps
  Detect gaps in case_id
```

---

**Description**

Function detecting gaps in the sequence of case identifiers

**Usage**

```
detect_case_id_sequence_gaps(activitylog, details, filter_condition)
```

**Arguments**

activitylog	The activity log
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

**Value**

data.frame providing an overview of the case identifiers which are expected, but which are not present in the activity log

**Examples**

```
data("hospital_actlog")
detect_case_id_sequence_gaps(activitylog = hospital_actlog)
```

---

`detect_conditional_activity_presence`*Detect conditional activity presence violations*

---

**Description**

Function detecting violations of conditional activity presence (i.e. an activity/activities that should be present when (a) particular condition(s) hold(s))

**Usage**

```
detect_conditional_activity_presence(  
  activitylog,  
  condition,  
  activities,  
  details,  
  filter_condition  
)
```

**Arguments**

<code>activitylog</code>	The activity log
<code>condition</code>	Condition which serve as an antecedent (if the condition in condition holds, then the activit(y)(ies) in activities should be present.)
<code>activities</code>	Vector of activity/activities which serve as a consequent (if the condition(s) in condition_vector hold, then the activity/activities in activity_vector should be recorded)
<code>details</code>	Boolean indicating wheter details of the results need to be shown
<code>filter_condition</code>	Condition that is used to extract a subset of the activity log prior to the application of the function

**Value**

Numeric vector containing the case identifiers of cases for which the specified conditional activity presence is violated

**Examples**

```
data("hospital_actlog")  
detect_conditional_activity_presence(activitylog = hospital_actlog,  
  condition = specialization == "TRAU",  
  activities = "Clinical exam")
```

---

`detect_duration_outliers`*Detect activity duration outliers*

---

## Description

Function detecting duration outliers for a particular activity

## Usage

```
detect_duration_outliers(activitylog, ..., details, filter_condition)
```

## Arguments

<code>activitylog</code>	The activity log
<code>...</code>	for each activity to be checked, an argument "activity_name" = <code>duration_within(...)</code> to define bounds. See <code>?duration_within</code>
<code>details</code>	Boolean indicating wheter details of the results need to be shown
<code>filter_condition</code>	Condition that is used to extract a subset of the activity log prior to the application of the function

## Value

`activitylog` containing the rows of the original activity log for which activity duration outliers are detected  
Information on the presence of activity duration outliers

## See Also

[duration\\_within](#)

## Examples

```
data("hospital_actlog")
detect_duration_outliers(activitylog = hospital_actlog,
  Treatment = duration_within(bound_sd = 1))
```



---

`detect_inactive_periods`*Detect inactive periods*

---

## Description

Function detecting inactive periods, i.e. periods of time in which no activity executions/arrivals are recorded in the activity log

## Usage

```
detect_inactive_periods(  
  activitylog,  
  threshold,  
  type,  
  timestamp,  
  start_activities,  
  details,  
  filter_condition  
)
```

## Arguments

<code>activitylog</code>	The activity log
<code>threshold</code>	Threshold after which a period without activity executions/arrivals is considered as an inactive period (expressed in minutes)
<code>type</code>	Type of inactive periods you want to detect. "arrivals" will look for periods without new cases arriving. "activities" will look for periods where no activities occur.
<code>timestamp</code>	Type of timestamp that needs to be taken into account in the analysis (either "start", "complete" or "both")
<code>start_activities</code>	List of activity labels marking the first activity in the process. When specified, an inactive period only occurs when the time between two consecutive arrivals exceeds the specified threshold (arrival is proxied by the activity/activities specified in this argument).
<code>details</code>	Boolean indicating wheter details of the results need to be shown
<code>filter_condition</code>	Condition that is used to extract a subset of the activity log prior to the application of the function

## Value

`tbl_df` providing an overview of the start and end of the inactive periods that have been detected, together with the length of the inactive period

**Examples**

```
data("hospital_actlog")
detect_inactive_periods(activitylog = hospital_actlog, threshold = 30)
```

---

detect\_incomplete\_cases

*Detect incomplete cases*

---

**Description**

Function detecting incomplete cases in terms of the activities that need to be recorded for a case. The function only checks the presence of activities, not the completeness of the rows describing the activity executions.

**Usage**

```
detect_incomplete_cases(activitylog, activities, details, filter_condition)
```

**Arguments**

activitylog	The activity log
activities	A vector of activity names which should be present for a case
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

**Value**

tbl\_df providing an overview of the traces (i.e. the activities executed for a particular case) in which the specified activities are not present, together with its occurrence frequency and cases having this trace

**Examples**

```
data("hospital_actlog")
detect_incomplete_cases(activitylog = hospital_actlog,
  activities = c("Registration", "Triage", "Clinical exam", "Treatment", "Treatment evaluation"))
```

---

```
detect_incorrect_activity_names
```

*Detect incorrect activity names*

---

## Description

Function returning the incorrect activity labels in the log as indicated by the user. If details are requested, the entire activity log's rows containing incorrect activities are returned.

## Usage

```
detect_incorrect_activity_names(  
  activitylog,  
  allowed_activities,  
  details,  
  filter_condition  
)
```

## Arguments

activitylog	The activity log
allowed_activities	Vector with correct activity labels. If NULL, user input will be asked.
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

## Value

activitylog containing the rows of the original activity log having incorrect activity labels

## Examples

```
data("hospital_actlog")  
detect_incorrect_activity_names(activitylog = hospital_actlog,  
  allowed_activities = c(  
    "Registration",  
    "Triage",  
    "Clinical exam",  
    "Treatment",  
    "Treatment evaluation"))
```

---

detect\_missing\_values *Detect missing values*

---

## Description

Function detecting missing values at different levels of aggregation

- overview: presents an overview of the absolute and relative number of missing values for each column
- column: presents an overview of the absolute and relative number of missing values for a particular column
- activity: presents an overview of the absolute and relative number of missing values for each column, aggregated by activity

## Usage

```
detect_missing_values(  
  activitylog,  
  level_of_aggregation,  
  column,  
  details,  
  filter_condition  
)
```

## Arguments

activitylog	The activity log
level_of_aggregation	Level of aggregation at which missing values are identified (either "overview", "column" or "activity")
column	Column name of the column that needs to be analyzed when the level of aggregation is "column"
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

## Value

activitylog containing the rows of the original activity log which contain a missing value

**Examples**

```
data("hospital_actlog")
detect_missing_values(activitylog = hospital_actlog)
detect_missing_values(activitylog = hospital_actlog, level_of_aggregation = "activity")
detect_missing_values(activitylog = hospital_actlog, level_of_aggregation = "column",
  column = "triagecode")
```

---

detect\_multiregistration

*Detect multi-registration*

---

**Description**

Function detecting multi-registration for the same case or by the same resource at the same point in time

**Usage**

```
detect_multiregistration(
  activitylog,
  level_of_aggregation,
  timestamp,
  threshold_in_seconds,
  details,
  filter_condition
)
```

**Arguments**

activitylog	The activity log (renamed/formatted using functions <code>rename_activity_log</code> and <code>convert_timestamp_format</code> )
level_of_aggregation	Level of aggregation at which multi-registration should be detected (either "resource" or "case")
timestamp	Type of timestamp that needs to be taken into account in the analysis (either "start", "complete" or "both")
threshold_in_seconds	Threshold which is applied to determine whether multi-registration occurs (expressed in seconds) (time gaps smaller than threshold are considered as multi-registration)
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

**Value**

activitylog containing the rows of the original activity log for which multi-registration is present

**Examples**

```
data("hospital_actlog")
detect_multiregistration(activitylog = hospital_actlog, threshold_in_seconds = 10)
```

---

detect_overlaps	<i>Detect overlapping activity instances</i>
-----------------	--

---

**Description**

Detect overlapping activity instances

**Usage**

```
detect_overlaps(activitylog, details, level_of_aggregation, filter_condition)
```

**Arguments**

activitylog	The activity log
details	Boolean indicating wheter details of the results need to be shown
level_of_aggregation	Look for overlapping activity instances within a case or within a resource.
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

**Value**

tbl\_df providing an overview of activities which are performed in parallel by a resource, together with the occurrence frequency of the overlap and the average time overlap in minutes

**Examples**

```
data("hospital_actlog")
detect_overlaps(activitylog = hospital_actlog)
```

---

`detect_related_activities`*Detect missing related activities*

---

## Description

Function detecting missing related activity registration, i.e. detecting activities that should be registered for a case because another activity is registered for that case

## Usage

```
detect_related_activities(  
  activitylog,  
  antecedent,  
  consequent,  
  details,  
  filter_condition  
)
```

## Arguments

<code>activitylog</code>	The activity log
<code>antecedent</code>	Activity name of the activity that acts as a an antecedent (if antecedent occurs, then consequent should also occur)
<code>consequent</code>	Activity name of the activity that acts as a an consequent (if antecedent occurs, then consequent should also occur)
<code>details</code>	Boolean indicating wheter details of the results need to be shown
<code>filter_condition</code>	Condition that is used to extract a subset of the activity log prior to the application of the function

## Value

Numeric vector containing the case identifiers of cases for which related activities are not present

## Examples

```
data("hospital_actlog")  
detect_related_activities(activitylog = hospital_actlog,  
  antecedent = "Treatment evaluation",  
  consequent = "Treatment")
```

---

detect\_similar\_labels *Search for similar labels in a column*

---

### Description

Function that tries to detect spelling mistakes in a given activity log column

### Usage

```
detect_similar_labels(
  activitylog,
  column_labels,
  max_edit_distance,
  show_NA,
  ignore_capitals,
  filter_condition
)
```

### Arguments

activitylog	The activity log
column_labels	The name of the column(s) in which to search for spelling mistakes
max_edit_distance	The maximum number of insertions, deletions and substitutions that are allowed to be executed in order for two strings to be considered similar.
show_NA	A boolean indicating if labels that do not show similarities with others should be shown in the output
ignore_capitals	A boolean indicating if capitalization should be included or excluded when calculating the edit distance between two strings
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

### Value

tbl\_df providing an overview of similar labels for the indicated column

### Examples

```
data("hospital_actlog")
detect_similar_labels(activitylog = hospital_actlog,
  column_labels = "activity",
  max_edit_distance = 3)
```



---

`detect_time_anomalies` *Detect time anomalies*

---

**Description**

Function detecting time anomalies, which can refer to activities with negative or zero duration

**Usage**

```
detect_time_anomalies(  
  activitylog,  
  anomaly_type = c("both", "negative", "zero"),  
  details = TRUE,  
  filter_condition = NULL  
)
```

**Arguments**

<code>activitylog</code>	The activity log
<code>anomaly_type</code>	Type of anomalies that need to be detected (either "negative", "zero" or "both")
<code>details</code>	Boolean indicating wheter details of the results need to be shown
<code>filter_condition</code>	Condition that is used to extract a subset of the activity log prior to the application of the function

**Value**

`activitylog` containing the rows of the original activity log for which a negative or zero duration is detected, together with the duration value and whether it constitutes a zero or negative duration

**Examples**

```
data("hospital_actlog")  
detect_time_anomalies(activitylog = hospital_actlog)
```

---

`detect_unique_values` *Search for unique values / distinct combinations*

---

**Description**

Function that lists all distinct combinations of the given columns in the activity log

**Usage**

```
detect_unique_values(activitylog, column_labels, filter_condition = NULL)
```

**Arguments**

activitylog	The activity log
column_labels	The names of columns in the activity log for which you want to show the different combinations found in the log. If only one column is provided, this results in a list of unique values in that column.
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

**Value**

activitylog containing the unique (distinct) values (combinations) in the indicated column(s)

**Examples**

```
data("hospital_actlog")
detect_unique_values(activitylog = hospital_actlog,
  column_labels = "activity")
detect_unique_values(activitylog = hospital_actlog,
  column_labels = c("activity", "originator"))
```

---

detect\_value\_range\_violations

*Detect value range violations*

---

**Description**

Function detecting violations of the value range, i.e. values outside the range of tolerable values

**Usage**

```
detect_value_range_violations(activitylog, ..., details, filter_condition)
```

**Arguments**

activitylog	The activity log
...	Define domain range using domain_numeric, domain_categorical and/or domain_time for each column
details	Boolean indicating wheter details of the results need to be shown
filter_condition	Condition that is used to extract a subset of the activity log prior to the application of the function

**Value**

activitylog containing the rows of the original activity log for which the provided value range is violated

**See Also**

[domain\\_categorical](#), [domain\\_time](#), [domain\\_numeric](#)

**Examples**

```
data("hospital_actlog")
detect_value_range_violations(activitylog = hospital_actlog,
  triagecode = domain_numeric(from = 0, to = 5))
```

---

domain_categorical	<i>Define allowable range of values</i>
--------------------	---

---

**Description**

Define allowable range of values

**Usage**

```
domain_categorical(allowed)
```

**Arguments**

allowed	Allowed values of categorical column (character or factor)
---------	--

**Value**

No return value, called for side effects

**See Also**

[detect\\_value\\_range\\_violations](#)

---

domain_numeric	<i>Define allowable range of values</i>
----------------	---

---

**Description**

Define allowable range of values

**Usage**

```
domain_numeric(from, to)
```

**Arguments**

from	Minimum of allowed range
to	Maximum of allowed range

**Value**

No return value, called for side effects

**See Also**

[detect\\_value\\_range\\_violations](#)

---

domain_time	<i>Define allowable time range</i>
-------------	------------------------------------

---

**Description**

Define allowable time range

**Usage**

```
domain_time(from, to, format = ymd_hms)
```

**Arguments**

from	Start time interval
to	End time interval
format	Format of to and from (either ymd_hms, dmy_hms, ymd_hm, ymd, dmy, dmy, ...). Both from and to should have the same format.

**Value**

No return value, called for side effects

**See Also**[detect\\_value\\_range\\_violations](#)

---

duration_within	<i>Define bounds for activity duration</i>
-----------------	--

---

**Description**

Function to define bounds on the duration of an activity during detection of duration outliers.

**Usage**

```
duration_within(bound_sd = 3, lower_bound = NA, upper_bound = NA)
```

**Arguments**

bound_sd	Number of standard deviations from the mean duration which is used to define an outlier in the absence of lower_bound and upper_bound (default value of 3 is used)
lower_bound	Lower bound for activity duration used during outlier detection (expressed in minutes). This means disregarding the sd and bound_sd for lower bound
upper_bound	Upper bound for activity duration used during outlier detection (expressed in minutes). This means disregarding the sd and bound_sd for upper bound

**Value**

No return value, called for side effects

**See Also**[detect\\_duration\\_outliers](#)

---

filter_anomalies	<i>Filter anomalies from the activity log</i>
------------------	---

---

**Description**

Function that filters detected anomalies from the activity log

**Usage**

```
filter_anomalies(activity_log, anomaly_log)
```

**Arguments**

- activity\_log     The activity log (renamed/formatted using functions rename\_activity\_log and convert\_timestamp\_format)
- anomaly\_log     The anomaly log generated from the different DAQAPO tests

**Value**

activitylog in which the anomaly rows are filtered out

---

fix	<i>Fix problems</i>
-----	---------------------

---

**Description**

Fix problems

**Usage**

fix(detected\_problems, ...)

**Arguments**

- detected\_problems     Output of a detect\_ function. Currently supported: detect\_resource\_inconsistencies.
- ...     Additional parameters, depending on type of anomalies to fix.

**Value**

No return value, called for side effects

---

hospital	<i>An activity log of 20 patients in a hospital (data frame)</i>
----------	--

---

**Description**

A dataset containing the logged activities in an illustrative hospital process. 20 patients are described in the log. Process activities include Registration, Triage, Clinical exam, Treatment and Treatment evaluation.

**Usage**

hospital

**Format**

A data frame with 53 rows and 7 variables:

**patient\_visit\_nr** the patient's identifier

**activity** the executed activity

**originator** the resource performing the activity execution

**start\_ts** the timestamp at which the activity was started

**complete\_ts** the timestamp at which the activity was completed

**triagecode** a case attribute describing the triage code

**specialization** a case attribute describing the specialization

**Source**

An illustrative example developed in-house for demonstrational purposes.

---

hospital_actlog	<i>An activity log of 20 patients in a hospital (activity log object)</i>
-----------------	---

---

**Description**

A dataset containing the logged activities in an illustrative hospital process. 20 patients are described in the log. Process activities include Registration, Triage, Clinical exam, Treatment and Treatment evaluation.

**Usage**

```
hospital_actlog
```

**Format**

An activity log with 53 rows and 7 variables:

**patient\_visit\_nr** the patient's identifier

**activity** the executed activity

**originator** the resource performing the activity execution

**start** the timestamp at which the activity was started

**complete** the timestamp at which the activity was completed

**triagecode** a case attribute describing the triage code

**specialization** a case attribute describing the specialization

**Source**

An illustrative example developed in-house for demonstrational purposes.

---

hospital_events	<i>An event log of 20 patients in a hospital</i>
-----------------	--

---

### Description

A dataset containing the logged activities in an illustrative hospital process. 20 patients are described in this log. Process activities include Registration, Triage, Clinical exam, Treatment and Treatment evaluation.

### Usage

```
hospital_events
```

### Format

A data frame with 53 rows and 7 variables:

**patient\_visit\_nr** the patient's identifier

**activity** the executed activity

**originator** the resource performing the activity execution

**event\_lifecycle\_state** the state the activity is in at the given timestamp

**timestamp** the moment in time the lifecycle state was reached

**triagecode** a case attribute describing the triage code

**specialization** a case attribute describing the specialization

**event\_matching** a specification of which events form a pair in the log

### Source

An illustrative example developed in-house for demonstrational purposes.



# Index

## \* datasets

- hospital, [22](#)
- hospital\_actlog, [23](#)
- hospital\_events, [24](#)

daqapo, [2](#)

detect\_activity\_frequency\_violations,  
[3](#)

detect\_activity\_order\_violations, [4](#)

detect\_attribute\_dependencies, [5](#)

detect\_case\_id\_sequence\_gaps, [6](#)

detect\_conditional\_activity\_presence,  
[7](#)

detect\_duration\_outliers, [8](#), [21](#)

detect\_inactive\_periods, [9](#)

detect\_incomplete\_cases, [10](#)

detect\_incorrect\_activity\_names, [11](#)

detect\_missing\_values, [12](#)

detect\_multiregistration, [13](#)

detect\_overlaps, [14](#)

detect\_related\_activities, [15](#)

detect\_similar\_labels, [16](#)

detect\_time\_anomalies, [17](#)

detect\_unique\_values, [17](#)

detect\_value\_range\_violations, [18](#),  
[19–21](#)

domain\_categorical, [19](#), [19](#)

domain\_numeric, [19](#), [20](#)

domain\_time, [19](#), [20](#)

duration\_within, [8](#), [21](#)

filter\_anomalies, [21](#)

fix, [22](#)

hospital, [22](#)

hospital\_actlog, [23](#)

hospital\_events, [24](#)