Package: sdtmchecks (via r-universe)

September 5, 2024

Title Data Quality Checks for Study Data Tabulation Model (SDTM) Datasets

Version 1.0.0

Description A series of checks to identify common issues in Study Data Tabulation Model (SDTM) datasets. These checks are intended to be generalizable, actionable, and meaningful for analysis.

License Apache License (>= 2)

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports dplyr (>= 1.1.1), tidyselect, openxlsx

Suggests knitr, rmarkdown (>= 2.7), testthat, DT

Config/testthat/edition 3

VignetteBuilder knitr

URL https://pharmaverse.github.io/sdtmchecks/, https://github.com/pharmaverse/sdtmchecks

BugReports https://github.com/pharmaverse/sdtmchecks/issues

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

RemoteUrl https://github.com/pharmaverse/sdtmchecks

RemoteRef HEAD

RemoteSha ccb59be39c05811c6b94e848a99f62b00a381ef2

Contents

`check_ae_aeacnoth` *Check for null AEACNOT1/2 when AEACNOTH = 'MULTIPLE'*

Description

Flag if patient has a record with null values of AEACNOT1 and AEACNOT2 but AEACNOTH = 'MULTIPLE', so a likely mapping issue

Usage

```
check_ae_aeacnoth(AE, preproc = identity, ...)
```

Arguments

`AE`	Adverse Events SDTM dataset with variables USUBJID, AETERM, AESTDTC, AEACNOTH, AEACNOT1/2, AESPID (optional)
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Ross Farrugia

Examples

```
AE <- data.frame(
 USUBJID = 1:7,
 AETERM = 1:7,
 AESTDTC = 1:7,
 AEACNOTH = 1:7,
 AEACNOT1 = 1:7,
 AEACNOT2 = 1:7,
 AESPID = "FORMNAME-R:13/L:13XXXX"
)

# pass
check_ae_aeacnoth(AE)

AE$AEACNOTH[1] = ""
AE$AEACNOT1[1] = ""
AE$AEACNOT2[1] = ""
AE$AEACNOTH[2] = "MULTIPLE"
AE$AEACNOT1[2] = "DOSE REDUCED"
AE$AEACNOT2[2] = "DRUG WITHDRAWN"
AE$AEACNOTH[3] = "MULTIPLE"
AE$AEACNOT1[3] = "DOSE REDUCED"
AE$AEACNOT2[3] = ""
AE$AEACNOTH[4] = "MULTIPLE"
AE$AEACNOT1[4] = ""
AE$AEACNOT2[4] = "DRUG WITHDRAWN"
AE$AEACNOTH[5] = "MULTIPLE"
AE$AEACNOT1[5] = ""
AE$AEACNOT2[5] = ""

# fail
check_ae_aeacnoth(AE)
check_ae_aeacnoth(AE,preproc=roche_derive_rave_row)

AE$AEACNOTH[1] = NA
AE$AEACNOT1[1] = NA
AE$AEACNOT2[1] = NA
AE$AEACNOT2[3] = NA
AE$AEACNOT1[4] = NA
AE$AEACNOT1[5] = NA
AE$AEACNOT2[5] = NA

# fail
check_ae_aeacnoth(AE)
check_ae_aeacnoth(AE,preproc=roche_derive_rave_row)


AE$AEACNOTH <- NULL
AE$AEACNOT1 <- NULL
AE$AEACNOT2 <- NULL
AE$AESPID <- NULL
check_ae_aeacnoth(AE)
```

`check_ae_aeacnoth_ds_disctx` *Check if, whenever a patient experiences an AE leading to study discontinuation, they also have a DS record indicating this.*

Description

This code checks that when a patient has an AE with AEACNOTx = "SUBJECT DISCONTINUED FROM STUDY" (x = "H", "1", "2", ...) then there should also be a record in DS where DS.DSSCAT = "STUDY COMPLETION/EARLY DISCONTINUATION" and DS.DSDECOD != "COMPLETED".

Usage

```
check_ae_aeacnoth_ds_disctx(AE, DS, preproc = identity, ...)
```

Arguments

`AE`	Adverse Events SDTM dataset with variables USUBJID, AEDECOD, AEACNOTx
`DS`	Disposition SDTM dataset with variables USUBJID, DSCAT, DSSCAT, DSDECOD
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check returns 0 obs, otherwise return subset dataframe.

Author(s)

Edoardo Mancini

Examples

```
AE <- data.frame(
    STUDYID  = "1001",
    USUBJID  = c("1","2","3","4","5","1"),
    AESTDTC  = rep('2020-05-05', 6),
   AEDECOD  = c("HEADACHE", "HEART ATTACK","CHILLS", "PNEUMONIA", "ARTHRITIS", "FATIGUE"),
    AEACNOTH = c("NONE", "SUBJECT DISCONTINUED FROM STUDY", "MULTIPLE", "NONE",
                 "SUBJECT DISCONTINUED FROM STUDY", "SUBJECT DISCONTINUED FROM STUDY"),
    AEACNOT1 = c("", "", "PROCEDURE/SURGERY", "", "", ""),
    AEACNOT2 = c("", "", "SUBJECT DISCONTINUED FROM STUDY", "", "", ""),
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
```

```
)

DS <- data.frame(
    USUBJID = c("1","5"),
    DSCAT   = c("DISPOSITION EVENT", "DISPOSITION EVENT"),
   DSSCAT  = c("STUDY COMPLETION/EARLY DISCONTINUATION", "STUDY COMPLETION/EARLY DISCONTINUATION"),
    DSDECOD = c("ADVERSE EVENT", "ADVERSE EVENT" ),
    stringsAsFactors = FALSE
)

check_ae_aeacnoth_ds_disctx(AE, DS)
check_ae_aeacnoth_ds_disctx(AE, DS, preproc=roche_derive_rave_row)
```

`check_ae_aeacnoth_ds_stddisc_covid` *Check for COVID-19 AE leading to Study Discon without DS Study Discon*

Description

Flag if patient has a COVID-19 AE where AE.AEDECOD matches a COVID-19 preferred term event action of Study Discontinuation (AE.AEACNOT* includes "DISCONTINUED FROM STUDY") but missing Study Discontinuation record in DS (DS.DSSCAT includes "STUDY" and "DISCON" and excludes "DRUG" and "TREATMENT")

Usage

```
check_ae_aeacnoth_ds_stddisc_covid(
  AE,
  DS,
  covid_terms = c("COVID-19", "CORONAVIRUS POSITIVE")
)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AEDECOD, AEACNOT* (can be multiple variables)
DS	Disposition SDTM dataset with variables USUBJID, DSSCAT, DSDECOD
covid_terms	A length >=1 vector of AE terms identifying COVID-19 (case does not matter)

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Natalie Springfield

See Also

Other COVID: check_ae_aeacn_ds_disctx_covid(), check_dv_ae_aedecod_covid(), check_dv_covid()

Examples

```
AE <- data.frame(
 USUBJID = 1:5,
 AEDECOD = c("This is a covid AE", "covid-19", "covid-19","Some AE", "CORONAVIRUS POSITIVE" ),
 AEACNOTH=c("SUBJECT DISCONTINUED FROM STUDY",
            "NONE",
            "NONE",
            "SUBJECT DISCONTINUED FROM STUDY",
            "NONE"),
 AEACNOTH1=c("SUBJECT DISCONTINUED FROM STUDY",
             "NONE",
             "SUBJECT DISCONTINUED FROM STUDY",
             "NONE",
             "SUBJECT DISCONTINUED FROM STUDY"),
 AEACNOTH2=c("SUBJECT DISCONTINUED FROM STUDY",
             "NONE",
             "NONE",
             "SUBJECT DISCONTINUED FROM STUDY",
             "NONE")
)

DS <- data.frame(
 USUBJID = 1:3,
 DSSCAT=c("TREATMENT DISCONTINUATION",
 "STUDY DISCONTINUATION",
 "STUDY DISCONTINUATION"),
 DSDECOD="DISCON REASON"
)

#expect fail
check_ae_aeacnoth_ds_stddisc_covid(AE,DS)

#use custom terms for identifying covid AEs
check_ae_aeacnoth_ds_stddisc_covid(
  AE,
  DS,
  covid_terms=c("COVID-19", "CORONAVIRUS POSITIVE","THIS IS A COVID AE")
  )
```

---

check_ae_aeacn_ds_disctx_covid

Check for COVID-19 AE with DRUG WITHDRAWN action without "ADVERSE EVENT" for DS Trt Discon

---

Description

This checks for a COVID-19 AE reported with Action Taken (AEACN*==DRUG WITHDRAWN) but without a corresponding DS record indicating DS.DSDECOD with "ADVERSE EVENT" on any Treatment Discontinuation form. This relies on DSSPID with the string "DISCTX" when DSCAT == "DISPOSITION EVENT" to select Treatment Discontinuation records in DS, as DSSCAT typically includes a text string referring to specific study drug(s).

Usage

```
check_ae_aeacn_ds_disctx_covid(
  AE,
  DS,
  covid_terms = c("COVID-19", "CORONAVIRUS POSITIVE")
)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AETERM, AEDECOD, AESTDTC, AEACNx
DS	Disposition SDTM dataset with variables USUBJID, DSSPID, DSCAT, DSDECOD
covid_terms	A length >=1 vector of AE terms identifying COVID-19 (case does not matter)

Value

boolean value if check returns 0 obs, otherwise return subset dataframe.

Author(s)

Sarwan Singh

See Also

Other COVID: check_ae_aeacnoth_ds_stddisc_covid(), check_dv_ae_aedecod_covid(), check_dv_covid()

Examples

```
AE <- data.frame(
    STUDYID = 1,
    USUBJID = c(1,2,3,1,2,3),
    AESTDTC = '2020-05-05',
    AETERM  = c("abc Covid-19", "covid TEST POSITIVE",rep("other AE",4)),
    AEDECOD = c("COVID-19", "CORONAVIRUS POSITIVE", rep("OTHER AE",4)),
    AEACN = c("DRUG WITHDRAWN", rep("DOSE NOT CHANGED",5)),
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

DS <- data.frame(
 USUBJID = c(1,1,2,3,4),
```

```
 DSSPID  = 'XXX-DISCTX-XXX',
 DSCAT   = "DISPOSITION EVENT",
 DSDECOD = "OTHER REASON",
 DSSEQ = c(1,2,1,1,1),
 stringsAsFactors = FALSE
)

# expect fail
check_ae_aeacn_ds_disctx_covid(AE, DS)


AE2 <- data.frame(
    AEACN1 =  rep(NA, nrow(AE)),
    AEACN2 = c("DRUG WITHDRAWN", rep("DOSE NOT CHANGED", nrow(AE)-1)),
    AEACN3 = c("DRUG WITHDRAWN", rep("DOSE NOT CHANGED", nrow(AE)-1)),
    AEACN4 = "",
    stringsAsFactors = FALSE
)
AE2 <- cbind(AE, AE2)
AE2$AEACN <- "MULTIPLE"

# expect fail
check_ae_aeacn_ds_disctx_covid(AE2, DS)

DS[1, "DSDECOD"] <- 'ADVERSE EVENT'
# this passes, one form with DSDECOD = "ADVERSE EVENT"
## NOTE: This may be a false negative if study-specific data collection
##       requires >1 record with DSDECOD = "ADVERSE EVENT" and not just one record
check_ae_aeacn_ds_disctx_covid(AE2, DS)

# expect pass
check_ae_aeacn_ds_disctx_covid(AE, DS)


# non-required variable is missing
DS$DSSEQ <- NULL
check_ae_aeacn_ds_disctx_covid(AE, DS)

# required variable is missing
DS$DSSPID <- NULL
check_ae_aeacn_ds_disctx_covid(AE, DS)
```

---

check_ae_aedecod *Check for missing AEDECOD values*

Description

This check looks for missing AEDECOD values

Usage

```
check_ae_aedecod(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AETERM, AEDECOD
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Yinghui Miao, Stella Banjo(HackR 2021)

Examples

```
AE <- data.frame(
 USUBJID = 1:5,
 DOMAIN = c(rep("AE", 5)),
 AESEQ = 1:5,
 AESTDTC = 1:5,
 AETERM = 1:5,
 AEDECOD = 1:5,
  AESPID = c("FORMNAME-R:13/L:13XXXX",
             "FORMNAME-R:16/L:16XXXX",
             "FORMNAME-R:2/L:2XXXX",
             "FORMNAME-R:19/L:19XXXX",
             "FORMNAME-R:5/L:5XXXX"),
 stringsAsFactors = FALSE
)

check_ae_aedecod(AE)

AE$AEDECOD[1] = NA
AE$AEDECOD[2] = "NA"
AE$AEDECOD[3:5] = ""
check_ae_aedecod(AE)
check_ae_aedecod(AE,preproc=roche_derive_rave_row)

AE$AEDECOD <- NULL
check_ae_aedecod(AE)
```

check_ae_aedthdtc_aesdth *Check AEs with AEDTHDTC value but AESDTH not "Y"*

Description

This check looks for AE entries with an AEDTHDTC (death date) value and AESDTH not equal to "Y"

Usage

```
check_ae_aedthdtc_aesdth(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Event SDTM dataset with variables USUBJID, AEDTHDTC, AESDTH, AEDECOD, AETERM, and AESTDTC
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Shumei Chi

Examples

```
AE <- data.frame(
USUBJID = c(1:7),
AEDECOD = c(letters[1:5], "", NA),
AETERM = letters[1:7],
AESDTH = "Y",
AEDTHDTC = "2020-01-02",
AESTDTC = c(1:7),
AESPID = "FORMNAME-R:5/L:5XXXX",
stringsAsFactors=FALSE)

# expect pass
check_ae_aedthdtc_aesdth(AE)
check_ae_aedthdtc_aesdth(AE,preproc=roche_derive_rave_row)

# expect fail
AE1 <- AE
AE1$AESDTH[3] <- "N"
check_ae_aedthdtc_aesdth(AE1)
```

```
check_ae_aedthdtc_aesdth(AE1,preproc=roche_derive_rave_row)

# expect fail with AESDTH = NA
AE2 <- AE
AE2$AESDTH[4] <- NA
check_ae_aedthdtc_aesdth(AE2)
check_ae_aedthdtc_aesdth(AE2,preproc=roche_derive_rave_row)

# non-required variable missing
AE2$AESPID <- NULL
check_ae_aedthdtc_aesdth(AE2)
check_ae_aedthdtc_aesdth(AE2,preproc=roche_derive_rave_row)

# required variable missing
AE2$AESDTH <- NULL
check_ae_aedthdtc_aesdth(AE2)
check_ae_aedthdtc_aesdth(AE2,preproc=roche_derive_rave_row)
```

`check_ae_aedthdtc_ds_death` *Check for missing AEDTHDTC where DS indicates death due to AE*

Description

This check looks for missing AEDTHDTC values if a patient has a DS record where DSDECOD=DEATH and DSTERM contains ADVERSE EVENT

Usage

```
check_ae_aedthdtc_ds_death(AE, DS)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AEDTHDTC
DS	Disposition SDTM dataset with variables USUBJID, DSDECOD, DSTERM, DSSTDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Aldrich Salva

Examples

```
AE <- data.frame(
 USUBJID = 1:3,
 AEDTHDTC = c(NA,NA,1)
)


# older mapping
DS <- data.frame(
 USUBJID = 1:4,
 DSTERM = c("DEATH DUE TO ADVERSE EVENT","DEATH DUE TO PROGRESSIVE DISEASE",
            "DEATH DUE TO ADVERSE EVENT","DEATH DUE TO ADVERSE EVENT")
            ,
 DSDECOD = rep("DEATH",4),
 DSSTDTC = "2020-01-01"
)

check_ae_aedthdtc_ds_death(AE,DS)

DS$DSSTDTC = NULL

check_ae_aedthdtc_ds_death(AE,DS)

# newer mapping that
DS <- data.frame(
 USUBJID = 1:4,
 DSTERM = c("DEATH DUE TO MYOCARDIAL INFARCTION","DEATH DUE TO PROGRESSIVE DISEASE",
            "DEATH DUE TO COVID-19","DEATH")
            ,
 DSDECOD = rep("DEATH",4),
 DSSTDTC = "2020-01-01"
 )

# pass for study with newer mapping, as another function (check_dd_death_date.R) covers this
check_ae_aedthdtc_ds_death(AE,DS)
```

`check_ae_aelat` *Check if AESOC has Eye, and Affected Eye is missing*

Description

This check looks if AESOC has Eye, and AELAT is missing.

Usage

```
check_ae_aelat(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Event Dataset for Ophtho Study with variables USUBJID, AELAT, AESOC, AEDECOD, AETERM, AESTDTC (if present), AESPID (if present)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah (HackR 2021 Team Eye)

See Also

Other OPHTH: check_cm_cmlat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_early_tot(), check_oe_bcva_4m_late_early_tot(), check_oe_bcva_4m_vs_1m_req(), check_oe_bcva_tot_mismatch(), check_oe_sc_lat_count_fingers(), check_pr_prlat(), check_sc_dm_eligcrit(), check_sc_dm_seyeselc()

Examples

```
AE <- data.frame(
   USUBJID = 1:5,
   AESTDTC = 1:5,
   AELOC   = c("", "EYE", "eye", "", "EYE"),
   AELAT   = c("Left", "","left", "RIGHT", ""),
   AETERM  = c("A", "B", "A", "B", "A"),
   AEDECOD = c("A", "B", "A", "B", "A"),
   AESOC   = c("Eye", "Eye","Eye Disorder","Eye Disorder", "Eye"),
   AESPID  = "FORMNAME-R:19/L:19XXXX",
   stringsAsFactors = FALSE)

check_ae_aelat(AE)
check_ae_aelat(AE,preproc=roche_derive_rave_row)

AE <- data.frame(
   USUBJID = 1:5,
   AESTDTC = 1:5,
   AELAT   = c("Left", "","Bilateral", "", ""),
   AETERM  = c("A", "B", "A", "B", "A"),
   AEDECOD = c("A", "B", "A", "B", "A"),
   AESOC   = c("Eye", "Eye","Eye Disorder","Eye Disorder", "Eye"),
   stringsAsFactors = FALSE)

check_ae_aelat(AE)
check_ae_aelat(AE,preproc=roche_derive_rave_row)
```

check_ae_aeout *Check for inconsistency between AE outcome (AEOUT) and death date (AEDTHDTC)*

Description

This check looks for AEs with Death date(AEDTHDTC) but outcome (AEOUT) is not FATAL and conversely AEs with no death date (AEDTHDTC) but outcome (AEOUT) is fatal

Usage

```
check_ae_aeout(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AEDTHDTC, AEOUT
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Shumei Chi

Examples

```
AE <- data.frame(
    USUBJID = 1:8,
   AEDTHDTC = c(NA, "NA", "2015-03-12", "2017-01-22", "1999-11-07","",NA, "2020-01-01"),
    AEOUT = c("", "", "","FATAL","RECOVERED/RESOLVED","FATAL","FATAL", NA),
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

check_ae_aeout(AE)
check_ae_aeout(AE,preproc=roche_derive_rave_row)

AE$AEDTHDTC <- NULL
check_ae_aeout(AE)

AE$AEOUT <- NULL
check_ae_aeout(AE)
```

check_ae_aeout_aeendtc_aedthdtc *Check for AE outcome (AEOUT) of 'FATAL' with non-missing resolution date that is not equal to the death date*

Description

This check looks for AEs with outcome of 'FATAL' but AE resolution date is not equal to AE death date. Note that these datapoints are not collected the same way for all trials - some trials leave AEENDTC missing if it was unresolved at death date. Confirm within your team before querying this issue.

Usage

```
check_ae_aeout_aeendtc_aedthdtc(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AETERM, AEDTHDTC, AEENDTC, AEOUT
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach, Stella Banjo(HackR 2021)

Examples

```
AE <- data.frame(
    USUBJID = 1:10,
    DOMAIN = "AE",
    AEDTHDTC = c(NA, "NA", rep("2015-03-12",4), NA, NA, "2020-01-01", ""),
    AEENDTC = c(NA, "NA", rep("2015-03-12",4), NA, "2020-01-01", NA, ""),
    AEOUT = c("", "", "","FATAL","RECOVERED/RESOLVED", rep("FATAL",5)),
    AETERM = 1:10,
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

check_ae_aeout_aeendtc_aedthdtc(AE)
check_ae_aeout_aeendtc_aedthdtc(AE,preproc=roche_derive_rave_row)

AE$AESPID <- NULL
```

```
check_ae_aeout_aeendtc_aedthdtc(AE)

AE$AEDTHDTC <- NULL
AE$AEOUT <- NULL
check_ae_aeout_aeendtc_aedthdtc(AE)
```

`check_ae_aeout_aeendtc_nonfatal` *Check for non-fatal AEs with inconsistent AEOUT and AEENDTC*

Description

Check for inconsistency between AE outcome (AEOUT) and AE end date (AEENDTC) for non-fatal AEs (based on AEOUT). AE flagged if AEENDTC not populated when AEOUT is "RECOVERED/RESOLVED", "RECOVERED/RESOLVED WITH SEQUELAE". AE also flagged if AEENDTC is populated when AEOUT is "UNKNOWN", "NOT RECOVERED/NOT RESOLVED", "RECOVERING/RESOLVING".

Usage

```
check_ae_aeout_aeendtc_nonfatal(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AETERM, AESTDTC, AEENDTC, AEOUT
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Jennifer Lomax

Examples

```
AE <- data.frame(
USUBJID = 1:10,
AETERM  = "AE",
AESTDTC = c(NA, "NA", "2015-03-09", "2010-10", "2017-01-20",   "1999-11-02",
        "",     NA,                   "2017-08-20",          "2014-12-01"),
AEENDTC = c(NA, "NA", "2015-03-12", "2010-10", "2017-01-22",   "1999-11-07",
        "",     NA,                   "2017-09-01",          "2015-01-01"),
AEOUT   = c("", "",   "",           "",        "NOT RECOVERED",
```

```
"RECOVERED/RESOLVED","FATAL","RECOVERED/RESOLVED", "RECOVERING/RESOLVING","UNKNOWN"),
AESPID  = "FORMNAME-R:13/L:13XXXX",
stringsAsFactors = FALSE
)

check_ae_aeout_aeendtc_nonfatal(AE)
check_ae_aeout_aeendtc_nonfatal(AE,preproc=roche_derive_rave_row)

AE$AEENDTC <- NULL
check_ae_aeout_aeendtc_nonfatal(AE)

AE$AEOUT <- NULL
check_ae_aeout_aeendtc_nonfatal(AE)
```

`check_ae_aerel` *Check for AEREL1 - AERELN when AEREL is missing and when AEREL is unexpected*

Description

Flag if patient has a record with null value of AEREL but AEREL1 - AERELN contain 'Y'/'N'/'NA', so a likely mapping issue or if AEREL is missing and there is no any AERELn variable or if AEREL has unexpected value

Usage

```
check_ae_aerel(AE, preproc = identity, ...)
```

Arguments

`AE`	Adverse Events SDTM dataset with variables USUBJID, AESEQ, AETERM, AESTDTC, AEREL, AERELn, AESPID (if present)
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Vira Vrakina

Examples

```
AE <- data.frame(
    STUDYID = 1001,
    USUBJID = c(1,2,3,1,2,3),
    AESTDTC = rep('2020-05-05',6),
    AETERM  = c("abc Covid-19", "covid TEST POSITIVE","CHILLS"),
    AESEQ   = c(1,1,1,2,2,2),
    AEREL   = c("Y", "N", "NA", "N", "N", "Y"),
    AEREL1  = c("Y", "N", "NA", "N", "NA", "Y"),
    AEREL2  = c("Y", "N", "NA", "N", "N", "N"),
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

check_ae_aerel(AE)

AE1 <- data.frame(
    STUDYID = 1001,
    USUBJID = c(1,2,3,1,2,3),
    AESTDTC = rep('2020-05-05',6),
    AETERM  = c("abc Covid-19", "covid TEST POSITIVE","CHILLS"),
    AESEQ   = c(1,1,1,2,2,2),
    AEREL   = c("Y", "N", "N", "N", "N", "N"),
    AEREL1  = c("Y", "N", "NA", "N", "N", ""),
    AEREL2  = c("Y", "N", " ", "N", "N", " "),
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

check_ae_aerel(AE1)
check_ae_aerel(AE1,preproc=roche_derive_rave_row)

AE2 <- data.frame(
    STUDYID = 1001,
    USUBJID = c(1,2,3,1,2,3),
    AESTDTC = rep('2020-05-05',6),
    AETERM  = c("abc Covid-19", "covid TEST POSITIVE","CHILLS"),
    AESEQ   = c(1,1,1,2,2,2),
    AEREL   = c("Y", "N", " ", "N", "N", " "),
    AEREL1  = c("NA", "N", "NA", "Y", "N", " "),
    AEREL2  = c("Y", "N", " ", "N", "N", " "),
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

check_ae_aerel(AE2)
check_ae_aerel(AE2,preproc=roche_derive_rave_row)

AE3 <- data.frame(
    STUDYID = 1001,
    USUBJID = c(1,2,3,1,2,3),
    AESTDTC = rep('2020-05-05',6),
```

```
    AETERM  = c("abc Covid-19", "covid TEST POSITIVE","CHILLS"),
    AESEQ   = c(1,1,1,2,2,2),
    AEREL   = c("Y", " ", " ", "N", " ", "NA"),
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

check_ae_aerel(AE3)
check_ae_aerel(AE3,preproc=roche_derive_rave_row)


AE4 <- data.frame(
    STUDYID = 1001,
    USUBJID = c(1,2,3,4,5,6),
    AESTDTC = rep('2020-05-05',6),
    AETERM  = c("abc Covid-19", "covid TEST POSITIVE","CHILLS"),
    AESEQ   = c(1,2,3,4,5,6),
    AEREL   = c("Y", "Y", "N", "", "Y", "NA"),
    AEREL1  = "",
    AEREL2  = "",
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

check_ae_aerel(AE4)
check_ae_aerel(AE4,preproc=roche_derive_rave_row)
```

`check_ae_aesdth_aedthdtc` *Check AEs with AESDTH of "Y" but No AEDTHDTC Value*

Description

This check looks for AE entries with AESDTH of "Y" but no AEDTHDTC (death date) value

Usage

```
check_ae_aesdth_aedthdtc(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Event SDTM dataset with variables USUBJID, AEDTHDTC, AESDTH, AETERM, AEDECOD and AESTDTC
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Shumei Chi

Examples

```
AE <- data.frame(
USUBJID = c(1:7),
AEDECOD = c(letters[1:5], "", NA),
AETERM = letters[1:7],
AESDTH = c(NA, rep("", 4), "Y", "Y"),
AEDTHDTC = c(1:5, "2020", "2020-01-02"),
AESTDTC = c(1:7),
AESPID = "FORMNAME-R:5/L:5XXXX",
stringsAsFactors=FALSE)

# expect pass
check_ae_aesdth_aedthdtc(AE)
check_ae_aesdth_aedthdtc(AE,preproc=roche_derive_rave_row)

# expect fail
AE1 <- AE
AE1$AEDTHDTC[3] <- NA
AE1$AESDTH[3] <- "Y"

check_ae_aesdth_aedthdtc(AE1)
check_ae_aesdth_aedthdtc(AE1,preproc=roche_derive_rave_row)

# expect fail
AE2 <- AE1
AE2$AEDTHDTC[4] <- ""
AE2$AESDTH[4] <- "Y"

check_ae_aesdth_aedthdtc(AE2)
check_ae_aesdth_aedthdtc(AE2,preproc=roche_derive_rave_row)

# non-required variable missing
AE2$AESPID <- NULL
check_ae_aesdth_aedthdtc(AE2)
check_ae_aesdth_aedthdtc(AE2,preproc=roche_derive_rave_row)

# required variable missing
AE2$AESDTH <- NULL
check_ae_aesdth_aedthdtc(AE2)
check_ae_aesdth_aedthdtc(AE2,preproc=roche_derive_rave_row)
```

---

check_ae_aestdtc_after_aeendtc

Check that all AE start dates are on or before AE end dates

Description

This check identifies AESTDTC values that are after AEENDTC values

Usage

```
check_ae_aestdtc_after_aeendtc(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Event SDTM dataset with variables USUBJID,AETERM,AEDECOD,AESTDTC,AEENDTC
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
AE <- data.frame(
 USUBJID = 1:12,
 AETERM = "SOME AE TERM",
 AEDECOD = "SOME AE PT",
 AESTDTC = c("2017-01-01","2017-01-03","2017-01-01T14:26","2017","2017-02","2017"    ,""    ,
         "2017"  ,"2017-01-01T14:26","2017-01-01T14:26","2017-01-01T14","2017-01-01T14:26:02")
              ,
 AEENDTC = c("2017-01-01","2017-01-02","2017-01-01T14:25","2015","2017-01","2016-01-01",
         "2000","2017-02","2017-01-01"    ,"2017-01","2017-01-01T13","2017-01-01T14:26:01")
              ,
 AESPID = "FORMNAME-R:19/L:19XXXX",
 stringsAsFactors=FALSE
)

check_ae_aestdtc_after_aeendtc(AE)
check_ae_aestdtc_after_aeendtc(AE,preproc=roche_derive_rave_row)

AE$AETERM <- NULL
check_ae_aestdtc_after_aeendtc(AE)
```

check_ae_aestdtc_after_dd *Check for AE dates occurring after death date*

Description

This check looks for AE dates that occur after death date

Usage

```
check_ae_aestdtc_after_dd(AE, DS, preproc = identity, ...)
```

Arguments

AE	Adverse Event SDTM dataset with variables USUBJID, AEDTHDTC, AESTDTC, AEDECOD, AETERM, AESPID (optional)
DS	Disposition SDTM dataset with variables USUBJID, DSSTDTC, DSDECOD, DSTERM, DSSPID (optional)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Nina Ting Qi

Examples

```
AE <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 AEDTHDTC = c(rep("", 4), "2016-01-01"),
                 AESTDTC = rep("2016-01-01", 5),
                 AEDECOD = c("","", rep("Myocarditis",3)),
                 AETERM = c("INJURY", rep("MYOCARDITIS", 4)),
                 AESPID = "FORMNAME-R:19/L:19XXXX",
                 stringsAsFactors = FALSE)

DS <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 DSSTDTC = rep("2016-01-02", 5),
                 DSDECOD = c(LETTERS[1:4], "death"),
                 DSSPID = "XXX-R:0",
                 DSTERM = letters[1:5],
                 stringsAsFactors = FALSE)

check_ae_aestdtc_after_dd(AE,DS)
```

```
AE$AESTDTC[1] <- "2016-01-03"
AE$USUBJID[1] <- AE$USUBJID[5]

check_ae_aestdtc_after_dd(AE, DS,preproc=roche_derive_rave_row)

AE$AESPID <- NULL
check_ae_aestdtc_after_dd(AE, DS)

DS$DSSPID <- NULL
check_ae_aestdtc_after_dd(AE, DS)

AE$AESTDTC <- NULL
check_ae_aestdtc_after_dd(AE, DS)
```

---

check_ae_aetoxgr *Check for missing AETOXGR and/or AESEV values*

Description

This check looks for missing AETOXGR and/or AESEV values and returns a data frame. If both variables exist it returns records where both are missing.

Usage

```
check_ae_aetoxgr(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AESTDTC, AEDECOD, AETERM, and AETOXGR (or AESEV)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Will Harris, Stella Banjo (HackR 2021)

Examples

```
# test with sample data


AE <- data.frame(
 USUBJID = 1:3,
 DOMAIN = c(rep("AE", 3)),
 AESEQ = 1:3,
 AESTDTC = 1:3,
 AETERM = c("FLU COUGH", "HEADACHE", "FEVER"),
 AEDECOD = c("", "Headache", "Fever"),
 AETOXGR = 1:3,
 AESEV = 1:3,
 AESPID = "FORMNAME-R:16/L:16XXXX",
 stringsAsFactors = FALSE
)

check_ae_aetoxgr(AE)

AE$AETOXGR[1] <- NA
check_ae_aetoxgr(AE)

AE$AESEV[1] <- NA
check_ae_aetoxgr(AE,preproc=roche_derive_rave_row)

AE$AETOXGR <- NULL
check_ae_aetoxgr(AE,preproc=roche_derive_rave_row)

AE$AESPID <- NULL
check_ae_aetoxgr(AE,preproc=roche_derive_rave_row)

AE$AESEV <- NULL
check_ae_aetoxgr(AE)

AE$AEDECOD <- NULL
check_ae_aetoxgr(AE)
```

---

check_ae_death *Check for Grade 5 AE death variable consistency*

Description

Checks for grade 5 AEs not marked fatal (AEOUT), death not indicated (AESDTH), or no death date (AESDTHDTC)

Usage

```
check_ae_death(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Event dataframe with variables USUBJID,AETOXGR,AEOUT,AEDTHDTC,AESDTH
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Iris Zhao

Examples

```
 AE <- data.frame(
 USUBJID = 1:10,
 AETOXGR = c(1:5,5,5,5,5,5),
 AEDTHDTC = c(rep(NA,4),rep("2020-01-01",6)),
 AESDTH = c(rep(NA,4),rep("Y",6)),
 AEOUT = c(rep(NA,4),rep("FATAL",6)),
  AESPID = "FORMNAME-R:13/L:13XXXX"
)

check_ae_death(AE)
check_ae_death(AE,preproc=roche_derive_rave_row)

AE$AEDTHDTC[5]="NA"
AE$AEDTHDTC[6]=NA
AE$AEDTHDTC[7]=""
AE$AESDTH[8]=NA
AE$AEOUT[9]=NA

check_ae_death(AE)
check_ae_death(AE,preproc=roche_derive_rave_row)
```

---

check_ae_death_ds_discon

Check if death in AE then there should be a study discon form

---

Description

This checks that if death is indicated in AE via AEDTHDTC/AESDTH/AEOUT (as well as grade 5 AE if AETOXGR exists) then there should be a study discontinuation record indicated by DS.DSSCAT

Usage

```
check_ae_death_ds_discon(AE, DS, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AEDTHDTC, AESDTH, AEOUT
DS	Disposition SDTM dataset with variables USUBJID, DSCAT, DSSCAT
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
AE <- data.frame(
 STUDYID = rep(1,6),
 USUBJID = 1:6,
 AEDTHDTC = c(NA,"2020-01-01",NA,NA,NA,NA),
 AESDTH = c(NA,NA,"Y",NA,NA,NA),
 AEOUT = c(NA,NA,NA,"FATAL",NA,NA),
 AETOXGR = c(NA,NA,NA,NA,"5",NA),
 AESPID="FORMNAME-R:2/L:2XXXX"
)

DS <- data.frame(
 STUDYID = 1,
 USUBJID = 1:3,
 DSCAT="DISPOSITION EVENT",
 DSSCAT=c("STUDY DISCON",
 "STUDY DISCON",
 "STUDY COMPLETION/EARLY DISCONTINUATION")
)

check_ae_death_ds_discon(AE,DS)
check_ae_death_ds_discon(AE,DS,preproc=roche_derive_rave_row)

DS$DSSCAT = NULL

check_ae_death_ds_discon(AE,DS)
```

`check_ae_ds_partial_death_dates` *Check for partial death dates in AE and DS*

Description

This checks looks for partial death dates in AE and DS

Usage

```
check_ae_ds_partial_death_dates(AE, DS, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID,AEDTHDTC,AEDECOD
DS	Dispostion SDTM dataset with variables USUBJID,DSSCAT,DSSTDTC,DSDECOD
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Will Harris

Examples

```
# test with sample data

AE <- data.frame(
 USUBJID = 1:3,
 AEDECOD = c("AE1","AE2","AE3"),
 AEDTHDTC = c("2017-01-01","2017",NA),
 AESPID = "FORMNAME-R:2/L:2XXXX",
 stringsAsFactors=FALSE
)

DS <- data.frame(
 USUBJID = 1:4,
 DSSCAT = "STUDY DISCON",
 DSDECOD = "DEATH",
 DSSTDTC = c("2017-01-01","2017","2017-01-02","2016-10"),
 stringsAsFactors=FALSE
)

check_ae_ds_partial_death_dates(AE,DS)
```

```
check_ae_ds_partial_death_dates(AE,DS,preproc=roche_derive_rave_row)

DS$DSSTDTC = NULL

check_ae_ds_partial_death_dates(AE,DS)
```

---

check_ae_dup *Check for duplicate AE entries*

Description

Identifies duplicated AE entries based on USUBJID, AETERM, AEDECOD, AESTDTC, AEENDTC, AEMODIFY (if present), AELAT (if present) and AETOXGR or AESEV

Usage

```
check_ae_dup(AE)
```

Arguments

AE	AE SDTM dataset with variables USUBJID, AETERM, AEDECOD, AESTDTC, AEENDTC, and AETOXGR or AESEV

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Edgar Manukyan

Examples

```
AE <- data.frame(USUBJID = c(1), AESTDTC = c("2020-01-01","2020-01-01","2020-02-01","2020-03-01"),
                 AEENDTC = rep("2020-02-01",4), AEDECOD = letters[c(1,1:3)],
                 AETERM = letters[c(1,1:3)], AETOXGR = c(1,1:3),
                 AESPID="FORMNAME-R:5/L:5XXXX",
                 stringsAsFactors=FALSE)

check_ae_dup(AE)
```

check_ae_fatal *Check for death variable consistency when AEOUT=="FATAL"*

Description

This check looks for consistency in AESDTH, AEDTHDTC, and AETOXGR (if applicable) when AEOUT is 'FATAL'. Note, this check expects AE grade/severity variables to be populated for either all records or none. In a case where both AETOXGR and AESEV exist and some records are supposed to have AETOXGR populated while others have AESEV (ie the two variables are mutually exclusive) then this check will likely return false positives.

Usage

```
check_ae_fatal(AE, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AEDECOD, AESTDTC, AEDTHDTC, AEOUT, AESDTH
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Aldrich Salva

Examples

```
# AETOXGR, no AESEV

AE <- data.frame(
 USUBJID = 1:5,
 AESTDTC = "01JAN2017",
 AEDECOD = c("AE1","AE2","AE3","AE4","AE5"),
 AEOUT = "FATAL",
 AEDTHDTC = c("01FEB2017",NA,"02FEB2017","03FEB2017",NA),
 AESDTH = c("Y","Y","N","Y",NA),
 AETOXGR = c("5","5","5",NA,NA),
 AESPID = "FORMNAME-R:12/L:2XXXX",
 stringsAsFactors = FALSE
)

check_ae_fatal(AE)
check_ae_fatal(AE,preproc=roche_derive_rave_row)
```

```
AE$AETOXGR <- NULL
check_ae_fatal(AE)

AE$AEDECOD <- NULL
check_ae_fatal(AE)


# AESEV, no AETOXGR

 AE <- data.frame(
 USUBJID = 1:5,
 AESTDTC = "01JAN2017",
 AEDECOD = c("AE1","AE2","AE3","AE4","AE5"),
 AEOUT = "FATAL",
 AEDTHDTC = c("01FEB2017","02FEB2017","03FEB2017","04FEB2017",NA),
 AESDTH = c("Y","Y","N","Y",NA),
 AESEV = c("SEVERE","MILD","SEVERE",NA,NA),
 AESPID = "FORMNAME-R:12/L:2XXXX",
 stringsAsFactors = FALSE
)

check_ae_fatal(AE)
check_ae_fatal(AE,preproc=roche_derive_rave_row)

AE$AESEV <- NULL
check_ae_fatal(AE)

# Both AESEV and AETOXGR have non-missing values

AE <- data.frame(
 USUBJID = 1:7,
 AESTDTC = "01JAN2017",
 AEDECOD = c("AE1","AE2","AE3","AE4","AE5","AE6","AE7"),
 AEOUT = "FATAL",
 AEDTHDTC = c("01FEB2017",NA,"02FEB2017","03FEB2017",NA,"04FEB2017","05FEB2017"),
 AESDTH = c("Y","Y","N","Y",NA,"Y","Y"),
 AESEV = c("SEVERE","MILD","SEVERE",NA,NA,"MILD","SEVERE"),
 AETOXGR = c("5","5","5",NA,NA,"1","5"),
 AESPID = "FORMNAME-R:12/L:2XXXX",
 stringsAsFactors = FALSE
)

check_ae_fatal(AE)
check_ae_fatal(AE,preproc=roche_derive_rave_row)


# Neither AESEV or AETOXGR

AE <- data.frame(
 USUBJID = 1:5,
 AESTDTC = "01JAN2017",
 AEDECOD = c("AE1","AE2","AE3","AE4","AE5"),
```

```
 AEOUT = "FATAL",
 AEDTHDTC = c("01FEB2017",NA,"02FEB2017","03FEB2017",NA),
 AESDTH = c("Y","Y","N","Y",NA),
 AESPID = "FORMNAME-R:12/L:2XXXX",
 stringsAsFactors = FALSE
)

check_ae_fatal(AE)

# AETOXGR exists but unmapped AESEV

 AE <- data.frame(
 USUBJID = 1:5,
 AESTDTC = "01JAN2017",
 AEDECOD = c("AE1","AE2","AE3","AE4","AE5"),
 AEOUT = "FATAL",
 AEDTHDTC = c("01FEB2017",NA,"02FEB2017","03FEB2017",NA),
 AESDTH = c("Y","Y","N","Y",NA),
 AESEV = rep(NA,5),
 AETOXGR = c("5","5","5",NA,NA),
 AESPID = "FORMNAME-R:12/L:2XXXX",
 stringsAsFactors = FALSE
)

check_ae_fatal(AE)
check_ae_fatal(AE,preproc=roche_derive_rave_row)

# AETOXGR and AESEV exist, by both are unmapped

 AE <- data.frame(
 USUBJID = 1:5,
 AESTDTC = "01JAN2017",
 AEDECOD = c("AE1","AE2","AE3","AE4","AE5"),
 AEOUT = "FATAL",
 AEDTHDTC = c("01FEB2017",NA,"02FEB2017","03FEB2017",NA),
 AESDTH = c("Y","Y","N","Y",NA),
 AESEV = NA,
 AETOXGR = NA,
 AESPID = "FORMNAME-R:12/L:2XXXX",
 stringsAsFactors = FALSE
)

check_ae_fatal(AE)
check_ae_fatal(AE,preproc=roche_derive_rave_row)
```

---

check_ae_withdr_ds_discon

Check if an AE leading to drug being withdrawn is reflected in DS

Description

This checks that if there is an AE with AEACN="DRUG WITHDRAWN" then there should be a treatment discontinuation record indicated by DS.DSSCAT

Usage

```
check_ae_withdr_ds_discon(AE, DS, TS, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AEACN
DS	Disposition SDTM dataset with variables USUBJID, DSCAT, DSSCAT
TS	Trial Summary SDTM dataset with variables TSPARMCD, TSVAL
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Yuliia Bahatska

Examples

```
AE <- data.frame(
 USUBJID = 1:6,
 AEACN = c("DRUG WITHDRAWN",NA,NA,NA,NA,NA),
 AETOXGR = c(NA,NA,NA,NA,"5",NA),
 AEDECOD=c("NAUSEA","HEADACHE"),
 AESPID = "FORMNAME-R:5/L:5XXXX"
)
DS <- data.frame(
 USUBJID = 1:3,
 DSCAT="DISPOSITION EVENT",
 DSSCAT="STUDY TREATMENT",
 DSDECOD=c("COMPLETED","ADVERSE EVENT","DEATH")
)

 TS <- data.frame(
 TSPARMCD="TRT",
 TSVAL="CHECK"
)

check_ae_withdr_ds_discon(AE,DS,TS)
check_ae_withdr_ds_discon(AE,DS,TS,preproc=roche_derive_rave_row)

DS$DSSCAT = NULL
```

```
check_ae_withdr_ds_discon(AE,DS,TS)
```

`check_ce_missing_month` *Check for clinical events dates with year and day known but month unknown*

Description

Check for missing month when clinical events dates (CESTDTC, CEENDTC, CEDTC) have a known year and day

Usage

```
check_ce_missing_month(CE, preproc = identity, ...)
```

Arguments

`CE`	Clinical Events SDTM dataset with variables USUBJID, CETERM, and at least one of the following date variables: CESTDTC, CEENDTC, CEDTC
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Ryan Marinelli

Examples

```
CE <- data.frame(
 USUBJID = c(1, 2, 3, 4),
 CETERM = c("Headache", "Nausea", "Dizziness", "Fever"),
 CESTDTC = c("2023---01", "2023-01-15", "2023-02-01", "2023-02-10"),
 CEENDTC = c("2023-01-02", "2023---01", "2023-02-02", "2023-02-12"),
 CEDTC = c("2023--01", "", "", ""),
 CESEV = c("Mild", "Moderate", "Mild", "Severe"),
 CESPID = "FORMNAME-R:13/L:13XXXX",
 stringsAsFactors=FALSE
 )

check_ce_missing_month(CE)
check_ce_missing_month(CE,preproc=roche_derive_rave_row)

CE <- data.frame(
```

```
 USUBJID = c(1, 2, 3, 4),
 CETERM = c("Headache", "Nausea", "Dizziness", "Fever"),
 CESTDTC = c("2023-01-01", "2023-01-15", "2023-02-01", "2023-02-10"),
 CEENDTC = c("2023-01-02", "2023-01-16", "2023-02-02", "2023-02-12"),
 CEENDTC = "",
 CESEV = c("Mild", "Moderate", "Mild", "Severe"),
 CESPID = "FORMNAME-R:13/L:13XXXX",
 stringsAsFactors=FALSE
 )

check_ce_missing_month(CE)

CE$CETERM = NULL

check_ce_missing_month(CE)
```

`check_cm_cmdecod` *Check for missing CMDECOD values*

Description

This check looks for missing CMDECOD values

Usage

```
check_cm_cmdecod(CM, preproc = identity, ...)
```

Arguments

`CM`	Concomitant Medications SDTM dataset with variables USUBJID, CMTRT, CMDECOD
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Lei Zhao, Stella Banjo (HackR 2021)

Examples

```
CM <- data.frame(
 USUBJID = 1:5,
 DOMAIN = rep("CM", 5),
 CMTRT = rep("DRUG TERM", 5),
 CMDECOD = rep("CODED DRUG TERM", 5),
 CMSTDTC = 1:5,
 CMENDTC = 1:5,
 CMCAT = "CONCOMITANT MEDICATIONS",
 CMSPID = c("FORMNAME-R:13/L:13XXXX",
             "FORMNAME-R:16/L:16XXXX",
             "FORMNAME-R:2/L:2XXXX",
             "FORMNAME-R:19/L:19XXXX",
             "FORMNAME-R:5/L:5XXXX"),
 stringsAsFactors=FALSE
)

check_cm_cmdecod(CM)

CM$CMDECOD[1] = NA
CM$CMDECOD[2] = "NA"
CM$CMDECOD[3:5] = ""
check_cm_cmdecod(CM)
check_cm_cmdecod(CM,preproc=roche_derive_rave_row)

CM$CMDECOD <- NULL
check_cm_cmdecod(CM)
```

---

check_cm_cmindc *Check for concomitant medication indication with text string "PROPHYL" when not given for prophylaxis*

Description

This check looks for patients with text string "PROPHYL" in CMINDC when CMPROPH is not checked as "Y" in studies with given for prophylaxis variable (CMPROPH)

Usage

```
check_cm_cmindc(CM, preproc = identity, ...)
```

Arguments

CM	Concomitant Medication SDTM dataset with variables USUBJID, CMTRT, CMSTDTC, CMINDC, CMPROPH, CMSPID (optional)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach, Stella Banjo (HackR 2021)

Examples

```
CM <- data.frame(
 USUBJID = c(rep(1,3),rep(2,3),rep(3,3)),
 CMTRT = letters[1:9],
 CMSTDTC = rep("2017-01-01",9),
 CMINDC = c(rep("INDICATION 1",2), rep("indication 2",2),
            rep("Prophylaxis",2),rep("PROPHYLACTIC",2),"PROPHYLAXIS FOR XYZ"),
 CMPROPH = c(rep("Y",3),rep(NA,2),rep("",2),"NA","."),
 CMSPID = "/F:XXX-D:12345-R:123",
 stringsAsFactors=FALSE
)

check_cm_cmindc(CM)
check_cm_cmindc(CM,preproc=roche_derive_rave_row)

CM$CMPROPH[7] = "Y"
check_cm_cmindc(CM)

CM$CMSPID = NULL
check_cm_cmindc(CM,preproc=roche_derive_rave_row)

CM$CMPROPH = NULL
check_cm_cmindc(CM)
```

---

check_cm_cmlat *Check if ocular concomitant medication has laterality missing or laterality field is populated but route is not eye-related.*

Description

This check assesses CMCAT = "CONCOMITANT MEDICATIONS" and flags potential ocular records with missing/inconsistent route and laterality: for eye-related CMROUTE ('INTRAVITREAL', 'OPHTHALMIC', etc.), CMLAT is not populated -or- CMROUTE is not eye-related (i.e., not INTRAVITREAL, OPHTHALMIC, TOPICAL, etc.) but CMLAT is LEFT/RIGHT/BILATERAL.

Usage

```
check_cm_cmlat(CM, preproc = identity, ...)
```

Arguments

CM	Concomitant Medications Dataset for Ophtho Study with variables USUBJID, CMCAT, CMLAT, CMDECOD, CMTRT, CMROUTE, CMSPID (if Present), CMSTDTC (if Present)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah (HackR 2021 Team Eye)

See Also

Other OPHTH: check_ae_aelat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_early_tot(), check_oe_bcva_4m_late_early_tot(), check_oe_bcva_4m_vs_1m_req(), check_oe_bcva_tot_mismatch(), check_oe_sc_lat_count_fingers(), check_pr_prlat(), check_sc_dm_eligcrit(), check_sc_dm_seyeselc()

Examples

```
CM <- data.frame(
   USUBJID = 1:7,
   CMCAT = "CONCOMITANT MEDICATIONS",
   CMSTDTC = 1:7,
   CMLAT   = c("Left", "","Bilateral", "", "", "LEFT", ""),
   CMTRT  = c("A", "B", "A", "B", "A", "A", "B"),
   CMDECOD = c("A", "B", "A", "B", "A", "A", "B"),
   CMROUTE = c("","OPHTHALMIC","INTRAVITREAL","INTRAVITREAL",
               "opHTHALMIC", "INTRaOCULAr", "INTRaOCULAr"),
   CMSPID  = "FORMNAME-R:13/L:13XXXX",
   stringsAsFactors = FALSE)
check_cm_cmlat(CM,preproc=roche_derive_rave_row)

CM <- data.frame(
   USUBJID = 1:5,
   CMCAT = rep("CONCOMITANT MEDICATIONS",5),
   CMSTDTC = 1:5,
   CMLAT   = c("Left", "LEFT","Bilateral",
               "RIGHT", "RIgHT"),
   CMTRT  = c("A", "B", "A", "B", "A"),
   CMDECOD = c("A", "B", "A", "B", "A"),
   CMROUTE = c("","OPHTHALMIC","INTRAVITREAL",
               "INTRaOCULAr", "opHTHALMIC"),
   stringsAsFactors = FALSE)
check_cm_cmlat(CM)

CM <- data.frame(
   USUBJID = 1:5,
```

```
   CMCAT = "CONCOMITANT MEDICATIONS",
   CMSTDTC = 1:5,
   CMLAT   = c("Left", "LEFT","Bilateral", "RIGHT", "RIgHT"),
   CMTRT  = c("A", "B", "A", "B", "A"),
   CMDECOD = c("A", "B", "A", "B", "A"),
   #CMROUTE = c("","OPHTHALMIC","INTRAVITREAL","INTRaOCULAr", "opHTHALMIC"),
   stringsAsFactors = FALSE)
check_cm_cmlat(CM)
```

`check_cm_cmlat_prior_ocular` *Check if ocular concomitant medication has laterality missing for specific "PRIOR OCULAR THERAPIES AND TREATMENTS" (or similar names) CRF page.*

Description

This check assesses ocular CMCAT records and flags records with missing/inconsistent laterality

Usage

```
check_cm_cmlat_prior_ocular(CM, preproc = identity, ...)
```

Arguments

`CM`	Concomitant Medications Dataset for Ophtha Study with variables USUBJID, CMCAT, CMLAT, CMTRT, CMSPID (if Present), CMSTDTC (if Present), CMLOC (if Present), CMINDC (if Present), CMDOSFRM (if Present)
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Tim Barnett (HackR 2021 Team Eye) (copied from check_cm_cmlat)

See Also

Other OPHTH: `check_ae_aelat()`, `check_cm_cmlat()`, `check_oe_bcva_1m_late_early_tot()`, `check_oe_bcva_4m_late_early_tot()`, `check_oe_bcva_4m_vs_1m_req()`, `check_oe_bcva_tot_mismatch()`, `check_oe_sc_lat_count_fingers()`, `check_pr_prlat()`, `check_sc_dm_eligcrit()`, `check_sc_dm_seyeselc()`

Examples

```
CM <- data.frame(
   USUBJID = 1:5,
   CMCAT = "PRIOR OCULAR THERAPIES AND TREATMENTS",
   CMSTDTC = 1:5,
   CMLAT   = c("Left", "","Bilateral", "", ""),
   CMTRT   = c("A", "B", "A", "B", "A"),
   CMDECOD = c("A", "B", "A", "B", "A"),
   CMROUTE = c("","OPHTHALMIC","INTRAVITREAL","INTRAVITREAL", "opHTHALMIC"),
   CMSPID  = "FORMNAME-R:13/L:13XXXX",
   stringsAsFactors = FALSE)
check_cm_cmlat_prior_ocular(CM,preproc=roche_derive_rave_row)

CM <- data.frame(
   USUBJID = 1:5,
   CMCAT = "Prior Ocular Therapies/Treatments",
   CMSTDTC = 1:5,
   CMLAT   = c("", "LEFT","Bilateral", "", "RIgHT"),
   CMTRT  = c("A", "B", "A", "B", "A"),
   CMDECOD = c("A", "B", "A", "B", "A"),
   #CMROUTE = c("","OPHTHALMIC","INTRAVITREAL","INTRAVITREAL", "opHTHALMIC"),
   stringsAsFactors = FALSE)
check_cm_cmlat_prior_ocular(CM)

CM <- data.frame(
   USUBJID = 1:5,
   CMCAT = "CONCOMITANT MEDICATIONS",
   CMSTDTC = 1:5,
   CMLAT   = c("Left", "LEFT","Bilateral", "RIGHT", "RIgHT"),
   CMTRT  = c("A", "B", "A", "B", "A"),
   CMDECOD = c("A", "B", "A", "B", "A"),
   CMROUTE = c("","OPHTHALMIC","INTRAVITREAL","INTRAVITREAL", "opHTHALMIC"),
   stringsAsFactors = FALSE)
check_cm_cmlat_prior_ocular(CM)

CM <- data.frame(
   USUBJID = 1:5,
   CMCAT = "CONCOMITANT MEDICATIONS",
   CMSTDTC = 1:5,
   CMLAT   = c("Left", "LEFT","Bilateral", "RIGHT", "RIgHT"),
   CMTRT  = c("A", "B", "A", "B", "A"),
   CMDECOD = c("A", "B", "A", "B", "A"),
   #CMROUTE = c("","OPHTHALMIC","INTRAVITREAL","INTRAVITREAL", "opHTHALMIC"),
   stringsAsFactors = FALSE)
check_cm_cmlat_prior_ocular(CM)

CM <- data.frame(
   USUBJID = 1:5,
  CMCAT = c(rep("Prior Ocular Therapies/Treatments",3), rep("Non-Ocular Therapies/Treatments",2)),
   CMSTDTC = 1:5,
   CMLAT   = c("", "LEFT","Bilateral", "", ""),
   CMTRT  = c("A", "B", "A", "B", "A"),
```

```
   CMDECOD = c("A", "B", "A", "B", "A"),
   #CMROUTE = c("","OPHTHALMIC","INTRAVITREAL","ORAL", "ORAL"),
   stringsAsFactors = FALSE)
check_cm_cmlat_prior_ocular(CM)
```

`check_cm_missing_month` *Check for conmed dates with year and day known but month unknown*

Description

Check for missing month when conmed start (CMSTDTC) or end dates (CMENDTC) have known year and day

Usage

```
check_cm_missing_month(CM, preproc = identity, ...)
```

Arguments

`CM`	Concomitant Medications SDTM dataset with variables USUBJID, CMTRT, CMSTDTC, CMENDTC
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Chandra Mannem

Examples

```
CM <- data.frame(
 USUBJID = 1:3,
 CMTRT = c("CM1","CM2","CM3"),
 CMSTDTC = c("2017-01-01","2017---01","2017-01-02"),
 CMENDTC = c("2017-02-01","2017-03-01","2017---01"),
 CMSPID = "/F:XXX-D:12345-R:123",
 stringsAsFactors=FALSE
)

check_cm_missing_month(CM)
check_cm_missing_month(CM,preproc=roche_derive_rave_row)
```

```
CM$CMSTDTC = NULL

check_cm_missing_month(CM)
```

---

check_dd_ae_aedthdtc_ds_dsstdtc *Check if death date is the same in AE and DS domains*

Description

This check compares death date in AE AEDTHDT with death date in DS DSSTDTC. It is expected that they are the same.

Usage

```
check_dd_ae_aedthdtc_ds_dsstdtc(AE, DS, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID and AEDTHDTC
DS	Disposition SDTM dataset with variables USUBJID, DSDECOD, DSSTDTC
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Hiral Raval

Examples

```
AE <- data.frame(
 STUDYID = rep(1, 3),
 USUBJID = 1:3,
 AEDTHDTC = c("2020-01-01","2020-01-02","2020-01-03"),
 AESPID = "FORMNAME-R:19/L:19XXXX"
)

DS <- data.frame(
 STUDYID = rep(1, 3),
 USUBJID = 1:3,
 DSDECOD = rep("DEATH", 3),
 DSSTDTC = c("2020-01-01","2020-01-02","2020-01-03"),
```

```
 DSSPID = "XXX-R:0",
 stringsAsFactors = FALSE
)

# no case
check_dd_ae_aedthdtc_ds_dsstdtc(AE, DS)

# 1 case
DS[3, "DSSTDTC"] <- "2000-01-01"
check_dd_ae_aedthdtc_ds_dsstdtc(AE, DS, preproc=roche_derive_rave_row)

# check for non existence of vars
DS$DSDECOD <- NULL
DS$DSSTDTC <- NULL
check_dd_ae_aedthdtc_ds_dsstdtc(AE, DS)
```

`check_dd_ae_aeout_aedthdtc` *Check if there is a death date and AEOUT='FATAL' agreement*

Description

This check looks for AE death dates if AEOUT='FATAL' and for the reverse, i.e if there is an AE death date, then AEOUT should have the value "FATAL".

Usage

```
check_dd_ae_aeout_aedthdtc(AE, preproc = identity, ...)
```

Arguments

`AE`	Adverse Events SDTM dataset with variables USUBJID, AEDTHDTC, AEDECOD, AESTDTC and AEOUT
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Joel Laxamana

Examples

```
AE <- data.frame(
 USUBJID = 1:3,
 AEDTHDTC = c("2020-01-01","2020-01-02","2020-01-03"),
 AEDECOD = 1:3,
 AESTDTC = 1:3,
 AEOUT = rep("FATAL", 3),
 AESPID = "FORMNAME-R:19/L:19XXXX",
 stringsAsFactors = FALSE
)

# pass
check_dd_ae_aeout_aedthdtc(AE)

# fail - 1 case (AEDTHDTC not populated but AEOUT == FATAL)
AE1 <- AE
AE1[3, "AEDTHDTC"] <- NA
check_dd_ae_aeout_aedthdtc(AE1)
check_dd_ae_aeout_aedthdtc(AE1,preproc=roche_derive_rave_row)

# pass -- even though AEDTHDTC populated
AE2 <- AE
AE2[1, "AEOUT"] <- NA
check_dd_ae_aeout_aedthdtc(AE2)
check_dd_ae_aeout_aedthdtc(AE2,preproc=roche_derive_rave_row)

# 2 cases
AE[3, "AEDTHDTC"] <- NA
AE[1, "AEOUT"] <- NA
check_dd_ae_aeout_aedthdtc(AE)
check_dd_ae_aeout_aedthdtc(AE,preproc=roche_derive_rave_row)

# 2 cases
AE[1, "AEOUT"] <- 'NOT RECOVERED/NOT RESOLVED'
check_dd_ae_aeout_aedthdtc(AE)
check_dd_ae_aeout_aedthdtc(AE,preproc=roche_derive_rave_row)

# non-critical variable missing
AE$AESPID <- NULL
check_dd_ae_aeout_aedthdtc(AE)
check_dd_ae_aeout_aedthdtc(AE,preproc=roche_derive_rave_row)

# critical variables are missing
AE$AEDTHDTC <- NULL
AE$USUBJID <- NULL
check_dd_ae_aeout_aedthdtc(AE)
check_dd_ae_aeout_aedthdtc(AE,preproc=roche_derive_rave_row)
```

`check_dd_death_date` *Check if patient with Death due to AE also has Death record in DS*

Description

Flag if patient has a Death in AE (i.e. AE record with non-missing AE.AEDTHDTC) but no Death in DS (i.e. record where DS.DSDECOD=DEATH and DS.DSTERM contains 'DEATH' and does not contain 'PROGRESSIVE DISEASE' or 'DISEASE RELAPSE' (so we can pick up records where DSTERM in 'DEATH','DEATH DUE TO ...' and exclude 'DEATH DUE TO PROGRESSIVE DISEASE', 'DEATH DUE TO DISEASE RELAPSE')

Usage

```
check_dd_death_date(AE, DS, preproc = identity, ...)
```

Arguments

AE	Adverse Events SDTM dataset with USUBJID, AEDTHDTC, AESPID (optional)
DS	Disposition SDTM dataset with USUBJID, DSDECOD, DSTERM
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Edgar Manukyan, N Springfield updated on 14SEP2020

Examples

```
AE <- data.frame(
 USUBJID = 1:5,
 AEDTHDTC = c("2018-01-01", "2018-01-02", "2018-01-03","2018-01-04", ""),
 AESPID="FORMNAME-R:13/L:13XXXX",
 stringsAsFactors = FALSE
)

DS <- data.frame(
 USUBJID = c(1,1,2,3,3,4),
 DSTERM=c("DEATH","RANDOM THING","ADVERSE EVENT",
          "DEATH DUE TO PROGRESSIVE DISEASE","ADVERSE EVENT",
          "DEATH DUE TO ABC"),
 DSDECOD=c("DEATH","ADVERSE EVENT","DEATH", "DEATH","OTHER", "DEATH"),
 stringsAsFactors=FALSE
)

check_dd_death_date(AE,DS)
check_dd_death_date(AE,DS,preproc=roche_derive_rave_row)
```

check_dm_actarm_arm *Check DM where ARM is not equal to ACTARM*

Description

This check looks for DM entries where ARM is not equal to ACTARM

Usage

```
check_dm_actarm_arm(DM)
```

Arguments

DM	Demographics SDTM dataset with variables USUBJID, ARM, and ACTARM

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Ying Yuen

Examples

```
DM <- data.frame(USUBJID = 1:5,
                 ARM = c(letters[1:3], letters[5:6]),
                 ACTARM = letters[1:5],
                 stringsAsFactors = FALSE)

check_dm_actarm_arm(DM)
```

check_dm_ae_ds_death *Check if death reported in DM then death indicator also present in DS or AE*

Description

This checks that when death is indicated in DM with either of DTHFL or DTHDTC then there should be death indicated in either AE or DS.

Usage

```
check_dm_ae_ds_death(DM, DS, AE)
```

Arguments

DM	Demographics SDTM dataset with variables USUBJID, DTHFL, DTHDTC
DS	Disposition SDTM dataset with variables USUBJID, DSDECOD, DSSTDTC
AE	Adverse Events SDTM dataset with variables USUBJID, AEDTHDTC, AESDTH, AEOUT

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
AE <- data.frame(
 STUDYID = 1,
 USUBJID = 1:3,
 AEDTHDTC = c(NA,1,NA),
 AESDTH = c(NA,"Y",NA),
 AEOUT = c(NA,"FATAL",NA),
 AETOXGR = c(NA,"5",NA)
)

DS <- data.frame(
 STUDYID = 1,
 USUBJID = 1:3,
 DSDECOD = c(NA,"DEATH",NA),
 DSSTDTC = c(NA,"DSDATE",NA)
)

DM <- data.frame(
 STUDYID = 1,
 USUBJID = 1:3,
 DTHFL=c(NA,"Y","Y"),
 DTHDTC = c(NA,"DMDATE","DMDATE")
 )

check_dm_ae_ds_death(DM,DS,AE)

DS$DSDECOD = NULL

check_dm_ae_ds_death(DM,DS,AE)
```

check_dm_age_missing *Check for patients with suspicious age values*

Description

Check for patients with missing AGE, AGE<18 or AGE>90 in DM

Usage

```
check_dm_age_missing(DM)
```

Arguments

DM	Demographics SDTM dataset with variables USUBJID,AGE

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Nina Qi

Examples

```
DM <- data.frame(
 USUBJID = 1:10,
 AGE = c(50,60,17,99,NA,33,500,40,22,NA)
)

check_dm_age_missing(DM)

DM$AGE = NULL

check_dm_age_missing(DM)
```

check_dm_armcd *Check for missing ARM or ARMCD values in DM*

Description

This check looks for missing ARM or ARMCD values

Usage

```
check_dm_armcd(DM)
```

Arguments

DM	Demographics SDTM dataset with variables USUBJID, ARM, ARMCD

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Rena Wang

Examples

```
DM <- data.frame(
 USUBJID = 1:3,
 ARM = 1:3,
 ARMCD = 1:3
)

check_dm_armcd(DM)

DM$ARMCD[1] <- NA
check_dm_armcd(DM)

DM$ARM[2] <- NA
check_dm_armcd(DM)

DM$ARMCD <- NULL
check_dm_armcd(DM)
```

---

check_dm_dthfl_dthdtc *Check that when DM.DTHFL is Y, DM.DTHDTC does not have a missing value, and vice versa*

Description

This check is bi-directional for consistency of DM.DTHFL and DM.DTHDTC and returns a data frame. Note there is a possible valid scenario for this issue if death date is truly unknown

Usage

```
check_dm_dthfl_dthdtc(DM)
```

Arguments

DM	Demographics SDTM dataset with variables USUBJID,DTHFL,DTHDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Ross Farrugia

Examples

```
DM <- data.frame(
 USUBJID = 1:7,
 DTHFL = 1:7,
 DTHDTC = 1:7
)

DM$DTHFL[1] = ""
DM$DTHDTC[1] = "2020-01-01"
DM$DTHFL[2] = "N"
DM$DTHDTC[2] = "2020-01-01"
DM$DTHFL[3] = "Y"
DM$DTHDTC[3] = "2020-01-01"
DM$DTHFL[4] = "Y"
DM$DTHDTC[4] = ""
DM$DTHFL[5] = "N"
DM$DTHDTC[5] = ""
DM$DTHFL[6] = "Y"
DM$DTHDTC[6] = "2020"
DM$DTHFL[7] = ""
DM$DTHDTC[7] = ""
check_dm_dthfl_dthdtc(DM)

DM$DTHFL <- NULL
DM$DTHDTC <- NULL
check_dm_dthfl_dthdtc(DM)
```

---

check_dm_usubjid_ae_usubjid

Check patients in the DM dataset who do not have records in the AE dataset

---

Description

This check looks for patients in the DM dataset who do not have records in the AE dataset, and obtains first treatment start date and earliest death date for these patients

Usage

```
check_dm_usubjid_ae_usubjid(DM, AE, DS, EX)
```

Arguments

DM	Demographics SDTM dataset with variable USUBJID
AE	Adverse Events SDTM dataset with variable USUBJID
DS	Disposition SDTM dataset with variables USUBJID, DSSTDTC, DSDECOD
EX	Exposure SDTM dataset with variables USUBJID, EXDOSE, EXSTDTC, EXTRT

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Vani Nimbal

Examples

```
USUBJID<- c(1:10)
DM=data.frame(USUBJID)
AE=data.frame(USUBJID)
AE$USUBJID[3]=NA
AE$USUBJID[8]=NA
AE$USUBJID[10]=NA

EX <- data.frame(
USUBJID = c(1:8, 6, 8, 10, 10, 10, 10),
EXOCCUR = rep("Y", times=14),
EXDOSE = rep(c(1,2), times=7),
EXSTDTC = c(rep("2012-01-01", 10),"2012-02-04","2012-02-04", "", "2012-02-07"),
EXTRT = "GDC",
stringsAsFactors=FALSE
)

DS <- data.frame(
USUBJID = c(2,8,8),
DSDECOD = rep("DEATH", times=3),
DSSTDTC = c("2012-12-01", NA, "2013-07-01"),
stringsAsFactors=FALSE
)
check_dm_usubjid_ae_usubjid(DM, AE, DS, EX)

EX$EXOCCUR[3]="N"

check_dm_usubjid_ae_usubjid(DM, AE, DS, EX)

EX$EXOCCUR=NULL

check_dm_usubjid_ae_usubjid(DM, AE, DS, EX)
```

check_dm_usubjid_dup *Check duplicate patient records in DM based on USUBJID*

Description

This check looks for duplicate patient demographics records in DM

Usage

```
check_dm_usubjid_dup(DM)
```

Arguments

DM	Demographics SDTM dataset with variable USUBJID

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Madeleine Ma, Stella Banjo (HackR 2021)

Examples

```
## duplicates and same patient number across sites for 3-part USUBJID
DM <- data.frame(USUBJID = c("GO12345-00000-1000",
                             "GO12345-11111-1000",
                             "GO12345-00000-1000",
                             "GO12345-00000-1001"),
      stringsAsFactors = FALSE)

check_dm_usubjid_dup(DM)


## no duplicate IDs in the dataframe for 3-part USUBJID
DM2 <- data.frame(USUBJID = c("GO12345-00000-1000",
                              "GO12345-11111-1001",
                              "GO12345-11111-1002"),
             stringAsFactors = FALSE)

check_dm_usubjid_dup(DM2)


## duplicates for 2-part USUBJID
DM3 <- data.frame(USUBJID = c("GO12345-1000",
                              "GO12345-1000"),
            stringAsFactors = FALSE)

check_dm_usubjid_dup(DM3)
```

```
##  no duplicate IDs in the dataframe for 2-part USUBJID
DM4 <- data.frame(USUBJID = c("GO12345-1000",
                              "GO12345-1001",
                              "GO12345-1002"),
            stringAsFactors = FALSE)

check_dm_usubjid_dup(DM4)

## dataframe with one or two additional variables, if there is variation across other variables
DM5 <- data.frame(USUBJID = c("GO12345-1000",
                              "GO12345-1000"),
                  SEX = c("M", "F"),
                  AGE = c(18, 60),
         stringAsFactors = FALSE)

check_dm_usubjid_dup(DM5)

## dataframe in which USUBJID is not present
DM6 <- data.frame(
         STUDYID = c("GO12345"),
         SEX = c("M"),
         AGE = c(72),
     stringAsFactors = FALSE)

check_dm_usubjid_dup(DM6)
```

`check_ds_ae_discon` *Check for treatment discontinuation consistency between DS and AE*

Description

This check looks for consistency when DS.DSSPID=DISCTX* then there should be AE.AEACN*=DRUG WITHDRAWN

Usage

```
check_ds_ae_discon(DS, AE)
```

Arguments

DS	Disposition SDTM dataset with variables USUBJID, DSSPID, DSCAT, DSDECOD, DSSTDTC
AE	Adverse Events SDTM dataset with variables USUBJID, AEDECOD, AESTDTC, AEACN*

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sarwan Singh

Examples

```
AE <- data.frame(
 USUBJID = 1:5,
 AESTDTC = "01JAN2017",
 AETERM  = c("AE1","AE2","AE3","AE4","AE5"),
 AEDECOD = c("AE1","AE2","AE3","AE4","AE5"),
 AEACN = c("DOSE REDUCED", "DOSE REDUCED", "DOSE NOT CHANGED",
 "DOSE NOT CHANGED", "NOT APPLICABLE"),
 stringsAsFactors = FALSE
)


DS <- data.frame(
 USUBJID = 1:5,
 DSSPID  = c('XXXDISCTXXXXX'),
 DSSTDTC = '01JAN2017',
 DSCAT   = rep("DISPOSITION EVENT", 5),
 DSSCAT   = rep("TX FORM", 5),
 DSDECOD = c("PHYSICIAN DECISION", "OTHER", "PHYSICIAN DECISION", "OTHER", "DEATH"),
 stringsAsFactors = FALSE
)

# no case
check_ds_ae_discon(DS, AE)

# 1 case
DS[3, "DSDECOD"] <- 'ADVERSE EVENT'
check_ds_ae_discon(DS, AE)

# mutliple AEACNx
AE <- data.frame(
 USUBJID = 1:5,
 AESTDTC = c("01JAN2017"),
 AETERM  = c("AE1","AE2","AE3","AE4","AE5"),
 AEDECOD = c("AE1","AE2","AE3","AE4","AE5"),
 AEACN   = rep("MULTIPLE", 5),
 AEACN1  = c("DOSE REDUCED", "DOSE NOT CHANGED", "DOSE NOT CHANGED",
 "DOSE NOT CHANGED", "NOT APPLICABLE"),
 stringsAsFactors = FALSE
)

check_ds_ae_discon(DS, AE)
```

check_ds_dsdecod_death *Check for study discontinuation record if death indicated*

Description

If a patient has a record where DS.DSDECOD == DEATH they should also have a Study Discon Record

Usage

```
check_ds_dsdecod_death(DS, preproc = identity, ...)
```

Arguments

DS	Disposition domain with variables USUBJID, DSDECOD, DSSCAT, and optional variables DSCAT, DSSTDTC, DSSPID
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach and Will Harris

Examples

```
DS <- data.frame(
 STUDYID = 1,
 USUBJID = 1:3,
 DSDECOD = c(NA,"DEATH",NA),
 DSSTDTC = c(NA,"DSDATE",NA),
 DSCAT = c('DISPOSITION EVENT', 'DISPOSITION EVENT', 'OTHER'),
 DSSCAT = c('STUDY COMPLETION/EARLY DISCONTINUATION',
            'TREATMENT DISCONTINUATION',
            'STUDY TREATMENT'),
 DSOTH = 1:3,
 DSSPID = "XXX-R:0",
 stringsAsFactors=FALSE
 )

check_ds_dsdecod_death(DS)
check_ds_dsdecod_death(DS,preproc=roche_derive_rave_row)

DS$DSSCAT[2] <- "STUDY COMPLETION/EARLY DISCONTINUATION"
```

```
check_ds_dsdecod_death(DS)

DS$DSDECOD = NULL
check_ds_dsdecod_death(DS)
```

`check_ds_dsdecod_dsstdtc` *Check DS with death record but no death date*

Description

This check looks for patients in DS who have a record indicating death but no corresponding record with death date in DS. For example, "Survival Follow Up" records often have no death dates, so for a data cut to be applied properly, you have to impute that missing death date from another record where its not missing (e.g. Study Discon form)

Usage

```
check_ds_dsdecod_dsstdtc(DS)
```

Arguments

DS	Disposition SDTMv dataset with variables USUBJID, DSDECOD, DSSCAT and DSSTDTC

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the test failed

Author(s)

Will Harris

Examples

```
DS <- data.frame(STUDYID = rep(1, 5),
                 USUBJID = c(1, 1, 1, 2, 3),
                 DSDECOD = c("DEATH", "DEATH", rep("", 3)),
                 DSSCAT = LETTERS[1:5],
                 DSSTDTC = c("", "2016-01-01", "", "", "2016-01-02"),
                 stringsAsFactors = FALSE)

check_ds_dsdecod_dsstdtc(DS)

DS$DSSTDTC[2] <- ""

check_ds_dsdecod_dsstdtc(DS)
```

`check_ds_dsscat` *Check for patients with more than one study discontinuation records*

Description

This check looks for patient who has more than one study discontinuation records

Usage

```
check_ds_dsscat(DS)
```

Arguments

DS	Disposition SDTM dataset with variables USUBJID,DSSCAT

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Madeleine Ma

Examples

```
DS <- data.frame(
 USUBJID = c(rep(1,3),rep(2,3),rep(3,3)),
 DSSCAT= rep(c("STUDY DISCONTINUATION", "ADVERSE EVENT", "PROTOCOL"),3),
 stringsAsFactors=FALSE
)
check_ds_dsscat(DS)

DS$DSSCAT[8] = "STUDY DISCONTINUATION"
check_ds_dsscat(DS)

DS$DSSCAT = NULL
check_ds_dsscat(DS)
```

check_ds_dsterm_death_due_to *Check missing cause of death information in DS*

Description

This check looks for DS.DSTERM values with missing death reason and returns a data frame (e.g. records where DSTERM = 'DEATH DUE TO')

Usage

```
check_ds_dsterm_death_due_to(DS)
```

Arguments

DS	Disposition SDTMv dataset with variables USUBJID, DSTERM, DSDECOD, DSDTC, DSSTDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
DS <- data.frame(
 STUDYID = 1,
 USUBJID = 1:4,
 DSTERM = c("DEATH DUE TO",
   "DEATH DUE TO ",
   "DEATH DUE TO ADVERSE EVENT",
   "DEATH DUE TO UNKNOWN"),
 DSDECOD = "DEATH",
 DSDTC = "2017-01-01",
 DSSTDTC = "2017-01-01",
 stringsAsFactors=FALSE
)

check_ds_dsterm_death_due_to(DS)

DS$DSDECOD <- NULL
check_ds_dsterm_death_due_to(DS)
```

check_ds_duplicate_randomization *Check for duplicate randomization records for a patient*

Description

Checks for duplicate subject IDs (USUBJID) in the DS domain when randomization is indicated

Usage

```
check_ds_duplicate_randomization(DS)
```

Arguments

DS	Disposition SDTM dataset with variables USUBJID, DSDECOD

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Madeleine Ma

Examples

```
DS <- data.frame(
 USUBJID = c("ID1","ID1","ID2","ID2","ID3","ID3"),
 DSDECOD = c("RANDOMIZATION","OTHER THING","RANDOMIZATION",
             "OTHER THING","RANDOMIZATION","RANDOMIZATION")
             , stringsAsFactors = FALSE
)

check_ds_duplicate_randomization(DS)

DS$DSDECOD <- NULL
check_ds_duplicate_randomization(DS)
```

check_ds_ex_after_discon *Check for patients who had Start/End date of treatment after study discontinuation date*

Description

Check for patients who had Start/End date of treatment after study discontinuation date in the DS and EX domains.

Usage

```
check_ds_ex_after_discon(DS, EX)
```

Arguments

DS	Disposition SDTM dataset with variables USUBJID, DSSCAT, DSCAT and DSSTDTC
EX	Exposure SDTM dataset with variables USUBJID, EXSTDTC, EXENDTC, EXTRT, EXDOSE and EXOCCUR (if available)

Value

Boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Saibah Chohan, Ashley Mao, Tina Cho (HackR 2021 Team STA-R)

Examples

```
DS <- data.frame(
 USUBJID = c(rep(1,2), rep(2,2)),
 DSSCAT= rep(c("STUDY COMPLETION/EARLY DISCONTINUATION", "ADVERSE EVENT"),2),
 DSCAT = rep(c("DISPOSITION EVENT", "OTHER"),2),
 DSSTDTC = c("2019-12-29", "2019-12-20", "2019-12-10", "2019-12-01"),
 stringsAsFactors = FALSE
 )

EX <- data.frame(
 USUBJID = c(rep(1,2), rep(2,2)),
 EXSTDTC = c("2019-12-20", "2019-12-28", "2019-12-26", "2019-12-27"),
 EXENDTC = c("2019-12-10", "2019-12-23", "2019-12-30", "2019-12-27"),
 EXTRT = c(rep("SOME DRUG", 2), rep("PLACEBO",2)),
 EXDOSE = c(10,10,0,0),
 stringsAsFactors = FALSE
 )

check_ds_ex_after_discon(DS, EX)
```

```
DS <- data.frame(
 USUBJID = c(rep(1,2), rep(2,2)),
 DSSCAT= rep(c("STUDY COMPLETION/EARLY DISCONTINUATION", "ADVERSE EVENT"),2),
 DSCAT = rep(c("DISPOSITION EVENT", "OTHER"),2),
 DSSTDTC = c("2019-12-29", "2019-12-20", "2019-12-10", "2019-12-01"),
 stringsAsFactors = FALSE
 )

EX <- data.frame(
 USUBJID = c(rep(1,2), rep(2,2)),
 EXSTDTC = c("2019-12-20", "2019-12-28", "2019-12-01", "2019-12-02"),
 EXENDTC = c("2019-12-10", "2019-12-23", "2020", "2020"),
 EXTRT = c(rep("SOME DRUG", 2), rep("PLACEBO",2)),
 EXDOSE = c(10,10,0,0),
 stringsAsFactors = FALSE
 )

check_ds_ex_after_discon(DS, EX)
```

`check_ds_multdeath_dsstdtc` *Check DS with multiple death records with death dates, where death dates do not match*

Description

This check looks for patients in DS who have multiple records indicating death, with non-missing mismatching death dates in DSSTDTC.

Usage

```
check_ds_multdeath_dsstdtc(DS, preproc = identity, ...)
```

Arguments

`DS`	Disposition SDTMv dataset with variables USUBJID, DSDECOD, and DSSTDTC
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the test failed

Author(s)

Kimberly Fernandes

Examples

```
DS_error1 <- data.frame(STUDYID = rep(1, 6),
  USUBJID = c(1, 1, 1, 2, 1,1),
  DSDECOD = c("DEATH", "DEATH", rep("", 2),"DEATH","DEATH"),
  DSSCAT = LETTERS[1:6],
  DSSTDTC = c("", "2016-01-01", "", "", "2016-01-02","2016-01-01"),
  stringsAsFactors = FALSE)

DS_error2 <- data.frame(STUDYID = rep(1, 6),
  USUBJID = c(1, 1, 1, 2, 1,1),
  DSDECOD = c("DEATH", "DEATH", rep("", 2),"DEATH","DEATH"),
  DSSCAT = LETTERS[1:6],
  DSSTDTC = c("", "2016-01", "", "", "2016-01-01","2016-01-01"),
  stringsAsFactors = FALSE)

 DS_noerror <- data.frame(STUDYID = rep(1, 6),
                          USUBJID = c(1, 1, 1, 2, 1,1),
                          DSDECOD = c("DEATH", "DEATH", rep("", 2),"DEATH","DEATH"),
                          DSSCAT = LETTERS[1:6],
                         DSSTDTC = c("", "2016-01-01", "", "", "2016-01-01","2016-01-01"),
                          stringsAsFactors = FALSE)

check_ds_multdeath_dsstdtc(DS_error1)
check_ds_multdeath_dsstdtc(DS_error2)
check_ds_multdeath_dsstdtc(DS_noerror)
```

---

check_ds_sc_strat *Check if randomized patients are missing stratification factor data*

Description

Check if Study is randomized (DS.DSDECOD == "RANDOMIZED" or "RANDOMIZATION"), and subject Characteristics Domain (SC) has no stratification factors reported.

Usage

```
check_ds_sc_strat(DS, SC)
```

Arguments

DS	Subject Dispostion Dataset with variable USUBJID, DSDECOD, DSSTDTC
SC	Subject Characteristics Dataset with variables USUBJID, SCTEST, SCTESTCD, SCCAT, SCORRES

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah

Examples

```
ds <- data.frame(USUBJID = c(1,2,2),
                 DSDECOD = c("RANDOMIZATION", "RANDOMIZED", "Randomized"),
                 DSSTDTC = c("2021-01-01", "2021-01-02", "2021-02-01"))
sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                 SCCAT = rep("STRATIFICATION", 6),
             SCTESTCD = c("STRAT 1", "STRAT 2", "STRAT 3", "STRAT 1", "STRAT 2", "STRAT 3"),
                 SCTEST   = c("Factor 1", "Factor 2", "Factor 3",
                              "Factor 1", "Factor 2", "Factor 3"),
                 SCORRES  = c("US", "Left", "Score > x", "RoW", "Right", "Score < x"),
                 stringsAsFactors = FALSE)

check_ds_sc_strat(ds, sc)

ds <- data.frame(USUBJID = c(1,2,2),
                 DSDECOD = c("RANDOMIZATION", "RANDOMIZED", "Randomized"),
                 DSSTDTC = c("2021-01-01", "2021-01-02", "2021-02-01"))
sc <- data.frame(USUBJID  = c(1,1,1),
                 SCCAT = rep("STRATIFICATION", 3),
                 SCTESTCD = c("STRAT 1", "STRAT 2", "STRAT 3"),
                 SCTEST   = c("Factor 1", "Factor 2", "Factor 3"),
                 SCORRES  = c("US", "Left", NA),
                 stringsAsFactors = FALSE)

check_ds_sc_strat(ds, sc)

ds <- data.frame(USUBJID = c(1,2),
                 DSDECOD = c("Open Label", "Open Label"),
                 DSSTDTC = c("2021-01-01", "2021-01-02"))
sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                 SCCAT = rep("No STRATIFICATION", 6),
             SCTESTCD = c("STRAT 1", "STRAT 2", "STRAT 3", "STRAT 1", "STRAT 2", "STRAT 3"),
                 SCTEST   = c("Factor 1", "Factor 2", "Factor 3",
                              "Factor 1", "Factor 2", "Factor 3"),
                 SCORRES  = c("US", "Left", NA, "RoW", "Right", "Score < x"),
                 stringsAsFactors = FALSE)

check_ds_sc_strat(ds, sc)
```

`check_dv_ae_aedecod_covid` *Check for consistency between DV and AE for COVID-19 events*

Description

If a patient has a DV record indicating COVID-19 then they should also have COVID-related AE where AE.AEDECOD matches covid.REFTERM.

Usage

```
check_dv_ae_aedecod_covid(
  AE,
  DV,
  covid_terms = c("COVID-19", "CORONAVIRUS POSITIVE")
)
```

Arguments

AE	Adverse Events SDTM dataset with variables USUBJID, AEDECOD
DV	Protocol Deviation SDTM dataset with variables USUBJID, DVREAS
covid_terms	A length >=1 vector of AE terms identifying COVID-19 (case does not matter)

Value

boolean value if check returns 0 obs, otherwise return subset dataframe.

Author(s)

Natalie Springfield

See Also

Other COVID: `check_ae_aeacn_ds_disctx_covid()`, `check_ae_aeacnoth_ds_stddisc_covid()`, `check_dv_covid()`

Examples

```
AE <- data.frame(
 USUBJID = 1:6,
 AEDECOD = c("pandemic", "covid-19","some AE","some AE","CORONAVIRUS POSITIVE","UNMAPPED")
)

DV <- data.frame(
 USUBJID = 1:6,
 DVREAS=c("SUSPECTED EPIDEMIC/PANDEMIC INFECTION",
          "UNKNOWN",
          "SUSPECTED EPIDEMIC/PANDEMIC INFECTION",
```

```
          "OTHER",
          "SUSPECTED EPIDEMIC/PANDEMIC INFECTION",
          "SUSPECTED EPIDEMIC/PANDEMIC INFECTION")
)

check_dv_ae_aedecod_covid(AE,DV)


# Pass specific covid terms

check_dv_ae_aedecod_covid(AE,DV,covid_terms=c("COVID-19", "CORONAVIRUS POSITIVE","PANDEMIC"))
```

`check_dv_covid` *Check for consistency in COVID-19 DV variables, DVREAS and DVEPRELI*

Description

This check looks for inconsistency between DVREAS and DVEPRELI. If DVREAS indicates a COVID-19 related deviation, then DVEPRELI should not be missing and vice versa. This check applies to studies using the Protocol Deviation Management System (PDMS).

Usage

```
check_dv_covid(DV)
```

Arguments

DV	Protocol Deviations SDTM dataset with variables USUBJID, DVREAS, DVEPRELI

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Mij Rahman

See Also

Other COVID: `check_ae_aeacn_ds_disctx_covid()`, `check_ae_aeacnoth_ds_stddisc_covid()`, `check_dv_ae_aedecod_covid()`

Examples

```
DV <- data.frame(
    USUBJID = 1:3,
    DVEPRELI = c("Y","N","Y"),
    DVREAS=c("EPIDEMIC/PANDEMIC INFECTION","EPIDEMIC/PANDEMIC INFECTION",""),
    stringsAsFactors=FALSE
)

check_dv_covid(DV)
```

---

check_ec_sc_lat *Check if Study Drug is not administered in the Study Eye*

Description

Check if Study Drug is not administered in the Study Eye. 1.> Subset Exposure dataset (EC) for only ocular Study Drug Administration records, and pass the check if there are none. If EC.ECCAT variable is available then remove records containing EC.ECCAT = "FELLOW". If EC.ECCAT variable is not available then include all records, assuming drug administration is collected for study eye only. 2.> Subset Subject Characteristics dataset (SC) for only Study Eye Selection 3.> Compare Exposure dataset laterality (EC.ECLAT) with Subject Characteristics dataset laterality (SC.SCORRES - OS = LEFT, OD = RIGHT) and report if there is any mismatch.

Usage

```
check_ec_sc_lat(EC, SC)
```

Arguments

EC	Subject Exposure Dataset with variables USUBJID, ECCAT (if available), ECLOC, ECMOOD, ECLAT, ECSTDY, VISIT, ECSTDTC, ECOCCUR, ECROUTE
SC	Subject Characteristics Dataset for Ophtha Study with variables USUBJID, SCTEST, SCTESTCD, SCCAT, SCORRES, SCDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah (HackR 2021 Team Eye)

Examples

```
sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                 SCTEST   = c("Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " ",
                              "Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " "),
                 SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", ""),
                 SCCAT    = c("STUDY EYE SELECTION", "STUDY EYE SELECTION", "",
                              "STUDY EYE SELECTION", "STUDY EYE SELECTION", ""),
                 SCORRES  = c("LEFT", "OS", "", "RIGHT", "OD", ""),
                 SCDTC    = "2021-01-01",
                 stringsAsFactors = FALSE)

ec <- data.frame(USUBJID = c(1,1,1,1,1,2,2,2,2,2,2),
                 ECCAT   = c("Fellow", "Study", "Study", "Study", "StudY",
                             "Fellow", "Fellow", "STUDY", "STUDY", "STUDY", ""),
                 ECMOOD  = rep("Performed", 11),
                 ECLOC   = rep("Eye", 11),
                 ECLAT   = c("LEFT", "Left", "left", "LEFT", "LEFT", "RIGHT",
                             "right", "right", "RIGHT", "RIGHT", "right"),
                 ECSTDY  = c(1, 28, 56, 84, 112, 1, 28, 56, 84, 112, 140),
                 VISIT   = c("Week 1", "Week 4", "Week 8", "Week 12", "Week 16",
                          "Week 1", "Week 4", "Week 8", "Week 12", "Week 16", "Week 20"),
            ECSTDTC = c("2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                        "2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                             "2021-06-01"),
                 ECOCCUR = "Y",
                 ECROUTE = "INTRAVITREAL",
                 stringsAsFactors=FALSE)

check_ec_sc_lat(SC=sc, EC=ec)

sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                 SCTEST   = c("Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " ",
                              "Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " "),
                 SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", ""),
                 SCCAT    = c("STUDY EYE SELECTION", "STUDY EYE SELECTION", "",
                              "STUDY EYE SELECTION", "STUDY EYE SELECTION", ""),
                 SCORRES  = c("LEFT", "OS", "", "RIGHT", "OD", ""),
                 SCDTC    = "2021-01-01",
                 stringsAsFactors = FALSE)

ec <- data.frame(USUBJID = c(1,1,1,1,1,2,2,2,2,2,2),
                 ECCAT   = c("Fellow", "Study", "Study", "Study", "StudY",
                             "Fellow", "Fellow", "STUDY", "STUDY", "STUDY", ""),
                 ECMOOD  = rep("Performed", 11),
```

```
                  ECLOC   = rep("Eye", 11),
                  ECLAT   = c("LEFT", "Left", "left", "LEFT", "RIGHT", "RIGHT",
                              "right", "right", "RIGHT", "RIGHT", "left"),
                  ECSTDY  = c(1, 28, 56, 84, 112, 1, 28, 56, 84, 112, 140),
                  VISIT   = c("Week 1", "Week 4", "Week 8", "Week 12", "Week 16",
                           "Week 1", "Week 4", "Week 8", "Week 12", "Week 16", "Week 20"),
            ECSTDTC = c("2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                        "2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                              "2021-06-01"),
                  ECOCCUR = "Y",
                  ECROUTE = "OPHTHALMIC",
                  stringsAsFactors=FALSE)

check_ec_sc_lat(SC=sc, EC=ec)

sc <- data.frame(USUBJID  = c(1,1,1,2,2,2,3),
                  SCTEST   = c("Eye Meeting Eligibility Criteria",
                               "Focus of Study-Specific Interest",
                               " ",
                               "Eye Meeting Eligibility Criteria",
                               "Focus of Study-Specific Interest",
                               " ",
                               "Focus of Study-Specific Interest"),
                  SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", "", "FOCID"),
                  SCCAT    = c("STUDY EYE SELECTION", "STUDY EYE SELECTION", "",
                               "STUDY EYE SELECTION",
                               "STUDY EYE SELECTION", "", "STUDY EYE SELECTION"),
                  SCORRES  = c("LEFT", "OS", "", "RIGHT", "OD", "", "RIGHT"),
                  SCDTC    = "2021-01-01",
                  stringsAsFactors = FALSE)

ec <- data.frame(USUBJID = c(1,1,1,1,1,2,2,2,2,2,2),
                  ECMOOD  = "Performed",
                  ECLOC   = "Eye",
                  ECLAT   = c("LEFT", "Left", "left", "LEFT", "RIGHT", "RIGHT",
                              "right", "right", "RIGHT", "RIGHT", "left"),
                  ECSTDY  = c(1, 28, 56, 84, 112, 1, 28, 56, 84, 112, 140),
                  VISIT   = c("Week 1", "Week 4", "Week 8", "Week 12", "Week 16",
                           "Week 1", "Week 4", "Week 8", "Week 12", "Week 16", "Week 20"),
            ECSTDTC = c("2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                        "2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                              "2021-06-01"),
                  ECOCCUR = c("Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "N"),
                  stringsAsFactors=FALSE)

check_ec_sc_lat(SC=sc, EC=ec)
```

`check_eg_egdtc_visit_ordinal_error` *Check that all ECG datetimes are earlier than last visit's (possible datetime data entry error)*

Description

This check identifies EGDTC values that are earlier than last visit's. Unscheduled visits are excluded.

Usage

```
check_eg_egdtc_visit_ordinal_error(EG)
```

Arguments

EG	ECG Test Results SDTM dataset with variables USUBJID, VISITNUM, VISIT, EGDTC, EGSTAT

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

James Zhang

Examples

```
# No case
EG<- data.frame(USUBJID = 101:102,
                EGDTC=rep(c("2017-01-01T08:25", "2017-01-05T09:25",
                "2017-01-15T10:25", "2017-01-20T08:25", "2017-01-25T08:25"), 2),
                VISITNUM=rep(1:5,2),
                VISIT=rep(c("Screening", "Cycle 1", "Cycle 2",
                "Cycle 3","UNschedUled"),2),
                EGSTAT="",
                stringsAsFactors=FALSE)
check_eg_egdtc_visit_ordinal_error(EG)

# Cases with earlier datetime
EG$EGDTC[EG$USUBJID == 101 & EG$VISIT == "Cycle 3"] <- "2017-01-10T08:25"
EG$EGDTC[EG$USUBJID == 102 & EG$VISIT == "Cycle 1"] <- "2017-01-01T06:25"
check_eg_egdtc_visit_ordinal_error(EG)

# Cases with duplicated datetime
EG$EGDTC[EG$USUBJID == 101 & EG$VISIT == "Cycle 3"] <- "2017-01-15T10:25"
EG$EGDTC[EG$USUBJID == 102 & EG$VISIT == "Cycle 2"] <- "2017-01-01T06:25"
check_eg_egdtc_visit_ordinal_error(EG)

# Not checking duplicates
```

```
EG<- data.frame(USUBJID = rep("101",6),
                EGDTC=rep("2017-01-01T08:25", 6),
                VISITNUM=rep(1:2,3),
                VISIT=rep("Screening",6),
                EGSTAT="",
                stringsAsFactors=FALSE)

check_eg_egdtc_visit_ordinal_error(EG)
```

---

check_ex_dup *Check for duplicate EX records*

Description

This check looks for duplicate treatment records in EX

Usage

```
check_ex_dup(EX)
```

Arguments

EX	Exposure SDTM dataset with variables USUBJID, EXTRT, EXDOSE, EXSTDTC, EXSTDTC. VISIT is optional.

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Fang Yuan

Examples

```
EX <- data.frame(
 USUBJID = rep(1,2),
 EXTRT = rep(1,2),
 EXDOSE = rep(1,2),
 EXSTDTC = rep(1,2),
 EXOCCUR = "Y"
)
check_ex_dup(EX)

EX$EXOCCUR <- NULL

check_ex_dup(EX)

EX$EXDOSE <- NULL
```

```
check_ex_dup(EX)

# test with sample data without duplicates

EX <- data.frame(
USUBJID = 1:2,
EXTRT = 1:2,
EXDOSE = 1:2,
EXSTDTC = 1:2,
EXOCCUR = "Y"
)

check_ex_dup(EX)

EX = rbind(EX,EX)

check_ex_dup(EX)

# check non existing vars

EX$EXTRT <- NULL
EX$EXOCCUR <- NULL

check_ex_dup(EX)
```

`check_ex_exdose_exoccur` *Check for Missing EXDOSE.*

Description

This checks looks for missing EXDOSE values when EXOCCUR="Y" or when EXOCCUR does not exist. It could be for a specified drug/treatment, or for all drugs/treatments in the dataset

Usage

```
check_ex_exdose_exoccur(EX, drug = NULL)
```

Arguments

`EX`	Exposure SDTM dataset with variables USUBJID, EXTRT, EXSTDTC, EXDOSE, and optional variable EXOCCUR and optional variable VISIT
`drug`	Drug name for EXTRT; used to subset the dataset. Default value is NULL (i.e. no filtering by drug)

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the test failed

Author(s)

Will Harris, Pasha Foroudi

Examples

```
EX <- data.frame(
 USUBJID = 1:3,
 EXSEQ   = 1:3,
 EXSTDTC = 1:3,
 EXTRT   = c(1,2,NA),
 EXOCCUR = "Y",
 EXDOSE  = 1:3,
 VISIT = c("CYCLE 1 DAY 1", "CYCLE 2 DAY 1", "CYCLE 3 DAY 1")
)

check_ex_exdose_exoccur(EX)

EX$EXDOSE[3]=NA
check_ex_exdose_exoccur(EX)

EX$EXVISIT = NULL
check_ex_exdose_exoccur(EX)

EX$EXDOSE = NULL
check_ex_exdose_exoccur(EX)
```

---

check_ex_exdose_pos_exoccur_no

Check for EXDOSE>0 When EXOCCUR is not "Y"

Description

This checks looks for EXDOSE values greater than 0 when EXOCCUR is not "Y". It could be for a specified drug/treatment, or for all drugs/treatments in the dataset.

Usage

```
check_ex_exdose_pos_exoccur_no(EX, drug = NULL)
```

Arguments

EX	Exposure SDTM dataset with variables USUBJID, EXTRT, EXSTDTC, EXOCCUR and EXDOSE
drug	Drug name for EXTRT; used to subset the dataset. Default value is NULL (i.e. no filtering by drug)

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
EX <- data.frame(
 USUBJID = 1:5,
 EXSTDTC = rep("2017-01-01",5),
 EXTRT   = c(rep("TRT A",2),rep("TRT B",3)),
 EXOCCUR = c(".","", "N", "N", "Y"),
 EXDOSE  = 0:4,
 VISIT = "VISIT 1",
 stringsAsFactors = FALSE
)

check_ex_exdose_pos_exoccur_no(EX)

check_ex_exdose_pos_exoccur_no(EX, drug = "TRT A")
check_ex_exdose_pos_exoccur_no(EX, drug = "TRT B")

EX$EXDOSE = NULL

check_ex_exdose_pos_exoccur_no(EX)
```

---

check_ex_exdosu *Check for missing EXDOSU records*

Description

This check looks for missing EXODOSU values for valid doses

Usage

```
check_ex_exdosu(EX)
```

Arguments

EX	Exposure SDTM dataset with variables USUBJID,EXTRT,EXSTDTC,EXDOSU

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Jen Chen

Examples

```
EX <- data.frame(
 USUBJID = 1:10,
 EXTRT = 1:10,
 EXSTDTC = 1:10,
 EXDOSE = 1:10,
 EXOCCUR = as.character(c(rep("Y",5),rep("N",5))),
 EXDOSU = as.character(rep("mg",10))
)

EX$EXDOSU[1] = ""
EX$EXDOSU[2] = "NA"
EX$EXDOSU[3] = NA

check_ex_exdosu(EX)

EX$EXSTDTC = NULL

check_ex_exdosu(EX)
```

---

check_ex_exoccur_exdose_exstdtc

Check for Invalid EXDOSE (Dose per Administration) and Missing/Incomplete EXSTDTC (Start Date) Values for valid exposures

---

Description

This check looks for valid exposures (EXOCCUR=Y or doesn't exist) but EXDOSE (dose per administration) is not > 0 (>= 0 in case of placebo) and/or EXSTDTC (start date/treatment date) is missing or incomplete in the EX (exposure) SDTM domain

Usage

```
check_ex_exoccur_exdose_exstdtc(EX)
```

Arguments

EX	Exposure SDTM dataset with variables USUBJID, VISIT, VISITNUM, EXOCCUR, EXTRT, EXDOSE, EXSTDTC and EXENDTC

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Sara Bodach

Examples

```
EX <- data.frame(USUBJID = LETTERS[1:5],
                 VISIT = paste0("Visit ", 1:5),
                 VISITNUM = 1:5,
                 EXOCCUR = c('Y', rep('', 4)),
                 EXTRT = LETTERS[1:5],
                 EXDOSE = 1:5,
                 EXSTDTC = c('2010-01-01', rep('', 4)),
                 EXENDTC = c('2010-01-01', rep('', 4)),
                 stringsAsFactors = FALSE)

EX$EXOCCUR[2] <- 'Y'
EX$EXSTDTC[2] <- '2011'
EX$EXDOSE[1] <- 0

check_ex_exoccur_exdose_exstdtc(EX)

EX$VISIT <- NULL

check_ex_exoccur_exdose_exstdtc(EX)
```

---

check_ex_exoccur_mis_exdose_nonmis

Check for missing EXOCCUR but EXDOSE not missing

Description

Checks for exposure records with missing EXOCCUR but EXDOSE not missing

Usage

```
check_ex_exoccur_mis_exdose_nonmis(EX)
```

Arguments

EX	Exposure dataframe with variables USUBJID, EXTRT, EXDOSE, EXOCCUR, EXSTDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Iris Zhao

Examples

```
 EX <- data.frame(
 USUBJID = 1:10,
 EXTRT = rep(1,10),
 EXOCCUR = c(rep(1,2),rep(NA,4),rep(2,4)),
 EXDOSE = c(rep(NA,4),rep(1,6)),
 EXSTDTC = 1:10
)

EX$EXOCCUR[6]="NA"
EX$EXOCCUR[7]=""
EX$EXOCCUR[8]=NA

check_ex_exoccur_mis_exdose_nonmis(EX)
```

---

check_ex_exstdtc_after_dd

Check for EX dates occurring after death date

Description

This check looks for EX dates that occur after death date

Usage

```
check_ex_exstdtc_after_dd(AE, DS, EX)
```

Arguments

AE	Adverse Event SDTM dataset with variables USUBJID, AEDTHDTC, AESTDTC, AEDECOD, and AETERM
DS	Disposition SDTM dataset with variables USUBJID, DSSTDTC, DSDECOD, and DSTERM
EX	Exposure SDTM dataset with variables USUBJID, EXSTDTC, EXTRT, and EXDOSE

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Nina Ting Qi

Examples

```
AE <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 AEDTHDTC = c(rep("", 4), "2016-01-01"),
                 AESTDTC = rep("2016-01-01", 5),
                 AEDECOD = LETTERS[1:5], AETERM = LETTERS[1:5],
                 stringsAsFactors = FALSE)

DS <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 DSSTDTC = rep("2016-01-02", 5),
                 DSDECOD = c(LETTERS[1:4], "death"),
                 DSTERM = letters[1:5],
                 stringsAsFactors = FALSE)

EX <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 EXSTDTC = rep("2015-12-31", 5),
                 EXTRT = LETTERS[1:5],
                 EXDOSE = 1:5,
                 stringsAsFactors = FALSE)

check_ex_exstdtc_after_dd(AE, DS, EX)

EX$EXSTDTC[1] <- "2016-01-03"
EX$USUBJID[1] <- EX$USUBJID[5]

check_ex_exstdtc_after_dd(AE, DS, EX)
```

---

check_ex_exstdtc_after_exendtc

Check that all exposure start dates are on or before exposure end dates

---

Description

This check identifies EXSTDTC values that are after EXENDTC values

Usage

```
check_ex_exstdtc_after_exendtc(EX)
```

Arguments

EX	Exposure SDTM dataset with variables USUBJID,EXTRT,EXSTDTC,EXENDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
EX <- data.frame(
 STUDYID = 1,
 USUBJID = 1:12,
 EXTRT = "SOME DRUG",
 EXSTDTC = c("2017-01-01","2017-01-03","2017-01-01T14:26","2017","2017-02","2017"    ,""   ,
         "2017"  ,"2017-01-01T14:26","2017-01-01T14:26","2017-01-01T14","2017-01-01T14:26:02")
             ,
 EXENDTC = c("2017-01-01","2017-01-02","2017-01-01T14:25","2015","2017-01","2016-01-01","2000",
            "2017-02","2017-01-01"     ,"2017-01","2017-01-01T13","2017-01-01T14:26:01")
              ,
 EXOCCUR = "Y",
 VISIT = "CYCLE 1 DAY 1",
 stringsAsFactors=FALSE
)

check_ex_exstdtc_after_exendtc(EX)

EX$EXOCCUR <- NULL
EX$VISIT <- NULL
check_ex_exstdtc_after_exendtc(EX)

EX$EXTRT <- NULL
check_ex_exstdtc_after_exendtc(EX)
```

---

check_ex_exstdtc_visit_ordinal_error

Check that all EX start dates are earlier than last visit's (possible date-time data entry error)

Description

This check identifies EXSTDTC values that are earlier than last visit's. Unscheduled visits are excluded.

Usage

```
check_ex_exstdtc_visit_ordinal_error(EX)
```

Arguments

EX	Exposure dataset with variables USUBJID, EXTRT, VISITNUM, VISIT, EXDTC, optional variable EXOCCUR

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

James Zhang

Examples

```
# no case
EX <- data.frame(USUBJID = 101:102,
                 EXTRT = rep(c("A", "B"), 5),
                 EXSTDTC = rep(c("2017-01-01T08:25", "2017-01-05T09:25",
                  "2017-01-15T10:25","2017-01-20T08:25","2017-01-25T08:25"), 2),
                 VISITNUM = rep(1:5,2),
             VISIT = rep(c("Cycle 1", "Cycle 2", "Cycle 3", "Cycle 4", "uNscheDuledd"), 2),
                 stringsAsFactors = FALSE)
check_ex_exstdtc_visit_ordinal_error(EX)

# adding cases with earlier date
EX$EXSTDTC[EX$USUBJID == 101 & EX$VISIT == "Cycle 4"] <- "2017-01-10T08:25"
EX$EXSTDTC[EX$USUBJID == 102 & EX$VISIT == "Cycle 2"] <- "2017-01-01T06:25"
check_ex_exstdtc_visit_ordinal_error(EX)

# adding cases with duplicated date
EX$EXSTDTC[EX$USUBJID == 101 & EX$VISIT == "Cycle 5"] <- "2017-01-10T08:25"
EX$EXSTDTC[EX$USUBJID == 102 & EX$VISIT == "Cycle 3"] <- "2017-01-01T06:25"
check_ex_exstdtc_visit_ordinal_error(EX)
```

---

check_ex_extrt_exoccur

Check for EX records where EXTRT is missing

Description

This check looks for EX records where EXTRT is missing but EXOCCUR=Y (or EXOCCUR doesn't exist) and returns a data frame

Usage

```
check_ex_extrt_exoccur(EX)
```

Arguments

EX	Exposure domain with variables USUBJID, EXSTDTC, EXTRT, EXDOSE

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Betty Wang

Examples

```
EX <- data.frame(
USUBJID = 1:10,
EXTRT = 1:10,
EXOCCUR = c(rep("Y",5), rep("",5)),
EXSTDTC = "2016-01-01",
EXDOSE = 1:10,
stringsAsFactors=FALSE
)

EX$EXTRT[1]=""
EX$EXTRT[2]="NA"
EX$EXTRT[3]=NA
EX$EXTRT[6]=""
EX$EXTRT[7]="NA"
EX$EXTRT[8]=NA

check_ex_extrt_exoccur(EX)

EX$EXOCCUR=NULL

check_ex_extrt_exoccur(EX)
```

---

check_ex_infusion_exstdtc_exendtc

Check that an infusion drug has same start/end exposure dates, also including missing start/end dates

---

Description

This check identifies that an infusion drug has same EXSTDTC and EXENDTC dateparts. If time is available for both dates, also check that end time is after start time. Missing start/end dates are also included.

Usage

```
check_ex_infusion_exstdtc_exendtc(EX)
```

Arguments

EX	Exposure SDTM dataset with variables USUBJID,EXTRT,EXSTDTC,EXENDTC,EXROUTE

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Anastasiia Khmelnytska, Stella Banjo(HackR 2021)

Examples

```
EX <- data.frame(
 STUDYID = 1,
 USUBJID = 1:12,
 EXTRT = "SOME DRUG",
 EXROUTE = "INTRAVENOUS",
 EXSTDTC = c("2017-01-01","2017-01-02","2017-01-01T14:36","2015","2017-02","2017"    ,""   ,
         "2017"  ,"2017-01-01T14:26","2017-01-01T14:26","2017-01-01T14","2017-01-01T14:26:01")
             ,
 EXENDTC = c("2017-01-01","2017-01-03","2017-01-01T14:35","2017","2017-01","2016-01-01","2000",
            "2017-02","2017-01-01"     ,"2017-01","2017-01-01T13","2017-01-02T14:26:02")
             ,
 EXOCCUR = "Y",
 VISIT = "CYCLE 1 DAY 1",
 stringsAsFactors=FALSE
)

 check_ex_infusion_exstdtc_exendtc(EX)

 EX2 <- data.frame(
 STUDYID = 1,
 USUBJID = 1:4,
 EXTRT = "SOME DRUG",
 EXROUTE = "INTRAVENOUS",
 EXSTDTC = c("2017-01-03", "", "2017-02-01T14:26", ""),
 EXENDTC = c("", "2017-02-03", "", "2017-02-02T14:26:02"),
 EXOCCUR = "Y",
 VISIT = "CYCLE 1 DAY 1",
 stringsAsFactors = FALSE
```

```
)

check_ex_infusion_exstdtc_exendtc(EX2)

EX3 <- data.frame(
    STUDYID = 1,
    USUBJID = 1:3,
    EXTRT = "SOME DRUG",
    EXROUTE = "INTRAVENOUS",
    EXSTDTC = c("2017-01-01", "2017-01-01T14:26", "2017-01-01T14:26"),
    EXENDTC = c("2017-01-01", "2017-01-01", "2017-01"),
    EXOCCUR = "Y",
    VISIT = "CYCLE 1 DAY 1",
    stringsAsFactors=FALSE
)

check_ex_infusion_exstdtc_exendtc(EX3)
```

`check_ex_visit` *Check for missing EX.VISIT*

Description

This check looks missing EX.VISIT values when EX.EXOCCUR=Y (or EX.EXOCCUR doesn't exist)

Usage

```
check_ex_visit(EX)
```

Arguments

`EX`	Exposure SDTM dataset with variables USUBJID,EXTRT,EXSTDTC,VISIT, and optional variable EXOCCUR

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Jen Chen

Examples

```
 EX <- data.frame(
 USUBJID = 1:3,
 EXTRT = 1:3,
 EXSTDTC = 1:3,
 EXOCCUR = "Y",
 VISIT = NA
)

check_ex_visit(EX)

EX$EXOCCUR=NULL

check_ex_visit(EX)

EX$VISIT=NULL

check_ex_visit(EX)#
```

`check_lb_lbdtc_after_dd` *Check for LB dates occurring after death date*

Description

This check looks for LB dates that occur after death date

Usage

```
check_lb_lbdtc_after_dd(AE, DS, LB)
```

Arguments

AE	Adverse Event SDTM dataset with variables USUBJID, AEDTHDTC, AESTDTC, AEDECOD, and AETERM
DS	Disposition SDTM dataset with variables USUBJID, DSSTDTC, DSDECOD, and DSTERM
LB	Laboratory Test Findings SDTM dataset with variables USUBJID, LBDTC, LBTESTCD, and LBORRES

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Nina Ting Qi

Examples

```
AE <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 AEDTHDTC = c(rep("", 4), "2016-01-01"),
                 AESTDTC = rep("2016-01-01", 5),
                 AEDECOD = LETTERS[1:5], AETERM = LETTERS[1:5],
                 stringsAsFactors = FALSE)

DS <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 DSSTDTC = rep("2016-01-02", 5),
                 DSDECOD = c(LETTERS[1:4], "death"),
                 DSTERM = letters[1:5],
                 stringsAsFactors = FALSE)

LB <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 LBDTC = rep("2015-12-31", 5),
                 LBTESTCD = letters[1:5],
                 LBORRES = 1:5,
                 stringsAsFactors = FALSE)

check_lb_lbdtc_after_dd(AE, DS, LB)

LB$LBDTC[1] <- "2016-01-03"
LB$USUBJID[1] <- LB$USUBJID[5]

check_lb_lbdtc_after_dd(AE, DS, LB)
```

---

check_lb_lbdtc_visit_ordinal_error

Check that all LB dates are duplicated or earlier than last visit's (possible datetime data entry error)

Description

This check identifies LBDTC values that are duplicated or earlier than last visit's. Records with LBSTAT == 'NOT DONE' and unscheduled visits (VISIT with the string "UNSCHEDU") and treatment discon visits (VISIT with the string "TREATMENT OR OBSERVATION FU COMP EARLY DISC") are excluded.

Usage

```
check_lb_lbdtc_visit_ordinal_error(LB)
```

Arguments

LB	SDTM dataset with variables USUBJID, VISITNUM, VISIT, LBDTC, LBTESTCD, LBSTAT

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Simon Luo

Examples

```
# no case
LB1 <- data.frame(USUBJID = c(rep("101", 5), rep("102", 5)),
                  LBCAT = "Hematology",
                  LBDTC = rep(c(
                  "2017-01-01T08:25",
                  "2017-01-05T09:25",
                  "2017-01-15T10:25",
                  "2017-01-20T08:25",
                  "2017-01-25T08:25"), 2),
                  VISITNUM = rep(1:5,2),
                  VISIT = rep(c(
                  "Visit 1",
                  "Visit 2",
                  "Visit 3",
                  "UNSCheduled!!!",
                  "VIsit 5"), 2),
                  LBSTAT = c(rep("", 9), "NOT DONE"),
                  stringsAsFactors = FALSE)

check_lb_lbdtc_visit_ordinal_error(LB1)

LB2 = LB1
LB2$LBCAT = "Virology"
LB3 <- rbind(LB1, LB2)
check_lb_lbdtc_visit_ordinal_error(LB3)

# adding cases with earlier date
LB3$LBDTC[LB3$USUBJID == 101 & LB3$VISIT == "Visit 3"] <- "2016-01-10T08:25"
LB3$LBDTC[LB3$USUBJID == 102 & LB3$VISIT == "Visit 2"] <- "2016-01-01T06:25"
check_lb_lbdtc_visit_ordinal_error(LB = LB3)

# adding cases with duplicated date
LB3$LBDTC[LB3$USUBJID == 102 & LB3$VISIT == "Visit 3"] <- "2017-01-15T10:25"
LB3 <- LB3[order(LB3$USUBJID, LB3$VISITNUM, LB3$LBDTC),]
check_lb_lbdtc_visit_ordinal_error(LB = LB3)

# check if all NOT DONE
LB4 = LB3
LB4$LBSTAT = "NOT DONE"
check_lb_lbdtc_visit_ordinal_error(LB = LB4)

# check dropping a required variable
LB4$LBSTAT = NULL
```

```
check_lb_lbdtc_visit_ordinal_error(LB = LB4)
```

---

check_lb_lbstnrlo_lbstnrhi *Check for missing lab reference ranges (LBSTNRLO, LBSTNRHI)*

Description

This check looks for missing lab reference ranges (LBSTNRLO, LBSTNRHI) in standard units when numeric result in standard unit (LBSTRESN) is not missing and returns a data frame

Usage

```
check_lb_lbstnrlo_lbstnrhi(DM, LB)
```

Arguments

DM	DM SDTM dataset with variable USUBJID, SITEID
LB	Lab SDTM dataset with variables USUBJID, LBTEST, LBSTRESN, LBSTNRLO, LBSTNRHI

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Lei Zhao

Examples

```
LB <- data.frame(
USUBJID = "1",
LBTEST = "Albumin",
LBSTRESN = 1:10,
LBSTNRLO = 1:10,
LBSTNRHI = 1:10,
stringsAsFactors=FALSE
)

LB$LBSTNRLO[1]=""
LB$LBSTNRLO[2]="NA"
LB$LBSTNRLO[3]=NA
LB$LBSTNRHI[3]=""
LB$LBSTNRHI[4]="NA"
LB$LBSTNRHI[5]=NA

DM <- data.frame(
```

```
USUBJID = "1",
SITEID = "123456",
stringsAsFactors=FALSE
)

check_lb_lbstnrlo_lbstnrhi(DM, LB)
```

`check_lb_lbstresc_char`

Check LBORRES/LBSTRESC populated with number beginning with character '>' or '<', which will yield missing AVAL in ADaM and records will be omitted in analyses such as Hy's Law

---

Description

This check looks for missing numeric standardized finding (LBSTRESN) when original finding (LBORRES) and character standardized finding (LBSTRESC) are not missing and LBORRES/LBSTRESC populated with number beginning with character '>' or '<'

Usage

```
check_lb_lbstresc_char(LB)
```

Arguments

LB	Lab SDTM dataset with variables USUBJID, LBTEST, LBDTC, LBORRES, LBORRESU, LBSTRESN, LBSTRESC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Vira Vrakina

Examples

```
 LB <- data.frame(
 USUBJID = c("Patient 1","Patient 2","Patient 3"),
 LBTEST   = "Test A",
 LBDTC   = "2017-01-01",
 LBORRES = c("5","3","7"),
 LBORRESU = rep("mg",3),
 LBSTRESC  = c("5","3","7"),
 LBSTRESN  = c(5,3,7),
 stringsAsFactors = FALSE
)
```

```
check_lb_lbstresc_char(LB)

 LB <- data.frame(
 USUBJID = c("Patient 1","Patient 2","Patient 3"),
 LBTEST   = rep("Test A", 3),
 LBDTC   = "2017-01-01",
 LBORRES = c("5","3","<7"),
 LBORRESU = rep("mg",3),
 LBSTRESC  = c("5","3","<7"),
 LBSTRESN  = c(5,3,NA),
 stringsAsFactors = FALSE
)

check_lb_lbstresc_char(LB)

LB <- data.frame(
 USUBJID = c("Patient 1","Patient 2","Patient 3"),
 LBTEST   = rep("Test A", 3),
 LBDTC   = rep("2017-01-01", 3),
 LBORRES = c("5","BLQ","<7"),
 LBORRESU = rep("mg",3),
 LBSTRESC  = c("5","BLQ","<7"),
 LBSTRESN  = c(5,NA,NA),
 stringsAsFactors = FALSE
)

check_lb_lbstresc_char(LB)
```

`check_lb_lbstresn_missing` *Check missing standard lab values (LBSTRESN/LBSTRESC)*

Description

This check looks for missing standardized finding (LBSTRESN/LBSTRESC) when original finding (LBORRES) is not missing

Usage

```
check_lb_lbstresn_missing(LB, preproc = identity, ...)
```

Arguments

LB	Lab SDTM dataset with variables USUBJID, LBTESTCD, LBDTC, LBORRES, LBORRESU, LBSTRESN, LBSTRESC, VISIT (optional), LBSPID (optional)

preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Madeleine Ma

Examples

```
LB <- data.frame(
 USUBJID = c("Patient 1","Patient 2","Patient 3"),
 LBTEST   = "Test A",
 LBTESTCD   = "TA",
 LBDTC   = "2017-01-01",
 LBORRES = c("5","6","7"),
 LBSTRESC = c("5","6","7"),
 LBORRESU = rep("mg",3),
 LBSTRESN  = c(5,6,NA),
 stringsAsFactors=FALSE
 )

check_lb_lbstresn_missing(LB)

LB$LBSTRESC[3] = ""
check_lb_lbstresn_missing(LB)

LB$LBSTRESC[1] = ""
check_lb_lbstresn_missing(LB)

LB$VISIT = "SCREENING"
check_lb_lbstresn_missing(LB)

LB$LBSPID= "FORMNAME-R:2/L:2XXXX"
check_lb_lbstresn_missing(LB,preproc=roche_derive_rave_row)

LB$LBSTRESN = NULL
check_lb_lbstresn_missing(LB)

LB$LBSTRESC = NULL
check_lb_lbstresn_missing(LB)
```

check_lb_lbstresu *Check for missing lab units (LBSTRESU)*

Description

This check identifies records where original lab values (LBORRES) exist but standard lab units (LBSTRESU) are not populated, excluding qualitative results (LBMETHOD) and excluding records when LBTESTCD in ("PH" "SPGRAV")

Usage

```
check_lb_lbstresu(LB, preproc = identity, ...)
```

Arguments

LB	Lab SDTM dataset with variables USUBJID, LBSTRESC, LBSTRESN, LBORRES, LBSTRESU, LBTESTCD, LBDTC, LBMETHOD (optional), LBSPID (optional), and VISIT (optional)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Iris Zhao

Examples

```
LB <- data.frame(
USUBJID = 1:10,
LBSTRESC = "5",
LBSTRESN = 1:10,
LBORRES = "5",
LBSTRESU = "g/L",
LBTESTCD = "ALB",
LBDTC = 1:10,
stringsAsFactors=FALSE
)

check_lb_lbstresu(LB)

LB$LBSTRESU[1]=""
check_lb_lbstresu(LB)

LB$LBSTRESU[2]="NA"
```

```
check_lb_lbstresu(LB)

LB$LBSTRESU[3]=NA
check_lb_lbstresu(LB)

LB$LBSPID= "FORMNAME-R:2/L:2XXXX"
check_lb_lbstresu(LB,preproc=roche_derive_rave_row)

LB$VISIT= "SCREENING"
check_lb_lbstresu(LB)

LB$LBSTRESU=NULL
check_lb_lbstresu(LB)
```

`check_lb_missing_month` *Check for lab dates with year and day known but month unknown*

Description

Check for missing month when lab specimen collection date (LBDTC) has known year and day

Usage

```
check_lb_missing_month(LB, preproc = identity, ...)
```

Arguments

`LB`	Laboratory data SDTM dataset with variables USUBJID,LBTEST,LBDTC,VISIT
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
LB <- data.frame(
 USUBJID = 1:4,
 LBTEST = c("TEST1","TEST2","TEST3","TEST3"),
 LBDTC = c("2017-01-01","2017-02-01","2017---01", "2017----01"),
 VISIT = c("VISIT1","VISIT2","VISIT3","VISIT3"),
 stringsAsFactors=FALSE
```

```
)

check_lb_missing_month(LB)

LB$LBSPID= "FORMNAME-R:2/L:2XXXX"

check_lb_missing_month(LB,preproc=roche_derive_rave_row)

LB$LBDTC = NULL

check_lb_missing_month(LB)
```

`check_mh_missing_month` *Check for MH dates with year and day known but month unknown*

Description

This check looks for partial missing dates in medical history start and end dates. That is, with only the month missing while the year and day are known

Usage

```
check_mh_missing_month(MH, preproc = identity, ...)
```

Arguments

MH	Medical History SDTM dataset with variables USUBJID, MHTERM and MHSTDTC
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Chandra Mannem

Examples

```
MH <- data.frame(USUBJID = LETTERS[1:5],
                 MHTERM = LETTERS[5:1],
                 MHSTDTC = c("2014", NA, "2014-01", "", "2014---02"),
                 stringsAsFactors = FALSE)

check_mh_missing_month(MH)
```

```
MH$MHSPID= "FORMNAME-R:2/L:2XXXX"

check_mh_missing_month(MH,preproc=roche_derive_rave_row)
```

`check_mi_mispec` *Check for missing values in the MISPEC variable*

Description

This check looks for missing values in the MISPEC variable, which is required. This will be flagged in P21. This may reflect a mapping issue.

Usage

```
check_mi_mispec(MI)
```

Arguments

`MI`	Microscopic Findings with variables USUBJID, MISPEC, MITESTCD, MIDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Stella Banjo (HackR 2021)

Examples

```
MI <- data.frame(
  USUBJID = c("1","2", "3"),
  DOMAIN = "MI",
  MISEQ = c(1, 2, 1),
  MISPEC = c("","BLOCK SLIDE",NA),
  MITESTCD = "TESTCD1",
  MIDTC = "2020-01-01",
stringAsFactors = FALSE
)

 check_mi_mispec(MI)

## No errors, MISPEC values present
MI2 <- data.frame(
 USUBJID = c("1","2", "3"),
 DOMAIN = "MI",
 MISEQ = 1,
 MISPEC = c("SLIDE", "TUMOR TISSUE", "BLOCK SLIDE"),
 MITESTCD = "TESTCD1",
```

```
 MIDTC = "",
 stringsAsFactors = FALSE
)

 check_mi_mispec(MI2)
```

`check_oe_bcva_1m_late_early_tot` *Check if 1m BCVA test stops too late, too early and has correct total*

Description

This ophthalmology check is for BCVA 1m test. It checks three conditions: <1> BCVA test stops too late, meaning that lines were read after number of correct letters is <= 3. <2> BCVA test stops too early, meaning that further lines were not read when all numbers of correct letters is > 3. <3> BCVA total score is not correct, meaning that the sum of the number of correct at 1 meter doesn't match with what has been recorded in eCRF (BCVA Scores eCRF Page - C. Total number correct at 1m). Please note that this check only works with USUBJID, VISIT, VISITNUM, OELOC, OELAT combination has unique dates (OEDTC). If your datasets are having situations like 1) unscheduled visits happening on different dates or 2) BCVA TOTAL happens on a different date from BCVA row tests, such combinations will be removed from check. Please note that this check excludes forms BCVA Low Vision Test (BCV5), BCVA Scores (BCV7), BCVA Low Luminance Scores (BCVLL5), BCVA Combined Assessments (BCVAC), BCVA Low Luminance Combined Assessments (BCVACLL) before running check as these forms do not include Row numbers.

Usage

```
check_oe_bcva_1m_late_early_tot(OE)
```

Arguments

OE	Ophtho Dataset with variables USUBJID, OESPID, OECAT, OESCAT, OETSTDTL, OESTRESN, OESTAT, OELOC, OELAT, OERESCAT, VISIT, VISITNUM, OEDTC, OEDY

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Rosemary Li (HackR 2021 Team Eye)

See Also

Other OPHTH: `check_ae_aelat()`, `check_cm_cmlat()`, `check_cm_cmlat_prior_ocular()`, `check_oe_bcva_4m_late_e`
`check_oe_bcva_4m_vs_1m_req()`, `check_oe_bcva_tot_mismatch()`, `check_oe_sc_lat_count_fingers()`,
`check_pr_prlat()`, `check_sc_dm_eligcrit()`, `check_sc_dm_seyeselc()`

Examples

```
OE_too_late <- data.frame(
  USUBJID = "1",
  OESPID = "FORMNAME-R:2/L:2XXXX",
  OECAT = "BEST CORRECTED VISUAL ACUITY",
  OETSTDTL = "TESTING DISTANCE: 1M",
  OESCAT = c(rep("", 6), "TOTAL"),
  OESTAT = "",
  ORESCAT = c("ROW 1 - SNELLEN 20/200",
              "ROW 2 - SNELLEN 20/160",
              "ROW 4 - SNELLEN 20/100",
              "ROW 3 - SNELLEN 20/125",
              "ROW 5 - SNELLEN 20/80",
              "ROW 6 - SNELLEN 20/63",
              ""),
  VISIT = "WEEK 1",
  VISITNUM = 5,
  OEDTC = "2020-06-01",
  OEDY = 8,
  OELOC = "EYE",
  OELAT = "LEFT",
  OESTRESN = c(5, 5, 5, 4, 3, 2, 24)
)
check_oe_bcva_1m_late_early_tot(OE_too_late)

OE_too_early <- data.frame(
  USUBJID = "1",
  OESPID = "FORMNAME-R:2/L:2XXXX",
  OECAT = "BEST CORRECTED VISUAL ACUITY",
  OETSTDTL = "TESTING DISTANCE: 1M",
  OESCAT = c(rep("", 5), "TOTAL"),
  OESTAT = "",
  ORESCAT = c("ROW 1 - SNELLEN 20/200",
              "ROW 2 - SNELLEN 20/160",
              "ROW 4 - SNELLEN 20/100",
              "ROW 3 - SNELLEN 20/125",
              "ROW 5 - SNELLEN 20/80",
              ""),
  VISIT = "WEEK 1",
  VISITNUM = 5,
  OEDTC = "2020-06-01",
  OEDY = 8,
  OELOC = "EYE",
  OELAT = "LEFT",
  OESTRESN = c(5, 5, 5, 4, 4, 23)
)
check_oe_bcva_1m_late_early_tot(OE_too_early)

OE_total_incorrect <- data.frame(
  USUBJID = "1",
  OESPID = "FORMNAME-R:2/L:2XXXX",
  OECAT = "BEST CORRECTED VISUAL ACUITY",
```

```
  OETSTDTL = "TESTING DISTANCE: 1M",
  OESCAT = c(rep("", 6), "TOTAL"),
  OESTAT = "",
  OERESCAT = c("ROW 1 - SNELLEN 20/200",
               "ROW 2 - SNELLEN 20/160",
               "ROW 4 - SNELLEN 20/100",
               "ROW 3 - SNELLEN 20/125",
               "ROW 5 - SNELLEN 20/80",
               "ROW 6 - SNELLEN 20/63",
               ""),
  VISIT = "WEEK 1",
  VISITNUM = 5,
  OEDTC = "2020-06-01",
  OEDY = 8,
  OELOC = "EYE",
  OELAT = "LEFT",
  OESTRESN = c(5, 5, 5, 4, 4, 2, 28)
)
check_oe_bcva_1m_late_early_tot(OE_total_incorrect)
```

`check_oe_bcva_4m_late_early_tot` *Check if 4m BCVA test stops too late, too early and has correct total*

Description

This ophthalmology check is for BCVA 4m test. It checks three conditions: <1> BCVA test stops too late, meaning that lines were read after number of correct letters is <= 3. <2> BCVA test stops too early, meaning that further lines were not read when all numbers of correct letters is > 3. <3> BCVA total score is not correct, meaning that the sum of the number of correct at 4 meters doesn't match with what has been recorded in eCRF (BCVA Scores eCRF Page - A. Total number correct at 4m). Please note that this check only works with USUBJID, VISIT, VISITNUM, OELOC, OELAT combination has unique dates (OEDTC). If your datasets are having situations like 1) unscheduled visits happening on different dates or 2) BCVA TOTAL happens on a different date from BCVA row tests, such combinations will be removed from check. Please note that this check excludes forms BCVA Low Vision Test (BCV5), BCVA Scores (BCV7), BCVA Low Luminance Scores (BCVLL5), BCVA Combined Assessments (BCVAC), BCVA Low Luminance Combined Assessments (BCVACLL) before running check as these forms do not include Row numbers.

Usage

```
check_oe_bcva_4m_late_early_tot(OE)
```

Arguments

OE	Ophtho Dataset with variables USUBJID, OESPID, OECAT, OESCAT, OETSTDTL, OESTRESN, OESTAT, OELOC, OELAT, OERESCAT, VISIT, VISITNUM, OEDTC, OEDY

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Rosemary Li (HackR 2021 Team Eye)

See Also

Other OPHTH: check_ae_aelat(), check_cm_cmlat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_e
check_oe_bcva_4m_vs_1m_req(), check_oe_bcva_tot_mismatch(), check_oe_sc_lat_count_fingers(),
check_pr_prlat(), check_sc_dm_eligcrit(), check_sc_dm_seyeselc()

Examples

```
OE_too_late <- data.frame(
  USUBJID = "1",
  OESPID = "FORMNAME-R:2/L:2XXXX",
  OECAT = "BEST CORRECTED VISUAL ACUITY",
  OETSTDTL = "TESTING DISTANCE: 4M",
  OESCAT = c(rep("", 6), "TOTAL"),
  OESTAT = "",
  OERESCAT = c("ROW 1 - SNELLEN 20/200",
               "ROW 2 - SNELLEN 20/160",
               "ROW 4 - SNELLEN 20/100",
               "ROW 3 - SNELLEN 20/125",
               "ROW 5 - SNELLEN 20/80",
               "ROW 6 - SNELLEN 20/63",
               ""),
  VISIT = "WEEK 1",
  VISITNUM = 5,
  OEDTC = "2020-06-01",
  OEDY = 8,
  OELOC = "EYE",
  OELAT = "LEFT",
  OESTRESN = c(5, 5, 5, 4, 3, 2, 24)
)
check_oe_bcva_4m_late_early_tot(OE_too_late)

OE_too_early <- data.frame(
  USUBJID = "1",
  OESPID = "FORMNAME-R:2/L:2XXXX",
  OECAT = "BEST CORRECTED VISUAL ACUITY",
  OETSTDTL = "TESTING DISTANCE: 4M",
  OESCAT = c(rep("", 6), "TOTAL"),
  OESTAT = "",
  OERESCAT = c("ROW 1 - SNELLEN 20/200",
               "ROW 2 - SNELLEN 20/160",
               "ROW 4 - SNELLEN 20/100",
               "ROW 3 - SNELLEN 20/125",
               "ROW 5 - SNELLEN 20/80",
               "ROW 6 - SNELLEN 20/63",
```

```
                "",
    VISIT = "WEEK 1",
    VISITNUM = 5,
    OEDTC = "2020-06-01",
    OEDY = 8,
    OELOC = "EYE",
    OELAT = "LEFT",
    OESTRESN = c(5, 5, 5, 4, 4, 5, 28)
  )
  check_oe_bcva_4m_late_early_tot(OE_too_early)

  OE_total_incorrect <- data.frame(
    USUBJID = "1",
    OESPID = "FORMNAME-R:2/L:2XXXX",
    OECAT = "BEST CORRECTED VISUAL ACUITY",
    OETSTDTL = "TESTING DISTANCE: 4M",
    OESCAT = c(rep("", 6), "TOTAL"),
    OESTAT = "",
    OERESCAT = c("ROW 1 - SNELLEN 20/200",
                 "ROW 2 - SNELLEN 20/160",
                 "ROW 4 - SNELLEN 20/100",
                 "ROW 3 - SNELLEN 20/125",
                 "ROW 5 - SNELLEN 20/80",
                 "ROW 6 - SNELLEN 20/63",
                 ""),
    VISIT = "WEEK 1",
    VISITNUM = 5,
    OEDTC = "2020-06-01",
    OEDY = 8,
    OELOC = "EYE",
    OELAT = "LEFT",
    OESTRESN = c(5, 5, 5, 4, 4, 2, 28)
  )
  check_oe_bcva_4m_late_early_tot(OE_total_incorrect)
```

`check_oe_bcva_4m_vs_1m_req` *Check if 1m BCVA test is completed per BCVA 4m result*

Description

This ophthalmology function is to check if BCVA 1m test is done per BCVA 4m result. Patient, Visits, Laterality where Low Vision Tests were done are excluded from this check. 1> If 4m test total <= 19 and 1m test is not done. 2> If 4m test total >= 20 and 1m test is performed Above two conditions will be outputted in the final result data frame, which includes USUBJID, VISIT, OEDTC, OELAT, BCVA_4M_TOTAL, BCVA_1M_TOTAL, ISSUE. Please note that this check

will assume that the BCVA 1m and 4m total are accurate and they happen on the same day. If they are happening on different dates, such records will be removed and not checked.

Usage

```
check_oe_bcva_4m_vs_1m_req(OE)
```

Arguments

OE	Ophtho Dataset with variables USUBJID, OECAT, OESCAT, OETSTDTL, OESTRESN, OESTAT, OELAT, OERESCAT, VISIT, OEDTC, OEDY

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Rosemary Li (HackR 2021 Team Eye)

See Also

Other OPHTH: check_ae_aelat(), check_cm_cmlat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_e
check_oe_bcva_4m_late_early_tot(), check_oe_bcva_tot_mismatch(), check_oe_sc_lat_count_fingers(),
check_pr_prlat(), check_sc_dm_eligcrit(), check_sc_dm_seyeselc()

Examples

```
OE_1m_done <- data.frame(
  USUBJID = "1",
  OECAT = "BEST CORRECTED VISUAL ACUITY",
  OETSTDTL = c(rep("TESTING DISTANCE: 4M", 4), rep("TESTING DISTANCE: 1M", 3)),
  OESCAT = c(rep("", 3), "TOTAL", rep("", 2), "TOTAL"),
  OESTAT = rep("", 7),
  OERESCAT = c("ROW 1 - SNELLEN 20/200",
               "ROW 2 - SNELLEN 20/160",
               "ROW 3 - SNELLEN 20/125",
               "",
               "ROW 1 - SNELLEN 20/200",
               "ROW 2 - SNELLEN 20/160",
               ""),
  VISIT = "WEEK 1",
  VISITNUM = 5,
  OEDTC = "2020-06-01",
  OEDY = 8,
  OELOC = "EYE",
  OELAT = "LEFT",
  OESTRESN = c(9, 9, 3, 21, 3, 2, 5)
)
check_oe_bcva_4m_vs_1m_req(OE_1m_done)

OE_1m_not_done <- data.frame(
```

```
  USUBJID = "1",
  OECAT = "BEST CORRECTED VISUAL ACUITY",
  OETSTDTL = "TESTING DISTANCE: 4M",
  OESCAT = c(rep("", 3), "TOTAL"),
  OESTAT = "",
  OERESCAT = c("ROW 1 - SNELLEN 20/200",
               "ROW 2 - SNELLEN 20/160",
               "ROW 3 - SNELLEN 20/125",
               ""),
  VISIT = "WEEK 1",
  VISITNUM = 5,
  OEDTC = "2020-06-01",
  OEDY = 8,
  OELOC = "EYE",
  OELAT = "LEFT",
  OESTRESN = c(5, 5, 2, 12)
)
check_oe_bcva_4m_vs_1m_req(OE_1m_not_done)
```

`check_oe_bcva_tot_mismatch` *Check mismatch between Derived BCVA Total Score & Total BCVA Score from Data*

Description

This ophthalmology check looks for any mismatch between the Derived Best Corrected Visual Acuity (BCVA) Total Score & reported Total BCVA Score from Data based on OETESTCD = "LOGSCORE" for older studies or OETESTCD = "VACSCORE" for newer studies

Usage

```
check_oe_bcva_tot_mismatch(OE)
```

Arguments

`OE`	Ophtho Dataset with variables USUBJID, OETESTCD, OECAT, OESCAT, OETSTDTL, OESTRESN, OESTAT (if present), OELAT, VISIT, OEDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah (HackR 2021 Team Eye)

See Also

Other OPHTH: check_ae_aelat(), check_cm_cmlat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_e
check_oe_bcva_4m_late_early_tot(), check_oe_bcva_4m_vs_1m_req(), check_oe_sc_lat_count_fingers(),
check_pr_prlat(), check_sc_dm_eligcrit(), check_sc_dm_seyeselc()

Examples

```
#Using Old Standard, FAIL Case (4m <=19, so 4m + 1m to match with Rave Total)
OE <- data.frame(
   USUBJID = 1,
   OETESTCD = c("NUMLCOR", "NUMLCOR", "LCORCON", "LOGSCORE"),
   OECAT   = rep("BEST CORRECTED VISUAL ACUITY", 4),
   OESCAT   = c("TOTAL", "TOTAL","", ""),
   OETSTDTL  = c("TESTING DISTANCE: 4M", "TESTING DISTANCE: 1M", "", ""),
   OESTRESN = c(18, 0, 30, 48),
   OESTAT= rep("", 4),
   OELOC   = rep("EYE", 4),
   OELAT   = rep("LEFT", 4),
   VISIT   = rep("SCREENING", 4),
   VISITNUM   = rep(99, 4),
   OEDTC = rep("2021-05-19", 4),
   OEDY  = rep(1, 4),
   stringsAsFactors = FALSE)

check_oe_bcva_tot_mismatch(OE)


#Using New Standard, PASS Case

OE <- data.frame(
   USUBJID = 1,
   OETESTCD = c("NUMLCOR", "NUMLCOR", "LCORCON", "VACSCORE"),
   OECAT   = rep("BEST CORRECTED VISUAL ACUITY", 4),
   OESCAT   = c("NORMAL LIGHTING SCORE", "NORMAL LIGHTING SCORE","", ""),
   OETSTDTL  = c("TESTING DISTANCE: 4M", "TESTING DISTANCE: 1M", "", ""),
   OESTRESN = c(22, 0, 30, 52),
   OESTAT= rep("", 4),
   OELOC   = rep("EYE", 4),
   OELAT   = rep("LEFT", 4),
   VISIT   = rep("SCREENING", 4),
   VISITNUM   = rep(99, 4),
   OEDTC = rep("2021-05-19", 4),
   OEDY  = rep(1, 4),
   stringsAsFactors = FALSE)
check_oe_bcva_tot_mismatch(OE)


#Using New Standard, FAIL Case (Total 4m + 1m (As 4m <=19) not equal to CRF Total Score)

OE <- data.frame(
   USUBJID = 1,
   OETESTCD = c("NUMLCOR", "NUMLCOR", "LCORCON", "VACSCORE"),
```

```
   OECAT   = "BEST CORRECTED VISUAL ACUITY",
   OESCAT   = c("NORMAL LIGHTING SCORE", "NORMAL LIGHTING SCORE","", ""),
   OETSTDTL  = c("TESTING DISTANCE: 4M", "TESTING DISTANCE: 1M", "", ""),
   OESTRESN = c(17, 12, 0, 27),
   OESTAT= "",
   OELOC   = "EYE",
   OELAT   = "LEFT",
   VISIT   = "SCREENING",
   VISITNUM   = 99,
   OEDTC = "2021-05-19",
   OEDY  = 1,
   stringsAtors = FALSE)
check_oe_bcva_tot_mismatch(OE)


#FAIL Case without optional variable, OESTAT

OE$OESTAT <- NULL
check_oe_bcva_tot_mismatch(OE)


#missing required variable, OETESTCD

OE$OETESTCD <- NULL
check_oe_bcva_tot_mismatch(OE)
```

`check_oe_sc_lat_count_fingers` *Check if Post Treatment Count Fingers in Study Eye laterality does not match with Subject Characteristics Study Eye laterality*

Description

Check If Post Treatment Count Fingers in Study Eye is done on the actual Study eye by comparing laterality from OE domain with SC domain. Check is ignored if Post Treatment Count Fingers is not collected in study as it is quite common for EP studies

Usage

```
check_oe_sc_lat_count_fingers(OE, SC)
```

Arguments

OE	Ophthalmic Examination Dataset for Ophtho Study with variables USUBJID, OECAT, OELAT, VISIT, OEDTC, OETEST, OELOC, OESTAT (if present)
SC	Subject Characteristics Dataset for Ophtho Study with variables USUBJID, SCTEST, SCTESTCD, SCCAT, SCORRES, SCDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah (HackR 2021 Team Eye)

See Also

Other OPHTH: check_ae_aelat(), check_cm_cmlat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_e
check_oe_bcva_4m_late_early_tot(), check_oe_bcva_4m_vs_1m_req(), check_oe_bcva_tot_mismatch(),
check_pr_prlat(), check_sc_dm_eligcrit(), check_sc_dm_seyeselc()

Examples

```
sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                 SCTEST   = c("Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " ",
                              "Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " "),
                 SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", ""),
                 SCCAT    = c("STUDY EYE SELECTION", "STUDY EYE SELECTION", "",
                              "STUDY EYE SELECTION", "STUDY EYE SELECTION", ""),
                 SCORRES  = c("LEFT", "OS", "", "RIGHT", "OD", ""),
                 SCDTC    = rep("2021-01-01", 6),
                 stringsAsFactors = FALSE)

oe <- data.frame(USUBJID = c(1,1,1,1,1,2,2,2,2,2,2),
                 OECAT  = rep("SAFETY ASSESSMENT OF LOW VISION", 11),
                 OELOC   = rep("Eye", 11),
                 OELAT   = c("LEFT", "Left", "left", "LEFT", "LEFT",
                             "RIGHT", "right", "right", "RIGHT", "RIGHT", "right"),
                 OEDY  = c(1, 28, 56, 84, 112, 1, 28, 56, 84, 112, 140),
                 VISIT   = c("Week 1", "Week 4", "Week 8", "Week 12", "Week 16",
                             "Week 1", "Week 4", "Week 8", "Week 12", "Week 16", "Week 20"),
           OEDTC = c("2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                     "2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                             "2021-06-01"),
                 OETEST = c("A", "B", "C", "D", "E", "A", "B", "C", "D", "E", "F"),
                 stringsAsFactors=FALSE)

check_oe_sc_lat_count_fingers(SC=sc, OE=oe)

sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                 SCTEST   = c("Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " ",
                              "Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " "),
```

```
                  SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", ""),
                  SCCAT    = c("STUDY EYE SELECTION", "STUDY EYE SELECTION", "",
                               "STUDY EYE SELECTION", "STUDY EYE SELECTION", ""),
                  SCORRES  = c("LEFT", "OS", "", "RIGHT", "OD", ""),
                  SCDTC    = rep("2021-01-01", 6),
                  stringsAsFactors = FALSE)

oe <- data.frame(USUBJID = c(1,1,1,1,1,2,2,2,2,2,2),
                  OECAT  = rep("SAFETY ASSESSMENT OF LOW VISION", 11),
                  OELOC   = rep("Eye", 11),
                  OELAT   = c("LEFT", "Left", "left", "LEFT", "right", "RIGHT",
                              "right", "right", "RIGHT", "RIGHT", "left"),
                  OEDY  = c(1, 28, 56, 84, 112, 1, 28, 56, 84, 112, 140),
                  VISIT   = c("Week 1", "Week 4", "Week 8", "Week 12", "Week 16",
                              "Week 1", "Week 4", "Week 8", "Week 12", "Week 16",
                              "Week 20"),
            OEDTC = c("2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                      "2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                              "2021-06-01"),
                  OETEST = c("A", "B", "C", "D", "E", "A", "B", "C", "D", "E", "F"),
                  stringsAsFactors=FALSE)

check_oe_sc_lat_count_fingers(SC=sc, OE=oe)

sc <- data.frame(USUBJID  = c(1,1,1,2,2,2,3),
                  SCTEST   = c("Eye Meeting Eligibility Criteria",
                               "Focus of Study-Specific Interest",
                               " ",
                               "Eye Meeting Eligibility Criteria",
                               "Focus of Study-Specific Interest",
                               " ",
                               "Focus of Study-Specific Interest"),
                  SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", "", "FOCID"),
                  SCCAT    = c("STUDY EYE SELECTION", "STUDY EYE SELECTION", "",
                               "STUDY EYE SELECTION", "STUDY EYE SELECTION",
                               "", "STUDY EYE SELECTION"),
                  SCORRES  = c("LEFT", "OS", "", "RIGHT", "OD", "", "OS"),
                  SCDTC    = "2021-01-01",
                  stringsAsFactors = FALSE)

oe <- data.frame(USUBJID = c(1,1,1,1,1,2,2,2,2,2,2),
                  OESTAT  = c("","","","","","","","","","", "not DONE"),
                  OECAT  = "SAFETY ASSESSMENT OF LOW VISION",
                  OELOC   = "Eye",
                  OELAT   = c("LEFT", "Left", "left", "LEFT", "right", "RIGHT",
                              "right", "right", "RIGHT", "RIGHT", "left"),
                  OEDY  = c(1, 28, 56, 84, 112, 1, 28, 56, 84, 112, 140),
                  VISIT   = c("Week 1", "Week 4", "Week 8", "Week 12", "Week 16",
                              "Week 1", "Week 4", "Week 8", "Week 12", "Week 16",
                              "Week 20"),
            OEDTC = c("2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                      "2021-01-01", "2021-02-01", "2021-03-01", "2021-04-01", "2021-05-01",
                              "2021-06-01"),
```

```
                OETEST = c("A", "B", "C", "D", "E", "A", "B", "C", "D", "E", "F"),
                stringsAsFactors=FALSE)

check_oe_sc_lat_count_fingers(SC=sc, OE=oe)
```

`check_pr_missing_month` *Check for procedure dates with year and day known but month unknown*

Description

This check looks for partial missing dates in PR Procedures start date and end date, if end date exists. If the day of the month is known, the month should be known.

Usage

```
check_pr_missing_month(PR, preproc = identity, ...)
```

Arguments

PR	Procedures SDTM dataset with variables USUBJID, PRTRT, PRSTDTC, PRENDTC (optional), PRSPID (optional)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Examples

```
PR <- data.frame(
 USUBJID = 1:3,
 PRTRT = c("Surgery Name","Procedure Name","Procedure"),
 PRSTDTC = c("2017-01-01","2017---01","2017-01-02"),
 PRENDTC = c("2017-02-01","2017-03-01","2017---01"),
 PRSPID = "/F:SURG-D:12345-R:1",
 PRCAT = "Form 1",
 stringsAsFactors=FALSE
)

check_pr_missing_month(PR)

check_pr_missing_month(PR,preproc=roche_derive_rave_row)

PR$PRENDTC = NULL
check_pr_missing_month(PR)
```

check_pr_prlat *Check if ocular procedures/surgeries has laterality missing for CRF pages which contain the word "OCULAR" (and not "NON-OCULAR").*

Description

This check assesses observations where PRCAT contains the word OCULAR and flags records with missing/inconsistent laterality

Usage

```
check_pr_prlat(PR, preproc = identity, ...)
```

Arguments

PR	Procedure/Surgery Dataset for Ophtho Study with variables USUBJID, PRCAT, PRLAT, PRTRT, PROCCUR, PRPRESP, PRSPID (if Present), PRSTDTC (if Present), PRINDC (if Present)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Tim Barnett (HackR 2021 Team Eye) Monarch Shah (Added Concurrent Ocular Procedure in this check) (copied from check_cm_cmlat)

See Also

Other OPHTH: check_ae_aelat(), check_cm_cmlat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_e
check_oe_bcva_4m_late_early_tot(), check_oe_bcva_4m_vs_1m_req(), check_oe_bcva_tot_mismatch(),
check_oe_sc_lat_count_fingers(), check_sc_dm_eligcrit(), check_sc_dm_seyeselc()

Examples

```
PR <- data.frame(
   USUBJID = 1:5,
   PRCAT = "PRIOR OCULAR SURGERIES AND PROCEDURES",
   PRSTDTC = 1:5,
   PRLAT   = c("Left", "","Bilateral", "", ""),
   PRTRT   = c("A", "B", "A", "B", "A"),
   PROCCUR = c("Y", "N", "N", "Y", "Y"),
   PRPRESP = "Y",
   PRSPID  = "FORMNAME-R:2/L:2XXXX",
```

```
   stringsAsFactors = FALSE)
check_pr_prlat(PR,preproc=roche_derive_rave_row)

PR <- data.frame(
   USUBJID = 1:5,
   PRCAT = "CONCURRENT OCULAR PROCEDURE",
   PRSTDTC = 1:5,
   PRLAT   = c("Left", "LEFT","Bilateral", "RIGHT", "RIgHT"),
   PRTRT  = c("A", "B", "A", "B", "A"),
   PROCCUR = NA,
   PRPRESP = NA,
   stringsAsFactors = FALSE)
check_pr_prlat(PR)

PR <- data.frame(
   USUBJID = 1:5,
   PRCAT = "CONCURRENT OCULAR PROCEDURE",
   PRSTDTC = 1:5,
   PRLAT   = c("Left", "LEFT","Bilateral", "RIGHT", ""),
   PRTRT  = c("A", "B", "A", "B", "A"),
   PROCCUR = NA,
   PRPRESP = NA,
   stringsAsFactors = FALSE)
check_pr_prlat(PR)

PR <- data.frame(
   USUBJID = 1:5,
   PRCAT = "CONCURRENT OCULAR PROCEDURE",
   PRSTDTC = 1:5,
   PRLAT   = c("Left", "","Bilateral", "RIGHT", ""),
   PRTRT  = c("A", "B", "A", "B", "A"),
   PROCCUR = c("Y", "N", "N", "Y", "Y"),
   PRPRESP = "Y",
   stringsAsFactors = FALSE)
check_pr_prlat(PR)

PR <- data.frame(
   USUBJID = 1:5,
  PRCAT = c(rep("CONCURRENT NON-OCULAR PROCEDURE",3),rep("CONCURRENT OCULAR PROCEDURE",2)),
   PRSTDTC = 1:5,
   PRLAT   = c("", "","", "RIGHT", ""),
   PRTRT  = c("A", "B", "A", "B", "A"),
   PROCCUR = c("Y", "N", "N", "Y", "Y"),
   PRPRESP = "Y",
   stringsAsFactors = FALSE)
check_pr_prlat(PR)
```

---

check_qs_dup *Check for multiple dates at the same visit in QS*

Description

Identifies multiple dates at the same visit in QS

Usage

```
check_qs_dup(QS)
```

Arguments

QS	QS SDTM dataset with variables USUBJID, QSCAT, VISIT, QSDTC

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the test failed

Author(s)

Yuliia Bahatska

Examples

```
QS1 <- data.frame(USUBJID = c(rep(101, 5), rep(102, 5)),
                  QSCAT = "DLQI",
                  QSDTC = rep(c("2017-01-01T08:25", "2017-01-05T09:25",
                   "2017-01-15T10:25","2017-01-20T08:25","2017-01-25T08:25"), 2),
                  VISITNUM = rep(1:5,2),
              VISIT = rep(c( "Visit 1", "Visit 2", "Visit 3", "UNSCheduled!!!","VIsit 5"), 2),
                  stringsAsFactors = FALSE)
check_qs_dup(QS = QS1)

# multiple dates for the same visit in QS
QS2 <- QS1
QS2$VISIT[QS2$USUBJID == 101] <- "Visit 1"
check_qs_dup(QS = QS2)

# multiple visit labels for the same date
QS3 <- QS1
QS3$QSDTC[QS3$USUBJID == 101] <- "2017-01-01"
QS3
check_qs_dup(QS = QS3)
```

`check_qs_qsdtc_after_dd` *Check for QS dates occurring after death date*

Description

This check looks for QS dates that occur after death date

Usage

```
check_qs_qsdtc_after_dd(AE, DS, QS)
```

Arguments

AE	Adverse Event SDTM dataset with variables USUBJID, AEDTHDTC, AESTDTC, AEDECOD, and AETERM
DS	DS Disposition SDTM dataset with variables USUBJID, DSSTDTC, DSDECOD, and DSTERM
QS	Questionnaire Test Findings SDTM dataset with variables USUBJID, QSDTC, QSCAT, and QSORRES

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah

Examples

```
AE <- data.frame(USUBJID = c(1,1,1,2,2,2),
                 AEDTHDTC = c("", "", "2016-01-01", "", "2016-01", "2016-01-01"),
                 AESTDTC = "2016-01-01",
                 AEDECOD = LETTERS[1:6],
                 AETERM = LETTERS[1:6],
                 stringsAsFactors = FALSE)

DS <- data.frame(USUBJID = c(1,1,1,2,2,2),
                 DSSTDTC = "2016-01-01",
                 DSDECOD = c("A", "B", "death", "AC", "BC", "death"),
                 DSTERM = letters[1:6],
                 stringsAsFactors = FALSE)

QS <- data.frame(USUBJID = c(1,1,1,2,2,2),
                 QSDTC   = c("2015-06-30", "2015-09-30", "2015-12-30",
                             "2015-06-30", "2015-09-30", "2015-12-30"),
                 QSCAT   = "A",
```

```
                  QSORRES =  LETTERS[1:6],
                  QSSTAT  = "",
                VISIT  =  c("Week 1", "Week 12", "Week 24", "Week 1", "Week 12", "Week 24"),
                  QSSTRESC = LETTERS[1:6],
                  stringsAsFactors = FALSE)

check_qs_qsdtc_after_dd(AE, DS, QS)

QS$QSDTC[3:5] <- "2016-01-03"
check_qs_qsdtc_after_dd(AE, DS, QS)

QS$QSSTAT[3] <- "Not Done"
check_qs_qsdtc_after_dd(AE, DS, QS)

DS$DSSTDTC <- NULL
check_qs_qsdtc_after_dd(AE, DS, QS)

AE1 <- data.frame(USUBJID = 1,
                  AEDTHDTC = "",
                  AESTDTC = c("2015-11-01", "2016-02-01"),
                  AEDECOD = "Rash",
                  AETERM = "RASH",
                  stringsAsFactors = FALSE)

DS1 <- data.frame(USUBJID = 1,
                  DSSTDTC = "2016-01",
                  DSCAT = c("DISPOSITION EVENT", "OTHER"),
                  DSSCAT = c('STUDY COMPLETION/EARLY DISCONTINUATION', ''),
                  DSDECOD = "DEATH",
                  DSTERM = c("DEATH", "DEATH DUE TO PROGRESSIVE DISEASE"),
                  stringsAsFactors = FALSE)

QS1 <- data.frame(USUBJID = 1,
                  QSDTC   = c("2015-06-30", "2016-01-15", "2016-01-15"),
                  QSCAT   = rep("EQ-5D-5L"),
                  QSORRES = "1",
                  QSSTAT  = "",
                  VISIT  =  c("Week 1", "Week 12", "Week 12"),
                  QSSTRESC = "1",
                  stringsAsFactors = FALSE)

check_qs_qsdtc_after_dd(AE=AE1, DS=DS1, QS=QS1)

AE1$AEDTHDTC[1:2] <- "2015-07-01"
check_qs_qsdtc_after_dd(AE=AE1, DS=DS1, QS=QS1)
```

---

check_qs_qsdtc_visit_ordinal_error

Check that all QS dates are duplicated or earlier than last visit's (possible datetime data entry error)

Description

This check identifies QSDTC values that are duplicated or earlier than last visit's. Unscheduled visits are excluded.

Usage

```
check_qs_qsdtc_visit_ordinal_error(QS)
```

Arguments

QS	SDTM dataset with variables USUBJID, QSCAT, QSORRES, VISITNUM, VISIT, QSDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Simon Luo

Examples

```
# no case
QS1 <- data.frame(USUBJID = c(rep(101, 5), rep(102, 5)),
                  QSCAT = "DLQI",
                  QSDTC = rep(c("2017-01-01T08:25", "2017-01-05T09:25",
                   "2017-01-15T10:25","2017-01-20T08:25","2017-01-25T08:25"), 2),
                  VISITNUM = rep(1:5,2),
             VISIT = rep(c( "Visit 1", "Visit 2", "Visit 3", "UNSCheduled!!!","VIsit 5"), 2),
                  stringsAsFactors = FALSE)

QS2 = QS1
QS2$QSCAT = "SKINDEX-29"

QS <- rbind(QS1, QS2)
check_qs_qsdtc_visit_ordinal_error(QS)

# adding cases with earlier date
QS$QSDTC[QS$USUBJID == 101 & QS$VISIT == "Visit 3"] <- "2017-01-10T08:25"
QS$QSDTC[QS$USUBJID == 102 & QS$VISIT == "Visit 2"] <- "2017-01-01T06:25"
check_qs_qsdtc_visit_ordinal_error(QS)

# adding cases with duplicated date
QS$QSDTC[QS$USUBJID == 102 & QS$VISIT == "Visit 3"] <- "2017-01-01T06:25"
 check_qs_qsdtc_visit_ordinal_error(QS)
```

check_qs_qsstat_qsreasnd *Check to confirm that there is a reason for a questionnaire being marked as not done*

Description

This code flags when QSSTAT Completion Status is marked as NOT DONE but QSREASND Reason Not Performed is not populated. Some but not all questionnaires in a study may collect Reason Not Performed information, so there may be instances of false positives in which no data correction is required. While QSREASND is a permissible variable, this scenario will be flagged in P21.

Usage

```
check_qs_qsstat_qsreasnd(QS)
```

Arguments

QS	Questionnaire SDTMv dataset with USUBJID, QSCAT, QSDTC, QSSTAT, QSREASND, VISIT (optional) variables

Value

boolean value if check returns 0 obs, otherwise return subset dataframe.

Author(s)

Katie Patel, Bonita Viegas Monteiro, Tom Stone (HackR 2021 Team WeRawesome)

Examples

```
QS <- data.frame(USUBJID = c(1,1,1,2,2,2),
                 QSDTC   = c("2015-06-30", "2015-09-30", "2015-12-30",
                             "2015-06-30", "2015-09-30", "2015-12-30"),
                 QSCAT   = "A",
               VISIT  =  c("Week 1", "Week 12", "Week 24", "Week 1", "Week 12", "Week 24"),
                 QSSTAT  = c("Not Done","NOT DONE","not done", rep("",3)),
                 QSREASND = c("Reasons",rep("",5)),
                 stringsAsFactors = FALSE)

 check_qs_qsstat_qsreasnd(QS)

 QS$QSSTAT=NULL

 check_qs_qsstat_qsreasnd(QS)
```

check_qs_qsstat_qsstresc *Check for non-missing QSSTRESC if QSSTAT is NOT DONE*

Description

This check is for studies with PRO outcomes data (i.e., QS domain), check that within a given instrument (e.g., QS.QSCAT='BFI' or QS.QSCAT ='MDASI"), if QS.QSSTAT=NOT DONE and QSTESTCD=QSALL, then there should be no populated responses(QS.QSSTRESC) for a particular visit (QS.VISIT), return a dataframe if otherwise

Usage

```
check_qs_qsstat_qsstresc(QS)
```

Arguments

QS	Questionnaires SDTM dataset with variables USUBJID, QSSTRESC, VISIT, QSSTAT, QSCAT, QSDTC, QSTESTCD

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
QS <- data.frame(
STUDYID = 1,
USUBJID = c(rep(1,6),rep(2,6)),
QSSTRESC = 1:12,
VISIT = c(rep(1,3),rep(2,3),rep(1,3),rep(2,3)),
QSSTAT = rep(c("DONE","NOT DONE"),6),
QSCAT = rep(c("INDIVIDUAL","OVERALL","BFI"),4),
QSDTC = "2016-01-01",
QSTESTCD = "QSALL",
stringsAsFactors = FALSE
)

check_qs_qsstat_qsstresc(QS)

QS$QSSTRESC[4]=" "
QS$QSSTRESC[6]=NA
QS$QSSTRESC[8]="."
check_qs_qsstat_qsstresc(QS)
```

```
QS$QSSTRESC=NULL
check_qs_qsstat_qsstresc(QS)
```

---

check_rs_rscat_rsscat *Check for patients with populated RSSCAT but missing RSCAT.*

Description

Check for patients with populated RSSCAT but missing RSCAT in RS domain to help flag a potential mapping issue for SPA; this does not warrant a query in Rave.

Usage

```
check_rs_rscat_rsscat(RS)
```

Arguments

RS	Response SDTM dataset with variables USUBJID, RSCAT and RSSCAT.

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Saibah Chohan, Ashley Mao, Tina Cho (HackR 2021 Team STA-R)

Examples

```
RS <- data.frame(
 USUBJID = c("id1", "id1", "id2", "id2", "id3"),
 RSCAT = c("A", "A", "B", NA, NA),
 RSSCAT = c("AA", "AA", "BB", "BB","AA"))
check_rs_rscat_rsscat(RS)

# Test with missing RSCAT
RS$RSCAT = NULL
check_rs_rscat_rsscat(RS)
```

check_rs_rsdtc_across_visit *Check RS records where the same date occurs across multiple visits*

Description

This check identifies records where the same date RSDTC occurs across multiple visits. Only applies to assessments by investigator, selected based on uppercased RSEVAL = "INVESTIGATOR" or missing or RSEVAL variable does not exist.

Usage

```
check_rs_rsdtc_across_visit(RS, preproc = identity, ...)
```

Arguments

RS	Disease Response SDTM dataset with variables USUBJID, RSDTC, VISIT, RSEVAL (optional)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Will Harris

Examples

```
# example that will be flagged
RS <- data.frame(
USUBJID = 1,
RSDTC = c(rep("2016-01-01",3), rep("2016-06-01",5), rep("2016-06-24",2)),
VISIT = c(rep("C1D1",3), rep("C1D2",3), rep("C2D1",4)),
RSSPID = "FORMNAME-R:13/L:13XXXX",
stringsAsFactors=FALSE)

check_rs_rsdtc_across_visit(RS)
check_rs_rsdtc_across_visit(RS, preproc=roche_derive_rave_row)

# example that will not be flagged because not Investigator
RS0 <- RS
RS0$RSEVAL <- "INDEPENDENT ASSESSOR"
check_rs_rsdtc_across_visit(RS0)
check_rs_rsdtc_across_visit(RS0, preproc=roche_derive_rave_row)
```

```
# example with log line differences in Rave form with records flagged
RS1 <- RS
RS1$RSSPID = c(rep("FORMNAME-R:13/L:13XXXX",4),
rep("FORMNAME-R:13/L:14XXXX",2),
rep("FORMNAME-R:03/L:13XXXX",2),
 rep("FORMNAME-R:9/L:13XXXX", 2))

check_rs_rsdtc_across_visit(RS1)
check_rs_rsdtc_across_visit(RS1, preproc=roche_derive_rave_row)

# example with RSTESTCD with records flagged
RS2 <- RS1
RS2$RSTESTCD = c(rep("OVRLRESP", 2), rep("OTHER", 2),
   rep("OVRLRESP", 2), rep("OTHER", 2), rep("OVRLRESP", 2))
check_rs_rsdtc_across_visit(RS2)
check_rs_rsdtc_across_visit(RS2, preproc=roche_derive_rave_row)

# example with records flagged without xxSPID
RS3 <- RS
RS3$RSSPID <- NULL
check_rs_rsdtc_across_visit(RS3)
check_rs_rsdtc_across_visit(RS3, preproc=roche_derive_rave_row)

# example without required variable
RS4 <- RS
RS4$VISIT <- NULL
check_rs_rsdtc_across_visit(RS4)
check_rs_rsdtc_across_visit(RS4, preproc=roche_derive_rave_row)
```

`check_rs_rsdtc_visit` *Check missing RSDTC and VISIT*

Description

This check looks for missing RSDTC or VISIT values when RSORRES is not missing and RSSTAT not equal to "NOT DONE" in RS dataset and returns a data frame. Only applies to assessments by investigator.

Usage

```
check_rs_rsdtc_visit(RS)
```

Arguments

RS	Disease Response SDTM dataset with variables USUBJID, RSDTC, RSORRES, VISIT, RSSTAT

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Peggy Wen, Stella Banjo (HackR 2021)

Examples

```
RS <- data.frame(
USUBJID = 1:10,
RSDTC = 1:10,
RSORRES = "THING",
VISIT = "C1D1",
RSSTAT = 1:10,
RSEVAL = c("NA","","IRF","investigator",rep("INVESTIGATOR",6)),
stringsAsFactors=FALSE
)

RS$RSDTC[1]=""
RS$RSDTC[2]="NA"
RS$RSDTC[3]=NA
RS$VISIT[3]=""
RS$VISIT[4]="NA"
RS$VISIT[5]=NA
check_rs_rsdtc_visit(RS)

RS$RSORRES[1]=""
check_rs_rsdtc_visit(RS)

RS$RSORRES[4] = "THING 1"
RS$RSORRES[5] = "THING 2"

check_rs_rsdtc_visit(RS)
```

---

check_rs_rsdtc_visit_ordinal_error

Check that all RS dates for INV Overall Response are duplicated or earlier than last visit's (possible date entry error)

Description

This check identifies RSDTC values when RSEVAL == 'INVESTIGATOR' and RSTESTCD == 'OVRLRESP' that are duplicated or earlier than last visit's. Unscheduled and 'NOT DONE' visits are excluded.

Usage

```
check_rs_rsdtc_visit_ordinal_error(RS)
```

Arguments

RS	Response SDTM dataset with variables USUBJID, VISITNUM, VISIT, RSDTC, RSTESTCD, RSEVAL, RSSTAT

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

James Zhang

Examples

```
# no cases
RS<- data.frame(USUBJID = 101:102,
                RSDTC=rep(c("2017-01-01T08:25", "2017-01-05T09:25",
                       "2017-01-15T10:25","2017-01-20T08:25","2017-01-25T08:25"), 2),
                VISITNUM=rep(1:5,2),
                VISIT=rep(c("Screening", "Cycle 1", "Cycle 2","Cycle 3","Follow-up"),2),
                RSTESTCD="OVRLRESP",
                RSEVAL="INVESTIGATOR",
                RSSTAT="",
                stringsAsFactors=FALSE)
check_rs_rsdtc_visit_ordinal_error(RS)

# adding cases with earler date
RS$RSDTC[RS$USUBJID == 101 & RS$VISIT == "Cycle 3"] <- "2017-01-02T08:25"
RS$RSDTC[RS$USUBJID == 102 & RS$VISIT == "Cycle 1"] <- "2017-01-01T06:25"
check_rs_rsdtc_visit_ordinal_error(RS)
```

---

check_sc_dm_eligcrit *Check SC Eye Meeting Eligibility Criteria assignments among DM patients*

Description

Check if SC.SCCAT = "STUDY EYE SELECTION" and SC.SCTESTCD = "ELIGEYE", then SC.SCORRES should have "OS", "OD", or "OU" values. Flag if subject is in DM and without an associated SC.SCORRES value or the ELIGEYE Eye Meeting Eligibility Criteria value is not "OS", "OD", or "OU".

Usage

```
check_sc_dm_eligcrit(DM, SC)
```

Arguments

DM	Subject Demographics SDTM dataset with variable USUBJID
SC	Subject Characteristics SDTM dataset for Ophtho Study with variables USUBJID, SCTESTCD, SCTEST, SCCAT, SCORRES, SCDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah (HackR 2021 Team Eye)

See Also

Other OPHTH: check_ae_aelat(), check_cm_cmlat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_e
check_oe_bcva_4m_late_early_tot(), check_oe_bcva_4m_vs_1m_req(), check_oe_bcva_tot_mismatch(),
check_oe_sc_lat_count_fingers(), check_pr_prlat(), check_sc_dm_seyeselc()

Examples

```
dm <- data.frame(USUBJID = c(1,2))
sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                 SCTEST   = c("Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " ",
                              "Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest", " "),
                 SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", ""),
                 SCCAT    = c("STUDY EYE SELECTION",
                              "STUDY EYE SELECTION",
                              "",
                              "STUDY EYE SELECTION",
                              "STUDY EYE SELECTION",
                              ""),
                 SCORRES  = c("OS", "OS", "", "", "OU", ""),
                 SCDTC    = rep("2021-01-01", 6),
                 stringsAsFactors = FALSE)

check_sc_dm_eligcrit(SC=sc, DM=dm)

dm <- data.frame(USUBJID = c(1,2,3,4))
sc$SCORRES[4] = "OS"
check_sc_dm_eligcrit(SC=sc, DM=dm)

sc$SCORRES <- NULL
check_sc_dm_eligcrit(SC=sc, DM=dm)
```

check_sc_dm_seyeselc *Check SC Study Eye Selection assignments among DM patients*

Description

Check if SC.SCCAT = "STUDY EYE SELECTION" and SC.SCTESTCD = "FOCID", then SC.SCORRES should have "OS", "OD", or "OU" values. Flag if subject is in DM and without an associated SC.SCORRES value or the STUDY EYE SELECTION value is not "OS", "OD", or "OU".

Usage

```
check_sc_dm_seyeselc(DM, SC)
```

Arguments

DM	Subject Demographics SDTM dataset with variable USUBJID
SC	Subject Characteristics SDTM dataset for Ophtho Study with variables USUBJID, SCTESTCD, SCTEST, SCCAT, SCORRES, SCDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Monarch Shah (HackR 2021 Team Eye)

See Also

Other OPHTH: check_ae_aelat(), check_cm_cmlat(), check_cm_cmlat_prior_ocular(), check_oe_bcva_1m_late_e
check_oe_bcva_4m_late_early_tot(), check_oe_bcva_4m_vs_1m_req(), check_oe_bcva_tot_mismatch(),
check_oe_sc_lat_count_fingers(), check_pr_prlat(), check_sc_dm_eligcrit()

Examples

```
dm <- data.frame(USUBJID = c(1,2))
sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                 SCTEST   = c("Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " ",
                              "Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest", " "),
                 SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", ""),
                 SCCAT    = c("STUDY EYE SELECTION",
                              "STUDY EYE SELECTION",
                              "",
```

```
                              "STUDY EYE SELECTION",
                              "STUDY EYE SELECTION", ""),
                  SCORRES  = c("LEFT", "OS", "", "RIGHT", "OD", ""),
                  SCDTC    = rep("2021-01-01", 6),
                  stringsAsFactors = FALSE)

check_sc_dm_seyeselc(SC=sc, DM=dm)

dm <- data.frame(USUBJID = c(1,2,3,4))
sc <- data.frame(USUBJID  = c(1,1,1,2,2,2),
                  SCTEST   = c("Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " ",
                              "Eye Meeting Eligibility Criteria",
                              "Focus of Study-Specific Interest",
                              " "),
                  SCTESTCD = c("ELIGEYE", "FOCID", "", "ELIGEYE", "FOCID", ""),
                  SCCAT    = c("STUDY EYE SELECTION",
                              "STUDY EYE SELECTION",
                              "",
                              "STUDY EYE SELECTION",
                              "STUDY EYE SELECTION", ""),
                  SCORRES  = c("LEFT", "OS", "", "RIGHT", "", ""),
                  SCDTC    = rep("2021-01-01", 6),
                  stringsAsFactors = FALSE)

check_sc_dm_seyeselc(SC=sc, DM=dm)
```

`check_ss_ssdtc_alive_dm` *Check non-missing last ALIVE status date in SS is before than death date in DM*

Description

This check looks for non-missing SS.SSDTC when SS.SSORRES contains 'ALIVE' and Subject Status Date/Time of Assessments is greater then Start Date/Time of Disposition Event(SS.SSDTC > DS.DSSTDTC)

Usage

```
check_ss_ssdtc_alive_dm(SS, DM)
```

Arguments

SS	Subject Status SDTM dataset with variables USUBJID, SSDTC, SSORRES, SSTESTCD, VISIT
DM	Demographics SDTM dataset with variables USUBJID, DTHDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Vira Vrakina

Examples

```
SS <- data.frame(
 USUBJID = 1:5,
 SSDTC = "2020-01-02",
 SSTESTCD = "SURVSTAT",
 SSORRES  = c("DEAD","DEAD","ALIVE","DEAD","ALIVE"),
 VISIT = "WEEK 4"
)


DM <- data.frame(
 USUBJID = 1:5,
 DTHDTC = "2020-01-03"
)

check_ss_ssdtc_alive_dm(SS, DM)

SS <- data.frame(
 USUBJID = 1:5,
 SSDTC = "2020-01-04",
 SSTESTCD = "SURVSTAT",
 SSORRES  = c("DEAD","DEAD","ALIVE","DEAD","ALIVE"),
 VISIT = "WEEK 4"
)

DM <- data.frame(
 USUBJID = 1:5,
 DTHDTC = c("2020-01-04", "2020-01-05", "2020-01-03", "2020-01-04", "2020-01-05")
)

check_ss_ssdtc_alive_dm(SS, DM)
```

---

check_ss_ssdtc_dead_ds

Check non-missing DEAD status date in SS and non-missing according DS record with death date where status date is greater or equal to death date

---

Description

This check looks for missing death date in DS dataset if there is DEAD status date in SS dataset or if Subject Status Date/Time of Assessments is less than Start Date/Time of Disposition Event(SS.SSDTC < DS.DSSTDTC)

Usage

```
check_ss_ssdtc_dead_ds(SS, DS, preproc = identity, ...)
```

Arguments

SS	Subject Status SDTM dataset with variables USUBJID, SSDTC, SSSTRESC, VISIT
DS	Disposition SDTM dataset with variables USUBJID, DSSTDTC, DSDECOD, DSCAT
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Vira Vrakina

Examples

```
SS <- data.frame(
 USUBJID = 1:5,
 SSDTC = "2020-01-02",
 SSSTRESC  = c("DEAD","DEAD","ALIVE","DEAD","ALIVE"),
 VISIT = "FOLLOW-UP",
 SSSPID = "FORMNAME-R:13/L:13XXXX"
)

DS <- data.frame(
 USUBJID = 1:5,
 DSSTDTC  = c("2020-01-02","2020-01-02", "2020-01-01", "2020-01-03", "2020-01-01"),
 DSDECOD = c(rep('DEATH', 5)),
 DSSPID = "FORMNAME-R:13/L:13XXXX",
 DSCAT   = c("OTHER EVENT", rep("DISPOSITION EVENT", 4))
)

check_ss_ssdtc_dead_ds(SS, DS)
check_ss_ssdtc_dead_ds(SS, DS, preproc=roche_derive_rave_row)


SS <- data.frame(
 USUBJID = 1:5,
```

```
 SSDTC = "2020-01-02",
 SSSTRESC  = c( rep("DEAD", 5)),
 VISIT = "FOLLOW-UP",
 SSSPID = "FORMNAME-R:13/L:13XXXX"
)

DS <- data.frame(
 USUBJID = 1:5,
 DSSTDTC  = c("2020-01-02","2020-01-02", "2020-01-01", "2020-01-03", "2020-01-01"),
 DSDECOD = c(rep('DEATH', 5)),
 DSSPID = "FORMNAME-R:13/L:13XXXX",
 DSCAT   = c(rep("DISPOSITION EVENT", 5))
)

check_ss_ssdtc_dead_ds(SS, DS)
check_ss_ssdtc_dead_ds(SS, DS, preproc=roche_derive_rave_row)


SS <- data.frame(
 USUBJID = 1:5,
 SSDTC = "2020-01-02",
 SSSTRESC  = c(rep("DEAD", 5)),
 VISIT = "FOLLOW-UP",
 SSSPID = "FORMNAME-R:13/L:13XXXX"
)

DS <- data.frame(
 USUBJID = 1:5,
 DSSTDTC  = 2,
 DSDECOD = c(rep('DEATH', 5)),
 DSSPID = "FORMNAME-R:13/L:13XXXX",
 DSCAT   = c(rep("DISPOSITION EVENT", 5))
)

check_ss_ssdtc_dead_ds(SS, DS)
```

`check_ss_ssdtc_dead_dthdtc` *Check non-missing DEAD status date in SS and an according DM record with death date where status date is greater or equal to death date*

Description

This check looks for non-missing SS.SSDTC when SS.SSSTRESC='DEAD' and Subject Status Date/Time of Assessments is less than Start Date/Time of Disposition Event(SS.SSDTC < DS.DSSTDTC)

Usage

```
check_ss_ssdtc_dead_dthdtc(SS, DM)
```

Arguments

SS	Subject Status SDTM dataset with variables USUBJID, SSDTC, SSSTRESC, VISIT
DM	Demographics SDTM dataset with variables USUBJID, DTHDTC

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Vira Vrakina

Examples

```
SS <- data.frame(
 USUBJID = 1:5,
 SSDTC = "2020-01-02",
 SSSTRESC  = c("DEAD","DEAD","ALIVE","DEAD","ALIVE"),
 VISIT = "DAY 10"
)

DM <- data.frame(
 USUBJID = 1:5,
 DTHDTC = "2020-01-02"
)

check_ss_ssdtc_dead_dthdtc(SS, DM)

SS <- data.frame(
 USUBJID = 1:5,
 SSDTC = "2020-01-02",
 SSSTRESC  = c("DEAD","DEAD","ALIVE","DEAD","ALIVE"),
 VISIT = "FOLLOW-UP"
)

DM <- data.frame(
 USUBJID = 1:5,
 DTHDTC = c("2020-01-01","2020-01-02","2020-01-03","2020-01-04","2020-01-02")
)

check_ss_ssdtc_dead_dthdtc(SS, DM)
```

check_ss_ssstat_ssorres *Check for non-missing SSORRES if SSSTAT is NOT DONE*

Description

This check is for studies with LTFU mapped to the SS domain, check that if 'NOT DONE' (Unable to Contact), then there should not be a response (SSORRES)

Usage

```
check_ss_ssstat_ssorres(SS, preproc = identity, ...)
```

Arguments

SS	Long-Term Survival Follow-Up SDTM dataset with variables USUBJID, VISIT, SSSTAT, SSDTC, SSORRES, SSSPID (optional)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
SS <- data.frame(
STUDYID = 1,
USUBJID = c(rep(1,6),rep(2,6)),
SSSTRESC = c("ALIVE", "DEAD", "ALIVE", "", "", "U"),
SSORRES = c("ALIVE", "DEAD", "ALIVE", "", "", "U"),
VISIT = rep(c("SURVIVAL FOLLOW UP 3 MONTHS"),6),
SSSTAT = rep(c("","NOT DONE"),6),
SSDTC = "2016-01-01",
SSSPID = "",
stringsAsFactors = FALSE
)

check_ss_ssstat_ssorres(SS)

SS$SSORRES[2]=NA
check_ss_ssstat_ssorres(SS)

SS$SSSPID="FORMNAME-R:5/L:5XXXX"
```

```
check_ss_ssstat_ssorres(SS,preproc=roche_derive_rave_row)

SS$SSORRES[6]=NA
SS$SSORRES[8]=""
SS$SSORRES[12]=NA
check_ss_ssstat_ssorres(SS)

SS$SSORRES=NULL
check_ss_ssstat_ssorres(SS)
```

---

check_tr_dup *Check for duplicate TR records*

Description

This check looks for duplicate TR records and returns a data frame. Only applies to assessments by Investigator, selected based on uppercased TREVAL = "INVESTIGATOR" or missing or TREVAL variable does not exist.

Usage

```
check_tr_dup(TR)
```

Arguments

TR	dataframe with variables USUBJID, TRCAT, TRLINKID/TRLNKID, TRTESTCD, TRSTRESC, TRDTC, TRSPID (if it exists)

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Joel Laxamana

Examples

```
# example with an error
TR <- data.frame(
 USUBJID  = c(1,1,2,2),
 TRCAT    = c(1,1,2,2),
 TRTESTCD = c(1,1,2,2),
 TRLINKID = c(1,1,2,2),
 TRDTC = c(rep("2016-01-01",2), rep("2016-06-01",2)),
 TRSTRESC = c(1,1,2,2),
 TRSPID = "FORMNAME-R:19/L:19XXXX",
```

```
 TREVAL = "INVESTIGATOR",
 stringsAsFactors = FALSE
)

check_tr_dup(TR)

TR1 <- TR
TR1$TRSPID <- NULL

check_tr_dup(TR1)

TR2 <- TR
TR2$TREVAL <- NULL

check_tr_dup(TR2)

# example with no records flagged because issues only among IRF records
TR3 <- TR
TR3$TREVAL <- "INDEPENDENT ASSESSOR"
check_tr_dup(TR3)

# example with required variable missing
TR4 <- TR
TR4$TRLINKID <- NULL
check_tr_dup(TR4)
```

`check_tr_trdtc_across_visit` *Check TR Longest Diameter Records where the same date occurs across multiple visits*

Description

This check identifies records where the same date TRDTC occurs across multiple visits for Longest Diameter measurements (TRTESTCD is "LDIAM"). Only applies to assessments by investigator, selected based on uppercased TREVAL = "INVESTIGATOR" or missing or TREVAL variable does not exist.

Usage

```
check_tr_trdtc_across_visit(TR, preproc = identity, ...)
```

Arguments

`TR`	Tumor Result SDTM dataset with variables USUBJID, TRDTC, TRTESTCD, VISIT, TREVAL (optional)
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Will Harris

Examples

```
TR <- data.frame(
USUBJID = 1,
TRDTC = c(rep("2016-01-01",3), rep("2016-06-01",5), rep("2016-06-24",2)),
VISIT = c(rep("C1D1",3), rep("C1D2",3), rep("C2D1",4)),
TRTESTCD = c(rep("LDIAM",7),rep("SAXIS",3)),
TRSPID = "FORMNAME-R:13/L:13XXXX",
stringsAsFactors=FALSE)

check_tr_trdtc_across_visit(TR)
check_tr_trdtc_across_visit(TR, preproc=roche_derive_rave_row)

TR2 <- TR
TR2$TRSPID[4:5] <- c("FORMNAME2-R:5/L:13XXXX", "FORMNAME3-R:0/L:13XXXX")

check_tr_trdtc_across_visit(TR2)
check_tr_trdtc_across_visit(TR2, preproc=roche_derive_rave_row)

# missing optional variable
TR3 <- TR
TR3$TRSPID <- NULL

check_tr_trdtc_across_visit(TR3)
check_tr_trdtc_across_visit(TR3, preproc=roche_derive_rave_row)

# missing required variable
TR4 <- TR
TR4$TRTESTCD <- NULL

check_tr_trdtc_across_visit(TR4)
check_tr_trdtc_across_visit(TR4, preproc=roche_derive_rave_row)
```

---

check_tr_trdtc_visit_ordinal_error

Check that all TR dates by INV are duplicated or earlier than last visit's (possible date entry error)

Description

This check identifies TRDTC values when TREVAL == 'INVESTIGATOR' are duplicated or earlier than last visit's. Unscheduled and 'NOT DONE' visits are excluded.

Usage

```
check_tr_trdtc_visit_ordinal_error(TR)
```

Arguments

TR	Tumor Response Measurement SDTM dataset with variables USUBJID, VISITNUM, VISIT, TRDTC, TREVAL, TRSTAT

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

James Zhang

Examples

```
# no case
TR<- data.frame(USUBJID = 101:102,
                TRSEQ=rep(1:5,2),
                TRDTC=rep(c("2017-01-01T08:25", "2017-01-05T09:25",
                   "2017-01-15T10:25","2017-01-20T08:25","2017-01-25T08:25"), 2),
                VISITNUM=rep(1:5,2),
                VISIT=rep(c("Screening", "Cycle 1", "Cycle 2","Cycle 3","Follow-up"),2),
                TREVAL="INVESTIGATOR",
                TRSTAT="",
                stringsAsFactors=FALSE)
check_tr_trdtc_visit_ordinal_error(TR)

# Cases with earler datetime
TR$TRDTC[TR$USUBJID == 101 & TR$VISIT == "Cycle 3"] <- "2017-01-02T08:25"
TR$TRDTC[TR$USUBJID == 102 & TR$VISIT == "Cycle 1"] <- "2017-01-01T06:25"
check_tr_trdtc_visit_ordinal_error(TR)
```

---

check_tr_trstresn_ldiam

Check for TR records with missing TRSTRESN for Longest Diameter (LDIAM)

Description

This checks looks for TR records with missing values in numeric result/finding for the Longest Diameter (TRTESTCD is LDIAM) tumor measurement. Only applies to assessments by investigator, selected based on uppercased TREVAL = "INVESTIGATOR" or missing or TREVAL variable does not exist.

Usage

```
check_tr_trstresn_ldiam(TR, preproc = identity, ...)
```

Arguments

TR	Tumor Results SDTM dataset with variables USUBJID, TRTESTCD, TRLINKID/TRLNKID, TRDTC, VISIT, TRORRES, TRSTRESN, TREVAL (optional), TRSTAT (optional), TRSPID (optional)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Will Harris

Examples

```
TR <- data.frame(USUBJID = 1:5,
                 TRTESTCD = c("OTHER", rep("LDIAM", 4)),
                 TRLINKID = 1:5,
                 TRDTC = 1:5,
                 VISIT = LETTERS[1:5],
                 TRORRES = LETTERS[1:5],
                 TRSTRESN = 1:5,
                 TRSTAT = "",
                 TREVAL = "INVESTIGATOR",
                 TRSPID = "FORMNAME-R:19/L:19XXXX",
                 stringsAsFactors = FALSE)

check_tr_trstresn_ldiam(TR)

TR1 <- TR
TR1$TRSTAT <- NULL
TR1$TREVAL <- NULL
TR1$TRSPID <- NULL

check_tr_trstresn_ldiam(TR1)

TR2 <- TR
TR2$TRSTRESN <- c("", "NA", NA, 1, 1)

check_tr_trstresn_ldiam(TR2)
check_tr_trstresn_ldiam(TR2,preproc=roche_derive_rave_row)
```

check_ts_aedict *Check for missing MedDRA version in TS*

Description

This check looks for missing MedDRA version; if it's present, also checking it's the current version

Usage

```
check_ts_aedict(TS)
```

Arguments

TS	Trial Summary SDTM dataset with variables TSPARMCD and TSVAL

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Vira Vrakina, Antony Howard (HackR 2021 Team Pentraxin1)

Examples

```
TS1 <- data.frame(
 STUDYID = 1,
 TSPARMCD = "AEDICT",
 TSVAL = "MedDRA 22.0",
 TSVAL2 = ""
)

TS2 <- data.frame(
 STUDYID = 2,
 TSPARMCD = "AEDICT",
 TSVAL = "",
 TSVAL1 = "meddra v22.0"
)

TS3 <- data.frame(
 STUDYID = 3,
 TSPARMCD = "AEDICT",
 TSVAL = ""
)

TS4 <-data.frame(
 STUDYID = 4,
 TSPARMCD = "CMDICT",
 TSVAL = ""
)
```

```
TS5 <- data.frame(
    STUDYID = 1,
    TSPARMCD = "AEDICT",
    TSVAL = "meddra 24.0",
    TSVAL2 = ""
)

TS6 <- data.frame(
    STUDYID = 1,
    TSPARMCD = "AEDICT",
    TSVAL = "    meddra    23.0   ",
    TSVAL2 = ""
)

check_ts_aedict(TS1)
check_ts_aedict(TS2)
check_ts_aedict(TS3)
check_ts_aedict(TS4)
check_ts_aedict(TS5)
check_ts_aedict(TS6)
check_ts_aedict(rbind(TS1,TS1))
```

`check_ts_cmdict` *Check for missing WHODrug version in TS*

Description

This check looks for missing WHODrug version; if it's present, also checking it's the current version

Usage

```
check_ts_cmdict(TS)
```

Arguments

TS	Trial Summary SDTM dataset with variables TSPARMCD and TSVAL

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Antony Howard (HackR 2021 Team Pentraxin1)

Examples

```
TS1 <- data.frame(
 STUDYID = 1,
 TSPARMCD = "CMDICT",
 TSVAL = "WHODRUG GLOBAL B3 MARCH 1, 2021",
 TSVAL2 = ""
)

TS2 <- data.frame(
 STUDYID = 2,
 TSPARMCD = "CMDICT",
 TSVAL = "",
 TSVAL1 = "WHODRUG GLOBAL B3 MARCH 1, 2021"
)

TS3 <- data.frame(
 STUDYID = 3,
 TSPARMCD = "CMDICT",
 TSVAL = ""
)

TS4 <-data.frame(
 STUDYID = 4,
 TSPARMCD = "AEDICT",
 TSVAL = ""
)

TS5 <- data.frame(
    STUDYID = 5,
    TSPARMCD = "CMDICT",
    TSVAL = "meddra 24.0",
    TSVAL2 = ""
)

TS6 <- data.frame(
 STUDYID = 6,
 TSPARMCD = "CMDICT",
 TSVAL = "WHODRUG vGLOBAL B3 MARCH 1, 2021",
 TSVAL2 = ""
)

check_ts_cmdict(TS1)
check_ts_cmdict(TS2)
check_ts_cmdict(TS3)
check_ts_cmdict(TS4)
check_ts_cmdict(TS5)
check_ts_cmdict(TS6)
check_ts_cmdict(rbind(TS1,TS1))
```

`check_ts_sstdtc_ds_consent` *Check for missing SSTDTC (Study Start Date) in TS*

Description

This check looks for missing SSTDTC (Study Start Date) in TS; if it's present, check that the date matches the earliest informed consent among any subject enrolled in the study. The FDA Technical Rejection Criteria for Study Data - effective September 15, 2021 requires Study Start Date (https://www.fda.gov/media/100743/download). If missing, no data queries are needed - this would be updating the assignment in the TS domain.

Usage

```
check_ts_sstdtc_ds_consent(DS, TS)
```

Arguments

DS	Disposition SDTM dataset with variables DSCAT, DSSCAT, DSDECOD, DSSTDTC
TS	Trial Summary SDTM dataset with variables TSPARMCD, TSPARM, TSVAL

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
TS1 <- data.frame(
 STUDYID = 1,
 TSPARMCD = "SSTDTC",
 TSPARM = "Study Start Date",
 TSVAL = "2017-01-01",
 TSVAL1 = "",
 TSVAL2 = ""
)

TS2 <- data.frame(
 STUDYID = 2,
 TSPARMCD = "AEDICT",
 TSPARM = "Study Start Date",
 TSVAL = "MedDRA v23.0",
 TSVAL1 = ""
)

TS3 <- data.frame(
```

```
 STUDYID = 3,
 TSPARMCD = "SSTDTC",
 TSPARM = "Study Start Date",
 TSVAL = ""
)

TS4 <- data.frame(
 STUDYID = 1,
 TSPARMCD = "SSTDTC",
 TSPARM = "Study Start Date",
 TSVAL = "2020-01-02",
 TSVAL1 = "",
 TSVAL2 = ""
)

TS5 = rbind(TS1, TS4)

TS6 <- data.frame(
 STUDYID = 1,
 TSPARMCD = "SSTDTC",
 TSPARM = "Study Start Date",
 TSVAL = "2020-01",
 TSVAL1 = "",
 TSVAL2 = ""
)

DS1 <- data.frame(
 USUBJID = c(1,1,2,3,4),
 DSCAT   = rep("PROTOCOL MILESTONE", 5),
 DSSCAT   = rep("PROTOCOL MILESTONE", 5),
 DSDECOD = c("INFORMED CONSENT OBTAINED", "OTHER", "PHYSICIAN DECISION",
             "OTHER", "INFORMED CONSENT OBTAINED"),
 DSSTDTC = c("2021-01-01", "2021-01-02", "2021-01-02", "2021-01-02", "2020-01-02"),
 stringsAsFactors = FALSE
)

check_ts_sstdtc_ds_consent(DS=DS1, TS=TS1)
check_ts_sstdtc_ds_consent(DS=DS1, TS=TS2)
check_ts_sstdtc_ds_consent(DS=DS1, TS=TS3)
check_ts_sstdtc_ds_consent(DS=DS1, TS=TS4)
check_ts_sstdtc_ds_consent(DS=DS1, TS=TS5)
check_ts_sstdtc_ds_consent(DS=DS1, TS=TS6)
```

---

check_tu_rs_new_lesions

Check for consistency between new lesions and overall PD response

---

Description

This checks for patients with new lesions in TU (TUSTRESC=='NEW') but no Overall Response assessment of PD (Disease Progression) or PMD (Progressive Metabolic Disease) in RS (i.e., (RSTESTCD=='OVRLRESP' and RSSTRESC %in% c('PD','PMD'))). Only applies to assessments by investigator, if TUEVAL and RSEVAL variables available.

Usage

```
check_tu_rs_new_lesions(RS, TU)
```

Arguments

RS	Response SDTM dataset with variables USUBJID, RSSTRESC, RSTESTCD
TU	Tumor Identification SDTM dataset with variables USUBJID, TUSTRESC, TUDTC

Value

TRUE if check passed and FALSE if check failed + 'msg' and 'data' attributes

Author(s)

Will Harris

Examples

```
TU <- data.frame(
 USUBJID = 1:3,
 TUSTRESC = c("INV001","NEW","NEW"),
 TUDTC = "2017-01-01"
)

RS <- data.frame(
 USUBJID = 1:2,
 RSSTRESC = c("SD","NE")
)

# required variable is missing
check_tu_rs_new_lesions(RS,TU)

RS$RSTESTCD = 'OVRLRESP'

# flag USUBJIDs with NEW
check_tu_rs_new_lesions(RS,TU)


RS$RSSTRESC[2] = "PD"

# flag USUBJID with NEW and without PD
check_tu_rs_new_lesions(RS,TU)

# Metabolic response in heme trials
```

```
RS$RSSTRESC[2] = "PMD"
check_tu_rs_new_lesions(RS,TU)


# pass when USUBJIDs with new have PD
RS <- data.frame(
 USUBJID = 1:3,
 RSSTRESC = c("SD","PD", "PD"),
 RSTESTCD = "OVRLRESP"
)

check_tu_rs_new_lesions(RS,TU)

TU$TUEVAL = "INDEPENDENT ASSESSOR"

RS$RSEVAL = "INDEPENDENT ASSESSOR"

## pass if by IRF, even if NEW in TU
check_tu_rs_new_lesions(RS,TU)

RS <- NULL

# required dataset missing
check_tu_rs_new_lesions(RS,TU)
```

`check_tu_tudtc` *Check for missing TUDTC values*

Description

This check looks for missing TUDTC values and returns a data frame. Only applies to assessments by investigator.

Usage

```
check_tu_tudtc(TU, preproc = identity, ...)
```

Arguments

`TU`	Tumor Identification SDTM dataset with variables USUBJID, TUDTC, VISIT, TUORRES, TUSPID (optional), TUTESTCD (optional)
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Beeya Na

Examples

```
TU <- data.frame(
USUBJID = "1001",
TUDTC = "2020-05-05",
VISIT = "C1D1",
TUORRES = 1:10,
TUSPID = "FORMNAME-R:19/L:19XXXX",
TUEVAL = "INVESTIGATOR",
TUTESTCD = "TUMIDENT",
stringsAsFactors = FALSE
)

TU$TUDTC[1]=""
TU$TUDTC[2]="NA"
TU$TUDTC[3]=NA

check_tu_tudtc(TU,preproc=roche_derive_rave_row)

TU$TUEVAL[1]=""
TU$TUTESTCD=NULL
check_tu_tudtc(TU,preproc=roche_derive_rave_row)

TU$TUEVAL[2]="INDEPENDENT ASSESSOR"
TU$TUEVAL[3]="INDEPENDENT ASSESSOR"
TU$TUDTC[4]=""
check_tu_tudtc(TU)

TU$TUSPID=NULL
check_tu_tudtc(TU)

TU$VISIT=NULL
check_tu_tudtc(TU)
```

---

check_tu_tudtc_across_visit

Check TU Records where the same date occurs across multiple visits

---

Description

This check identifies records where the same date TUDTC occurs across multiple visits. Only applies to assessments by investigator, selected based on uppercased TUEVAL = "INVESTIGATOR" or missing or TUEVAL variable does not exist.

Usage

```
check_tu_tudtc_across_visit(TU, preproc = identity, ...)
```

Arguments

TU	Tumor Identification SDTM dataset with variables USUBJID, TUDTC, VISIT, TUEVAL (optional), TUTESTCD (optional)
preproc	An optional company specific preprocessing script
...	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed.

Author(s)

Will Harris

Examples

```
# records flagged
TU <- data.frame(USUBJID = 1,
                 TUDTC = c(rep("2016-01-01",3), rep("2016-06-01",5), rep("2016-06-24",2)),
                  VISIT = c(rep("C1D1",3), rep("C1D2",3), rep("C2D1",4)),
                  TUSPID = "FORMNAME-R:13/L:13XXXX",
                  stringsAsFactors=FALSE)

check_tu_tudtc_across_visit(TU)
check_tu_tudtc_across_visit(TU, preproc=roche_derive_rave_row)

# no records flagged because non-Investigator results
TU2 <- TU
TU2$TUEVAL <- "INDEPENDENT ASSESSOR"

check_tu_tudtc_across_visit(TU2)
check_tu_tudtc_across_visit(TU2, preproc=roche_derive_rave_row)

# example with TUTESTCD and with records flagged
TU3 <- TU
TU3$TUTESTCD = c(rep("TUMIDENT", 2), rep("OTHER", 2),
   rep("TUMIDENT", 2), rep("OTHER", 2), rep("TUMIDENT", 2))
check_tu_tudtc_across_visit(TU3)
check_tu_tudtc_across_visit(TU3, preproc=roche_derive_rave_row)


# example without TUSPID and with records flagged
TU4 <- TU
TU4$TUSPID <- NULL

check_tu_tudtc_across_visit(TU4)
check_tu_tudtc_across_visit(TU4, preproc=roche_derive_rave_row)
```

```
# example with required variable missing
TU5 <- TU
TU5$VISIT <- NULL

check_tu_tudtc_across_visit(TU5)
check_tu_tudtc_across_visit(TU5, preproc=roche_derive_rave_row)
```

`check_tu_tudtc_visit_ordinal_error` *Check that all TU dates are duplicated or earlier than last visit's (possible datetime data entry error)*

Description

This check identifies TUDTC values that are duplicated or earlier than last visit's. Unscheduled visits are excluded.

Usage

```
check_tu_tudtc_visit_ordinal_error(TU)
```

Arguments

TU	Tumor Identification SDTM dataset with variables USUBJID,TUORRES ,TULOC, VISITNUM, VISIT, TUDTC, TUEVAL

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Jingyuan Chen

Examples

```
# no case
TU <- data.frame(USUBJID = 101:102,
                 TUORRES = rep(c("NEW", "TARGET"), 5),
                 TULOC=rep(c("BONE","LIVER"),5),
                 TUDTC = rep(c("2017-01-01T08:25", "2017-01-05T09:25",
                  "2017-01-15T10:25","2017-01-20T08:25","2017-01-25T08:25"), 2),
                 VISITNUM = rep(1:5,2),
                VISIT = rep(c( "Visit 1", "Visit 2", "Visit 3", "Visit 4","VIsit 5"), 2),
                 TUEVAL="INVESTIGATOR",
                 stringsAsFactors = FALSE)
check_tu_tudtc_visit_ordinal_error(TU)
```

```
# adding cases with earler date
TU$TUDTC[TU$USUBJID == 101 & TU$VISIT == "Visit 4"] <- "2017-01-10T08:25"
TU$TUDTC[TU$USUBJID == 102 & TU$VISIT == "Visit 2"] <- "2017-01-01T06:25"
check_tu_tudtc_visit_ordinal_error(TU)

# adding cases with duplicated date
TU$TUDTC[TU$USUBJID == 101 & TU$VISIT == "Visit 5"] <- "2017-01-10T08:25"
TU$TUDTC[TU$USUBJID == 102 & TU$VISIT == "Visit 3"] <- "2017-01-01T06:25"
 check_tu_tudtc_visit_ordinal_error(TU)
```

`check_tu_tuloc_missing` *Check for missing TULOC values*

Description

This check looks for target lesions with missing TULOC values and returns a data frame. Only applies to assessments by investigator.

Usage

```
check_tu_tuloc_missing(TU, preproc = identity, ...)
```

Arguments

`TU`	Tumor Identification SDTM dataset with variables USUBJID, TUDTC, VISIT, TUORRES, TULOC, TUSPID (optional)
`preproc`	An optional company specific preprocessing script
`...`	Other arguments passed to methods

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Will Harris

Examples

```
TU <- data.frame(
USUBJID = 1:10,
TUDTC = 1:10,
VISIT = "C1D1",
TUORRES = "TARGET",
TULOC = "LIVER",
TUSPID = "FORMNAME-R:19/L:19XXXX",
stringsAsFactors=FALSE
```

```
)

check_tu_tuloc_missing(TU)

TU$TULOC[1] = "NA"
TU$TULOC[2] = ""
TU$TULOC[3] = NA

check_tu_tuloc_missing(TU,preproc=roche_derive_rave_row)

TU$TUSPID <- NULL
check_tu_tuloc_missing(TU)
```

`check_vs_height` *Check for missing height values*

Description

This check looks for both records where height is missing as well as DM patients with no height records at all

Usage

```
check_vs_height(VS, DM)
```

Arguments

VS	Vital Signs SDTM dataset with variables USUBJID,VSTEST,VSTESTCD,VSSTRESN,VISIT
DM	Demographics SDTM dataset with variable USUBJID

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Author(s)

Sara Bodach

Examples

```
DM <- data.frame(
 STUDYID = 1,
 USUBJID = 1:10
)

VS <- data.frame(
 STUDYID = 1,
```

```
 USUBJID = 1:10,
 VSTEST = "HEIGHT",
 VSTESTCD = "HEIGHT",
 VSSTRESN = 1:10,
 VISIT = 1:10
)

check_vs_height(VS,DM)

DM <- data.frame(
 STUDYID = 1,
 USUBJID = 1:11
)

VS$VSSTRESN[1] = NA
VS$VSSTRESN[2] = "NA"
VS$VSSTRESN[3] = ""
VS$VSSTRESN[4] = "."

check_vs_height(VS,DM)
```

`check_vs_sbp_lt_dbp` *Check that DBP is not higher than SBP in VS*

Description

This check looks for non-missing diastolic BP is not higher than non-missing systolic BP

Usage

```
check_vs_sbp_lt_dbp(VS)
```

Arguments

VS	Vital Signs SDTM dataset with variables USUBJID,VISIT,VSDTC,VSTESTCD,VSSTRESN,VSSPID

Value

boolean value if check failed or passed with 'msg' attribute if the test failed

Examples

```
vs <- data.frame(
 STUDYID = 1,
 USUBJID = 1,
 VSSPID = c("1","2","1","2"),
 VISIT = 1,
 VSDTC = c("2010-01-01","2010-01-01","2010-01-01","2010-01-01"),
```

```
VSTESTCD = c("SYSBP","SYSBP",
             "DIABP","DIABP")
             ,
VSSTRESN = c(80,120,100,80)
)

vs0 <- subset(vs, select = c(USUBJID, VSSPID, VSSTRESN))

check_vs_sbp_lt_dbp(VS=vs)
check_vs_sbp_lt_dbp(VS=vs0)
```

`check_vs_vsdtc_after_dd` *Check for VS dates occurring after death date*

Description

This check looks for VS dates that occur after death date

Usage

```
check_vs_vsdtc_after_dd(AE, DS, VS)
```

Arguments

AE	Adverse Event SDTM dataset with variables USUBJID, AEDTHDTC, AESTDTC, AEDECOD, and AETERM
DS	Disposition SDTM dataset with variables USUBJID, DSSTDTC, DSDECOD, and DSTERM
VS	Vital Signs SDTM dataset with variables USUBJID, VSDTC, VSTESTCD, and VSORRES

Value

Boolean value for whether the check passed or failed, with 'msg' attribute if the check failed

Author(s)

Nina Ting Qi

Examples

```
AE <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 AEDTHDTC = c(rep("", 4), "2016-01-01"),
                 AESTDTC = rep("2016-01-01", 5),
                 AEDECOD = LETTERS[1:5], AETERM = LETTERS[1:5],
                 stringsAsFactors = FALSE)

DS <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 DSSTDTC = rep("2016-01-02", 5),
                 DSDECOD = c(LETTERS[1:4], "death"),
                 DSTERM = letters[1:5],
                 stringsAsFactors = FALSE)

VS <- data.frame(STUDYID = 1:5, USUBJID = LETTERS[1:5],
                 VSDTC = rep("2015-12-31", 5),
                 VSTESTCD = letters[1:5],
                 VSORRES = 1:5,
                 stringsAsFactors = FALSE)

check_vs_vsdtc_after_dd(AE, DS, VS)

VS$VSDTC[1] <- "2016-01-03"
VS$USUBJID[1] <- VS$USUBJID[5]

check_vs_vsdtc_after_dd(AE, DS, VS)
```

---

create_R_script *Create .R file with sdtmchecks function calls*

Description

Function that uses sdtmchecksmeta as input and creates .R file with function calls

Usage

```
create_R_script(
  metads = sdtmchecksmeta,
  file = "sdtmchecks_run_all.R",
  verbose = TRUE
)
```

Arguments

metads	sdtmchecksmeta file
file	filename and/or path to save to
verbose	Print information to console

Value

R script with user specified sdtmchecks based on sdtmchecksmeta file

Author(s)

Monarch Shah

See Also

Reporting-related utility functions `convert_var_to_ascii()`, `truncate_var_strings()`

Examples

```
# All checks are output to a file
fileName <- file.path(tempdir(), "run_all_checks.R")
create_R_script(file = fileName)

# Only include selected checks
fileName <- file.path(tempdir(), "run_some_checks.R")
mymetads = sdtmchecksmeta[sdtmchecksmeta$category == "ALL" & sdtmchecksmeta$priority == "High",]
create_R_script(metads = mymetads, file = fileName)

# Roche specific function calls
fileName <- file.path(tempdir(), "run_all_checks_roche.R")
mymetads = sdtmchecksmeta
mymetads$fxn_in=mymetads$fxn_in_roche
create_R_script(metads = mymetads, file = fileName)
```

---

diff_reports *Create a sdtmchecks list object with column indicating whether the issue was previously seen*

Description

This report will identify flagged records from an sdtmchecks report that are "new" and those that are "old" for a study. This will help quickly target newly emergent issues that may require a new query or investigation while indicating issues that were encountered from a prior report and may have already been queried.

This `diff_reports()` function requires a newer and older set of results from `sdtmchecks::run_all_checks()`, which will generate a list of check results. An added column "Status" is created with values of "NEW" and "OLD" in the list of check results, flagging whether a given record that is present in the new result (ie `new_report`) is also present in the old result (ie `old_report`). It makes a difference which report is defined as "new" and "old". This code only keeps results flagged in the new report and drops old results not in the new report because they were presumably resolved.

Usage

```
diff_reports(old_report, new_report)
```

Arguments

old_report	an older sdtmchecks list object as created by run_all_checks
new_report	a newer sdtmchecks list object as created by run_all_checks

Value

list of sdtmchecks results based on new_report with Status indicator

See Also

Example programs for running data checks report_to_xlsx(), run_all_checks(), run_check()

Examples

```
# Step 1: Simulate an older AE dataset with one missing preferred term

 ae <- data.frame(
 USUBJID = 1:5,
 DOMAIN = c(rep("AE", 5)),
 AESEQ = 1:5,
 AESTDTC = 1:5,
 AETERM = 1:5,
 AEDECOD = 1:5,
  AESPID = c("FORMNAME-R:13/L:13XXXX",
             "FORMNAME-R:16/L:16XXXX",
             "FORMNAME-R:2/L:2XXXX",
             "FORMNAME-R:19/L:19XXXX",
             "FORMNAME-R:5/L:5XXXX"),
 stringsAsFactors = FALSE
)

ae$AEDECOD[1] = NA

# Step 2: Use the run_all_checks() function to generate list of check results on this "old" data

# Filter sdtmchecksmeta so that only one check is present
metads <- sdtmchecksmeta[sdtmchecksmeta$check=="check_ae_aedecod",]
old <- run_all_checks(metads=metads)



#Step 3: Simulate a newer, updated AE dataset with another record with a new missing preferred term

 ae <- data.frame(
 USUBJID = 1:6,
 DOMAIN = c(rep("AE", 6)),
 AESEQ = 1:6,
```

```
 AESTDTC = 1:6,
 AETERM = 1:6,
 AEDECOD = 1:6,
  AESPID = c("FORMNAME-R:13/L:13XXXX",
             "FORMNAME-R:16/L:16XXXX",
             "FORMNAME-R:2/L:2XXXX",
             "FORMNAME-R:19/L:19XXXX",
             "FORMNAME-R:5/L:5XXXX",
             "FORMNAME-R:1/L:5XXXX"
             ),
 stringsAsFactors = FALSE
)

ae$AEDECOD[1] = NA
ae$AEDECOD[6] = NA

# Step 4: use the run_all_checks() function to generate list of check results  on this "new" data

new <- run_all_checks(metads=metads)

# Step 5: Diff to create a column indicating if the finding is new
res <- diff_reports(old_report=old, new_report=new)

## optionally output results as spreadsheet with sdtmchecks::report_to_xlsx()
# report_to_xlsx(res, outfile=paste0("saved_reports/sdtmchecks_diff_",Sys.Date(),".xlsx"))
```

`run_all_checks` *Run all data checks in sdtmchecks package using parallel processing*

Description

This function runs all checks in the sdtmchecks package. It expects SDTM domains saved as dataframe objects in your global environment. These dataframes should have lowercase names, e.g., dm.

Usage

```
run_all_checks(
  metads = sdtmchecksmeta,
  priority = c("High", "Medium", "Low"),
  type = c("ALL", "ONC", "COVID", "PRO", "OPHTH"),
  verbose = TRUE,
  ncores = 1
)
```

Arguments

metads	Metadata to use to execute the checks. The default is the sdtmchecksmeta dataframe available in the package. This object could easily be customized, subset, etc.
priority	Priority level of data checks, i.e., c("High", "Medium", "Low"). NULL runs all priority levels.
type	Type of data checks, i.e., c("ALL", "ONC", "COV", "PRO", "OPHTH"). NULL runs all type.
verbose	Whether to display messages while running
ncores	Number of cores for parallel processing, with default set to 1 (sequential)

Details

To look up documentation for the data checks in package, please use command ??sdtmchecks

Value

list with results from individual data check functions

See Also

Example programs for running data checks `diff_reports()`, `report_to_xlsx()`, `run_check()`

Examples

```
# Assuming sdtm datasets are in your global environment
# Note we are only specifying AE and DS here so all unrelated checks wont be run

ae <- data.frame(
    STUDYID = 1,
    USUBJID = c(1,2,3,1,2,3),
    AESTDTC = '2020-05-05',
    AETERM  = c("abc Covid-19", "covid TEST POSITIVE",rep("other AE",4)),
    AEDECOD = c("COVID-19", "CORONAVIRUS POSITIVE", rep("OTHER AE",4)),
    AEACN = c("DRUG WITHDRAWN", rep("DOSE NOT CHANGED",5)),
    AESPID = "FORMNAME-R:13/L:13XXXX",
    stringsAsFactors = FALSE
)

ds <- data.frame(
 USUBJID = c(1,1,2,3,4),
 DSSPID  = 'XXX-DISCTX-XXX',
 DSCAT   = "DISPOSITION EVENT",
 DSDECOD = "OTHER REASON",
 DSSEQ = c(1,2,1,1,1),
 stringsAsFactors = FALSE
)

all_rec<-run_all_checks(metads=sdtmchecksmeta,
                        verbose=FALSE)
```

run_check *Run a single check in sdtmchecks package*

Description

This function runs a single check in the sdtmchecks package. It expects a check name, the function that performs the check and some info for the pdf and Excel files. It also expects a T/F value that determines whether to display messages while running. Excluding verbose, the parameters for this function are usually passed to it by filtering the metads to only contain the row corresponding to the check of interest, and then assigning each parameter as the contents of the eponymous column of metads (see example below). This is because this function is mostly run inside of an mcmapply in the run_all_checks_parallel function, which loops over the checks in the rows of metads.

Usage

```
run_check(
  check,
  fxn_in,
  xls_title,
  pdf_title,
  pdf_subtitle,
  pdf_return,
  verbose
)
```

Arguments

check	Check name.
fxn_in	Function performing the check.
xls_title	Excel title.
pdf_title	PDF title.
pdf_subtitle	PDF subtitle.
pdf_return	Text to display in PDF if check does not run.
verbose	Whether to display messages while running

Details

to look up documentation for the data checks package, please use command ??sdtmchecks

Value

list with results from the check.

See Also

Example programs for running data checks diff_reports(), report_to_xlsx(), run_all_checks()

Examples

```
# Assuming sdtm datasets are in your global environment

 ae <- data.frame(
 USUBJID = 1:5,
 DOMAIN = c(rep("AE", 5)),
 AESEQ = 1:5,
 AESTDTC = 1:5,
 AETERM = 1:5,
 AEDECOD = 1:5,
  AESPID = c("FORMNAME-R:13/L:13XXXX",
             "FORMNAME-R:16/L:16XXXX",
             "FORMNAME-R:2/L:2XXXX",
             "FORMNAME-R:19/L:19XXXX",
             "FORMNAME-R:5/L:5XXXX"),
 stringsAsFactors = FALSE
)

ae$AEDECOD[1] = NA

# Filter sdtmchecksmeta so that only one check is present
metads <- sdtmchecksmeta[sdtmchecksmeta$check=="check_ae_aedecod",]

run_check(
  check = metads$check,
  fxn_in = metads$fxn_in,
  xls_title = metads$xls_title,
  pdf_title = metads$pdf_title,
  pdf_subtitle = metads$pdf_subtitle,
  pdf_return = metads$pdf_return,
  verbose = FALSE
)
```

---

sdtmchecksmeta *Metadata for sdtmchecks*

Description

A dataset containing the SDTM checks in the package. The data can be used as input into functions.

Usage

```
data(sdtmchecksmeta)
```

Format

A data frame with a row for each R check in the package:

check R check name, without .R file extension

category Therapeutic area grouping

priority High, Medium, Low

domains SDTM domains used in function

xls_title Excel title for tab

pdf_title PDF title for check

pdf_subtitle PDF subtitle for check, with * at the start of each subtitle line

pdf_return PDF return message when SDTM domain not available

fxn_in explicit string input of domain name(s) into R check function

fxn_in_roche explicit string input of domain name(s) into R check function, Roche specific

mapping Is this related to mapping? i.e. Not a site issue.

exist_string explicit string input to check existence of SDTM domain(s) before running check

Examples

```
data(sdtmchecksmeta)
head(sdtmchecksmeta[,1:5])
```

Index