

Package ‘Rbonaut2’

January 27, 2016

Type Package

Title CLIP2's Rbonaut

Version 0.3

Date 2016-01-27

Author Cavorit

Maintainer Harald Fiedler <harald.fiedler@cavorit.de>

Depends data.table

Description The CLIP2-Version of CLIP1's Rbonaut-Package

License This package is private and internal of Cavorit Consulting GmbH

LazyData TRUE

R topics documented:

Rbonaut2-package	2
detectItemID	2
detectItemResponse	3
getAdrWAlsListe	3
getFirstAdrW	4
getHW	5
getItemICC	5
getNachname	6
getSessionTimeStamp	7
getVorname	7
gibZahlFuehrendeNullen	8
isMultiTarget	9
istFormatNachnameKommaVorname	9
isUTF8	10
playedAngle	11
plotFBN	11
readItemBank	12
SQL2DF	13
Index	14

Rbonaut2-package

Rbonaut2

Description

CLIP2-Paket

Author(s)

Harald Fiedler (c) Cavorit

detectItemID

detectItemID

Description

Liefert die ItemID eines Balls/Stimulus zurück

Usage

detectItemID(Stimulus)

Arguments

Stimulus ein data.frame mit den Spalten isMultiTarg, MultiTargs, RW, AW, HW, vA, sL und sR und einer Zeile. Es handelt sich also um eine Zeile aus DF, die einen Ball darstellt

Details

Liefert die ItemID eines Balls/Stimulus zurück, z.B. "BL03". Die Funktion ist nicht vektorwertig implementiert, sondern kann immer nur eine Abfrage auf einmal durchführen

Value

character der Länge 1, z.B. c("BL03")

Author(s)

Harald Fiedler

detectItemResponse	<i>detectItemResponse</i>
--------------------	---------------------------

Description

Liefert das Ergebnis eines Balls/Stimulus zurück, z.B. 0 oder 1

Usage

```
detectItemResponse(Stimulus)
```

Arguments

Stimulus	ein data.frame mit den Spalten isMultitarg, MultiTargs, RW, AW, HW, vA, sL und sR und einer Zeile. Es handelt sich also um eine Zeile aus DF, die einen Ball darstellt
----------	--

Details

Liefert das Ergebnis eines Balls/Stimulus zurück, z.B. 0 oder 1. Aus dem data.frame ist nicht ersichtlich, welches für welches Modell die ItemResponse erhoben wird. Im dichotomen Rasch Modell wird das Ergebnis auf 0-1 codiert, während es für andere Modelle andere Erfassungen geben mag. Hier muss extern geklärt werden, welche ItemID welchem Modell zugeordnet ist.

Value

data frame mit der zusätzlichen Spalte ItemResponse

Author(s)

Harald Fiedler

getAdrWAlsListe	<i>getAdrWAlsListe</i>
-----------------	------------------------

Description

Hilfsfunktion von SQL2DF()

Usage

```
getAdrWAlsListe(adrW)
```

Arguments

adrW	character
------	-----------

Details

In den DB-Abfragen von CGoal findet sich die Variable `adrW` für die Zielfelder. Beim Umstellen von Single-Target auf Multi-Target wurde aus einer Zahl nun einen String, der einen JSON-Vektor darstellt. Wenn also Früher nur das Zielfeld 7 angegeben war, kann bei Multitarget nun der Ausdruck "7, 2, 21" angegeben sein. Die hier vorliegende Funktion arbeitet Vektorwertig und macht beispielsweise aus den Tabelleneinträgen `c("1, 2, 3, 4", "11, 12, 13, 14")` eine List der Form `list(c(1, 2, 3, 4), c(11, 12, 13, 14))`

Value

list mit numerischen Elementen

Author(s)

Harald Fiedler

Examples

```
adrW <- c("{1, 2, 3, 4}", "{11, 12, 13, 14}")
getAdrWAlsListe(adrW = adrW)
```

`getFirstAdrW`

getFirstArdW

Description

Hilfsfunktion von SQL2DF(): Gibt erstes Ziel in `adrW` im numerischen Format

Usage

```
getFirstAdrW(adrW)
```

Arguments

`adrW` character Vektor, etwa `c("2, 4, 5", "12,19", "4")`

Details

Bei der Umstellung von Single-Target auf Multi-Target wurden die Einträge in der FBN-Datenbank stark abgeändert. Wo früher beispielsweise eine Zahl 7 für das Zielfeld mit der Adresse 7 stand, ist nun "3, 5, 15" ein String, der die unterschiedlichen Zielfelder darstellt. Unabhängig davon, ob in `adrW` ein `multiTarget` oder `singleTarget`-Design hinterlegt wird, liefert diese Funktion nur das erste Ziel zurück, und zwar als Zahl.

Value

numeric

Author(s)

Harald Fiedler

Examples

```
getFirstAdrW(adrW=c("{2, 4, 5}", "{12, 19}", "{4}"))
```

getHW	<i>getHW</i>
-------	--------------

Description

Hilfsfunktion von SQL2DF zur Ermittlung von Höhenwinkel FF-FH-HF-HH

Usage

```
getHW(SQL)
```

Arguments

SQL data.frame welches durch read.csv() einer SQL-Query entnommen wurde

Details

Je nachdem ob eine obere Ballkanonen oder eine untere Ballkanone zum Zuge kommt, oder ein unteres Ziel respektive oberes Ziel, kommt ein anderer Höhenwinkel zu stande.

Value

character mit Einträgen aus c("FF", "FH", "HF", "HH"), wobei FF=Flach Flach bedeutet und HH=Hoch Hoch.

Author(s)

Harald Fiedler

getItemICC	<i>getItemICC</i>
------------	-------------------

Description

Fügt die Spalten ItemID, ICCa, ICCb, ICCc, ICCd

Usage

```
getItemICC(DF, ItemBank, MaximaleToleranz = 10)
```

Arguments

DF data.frame Mittels SQL2DF(SQL=SQL) erzeugt wird
ItemBank data.frame Wird etwa durch ItemBank=readItemBank() gewonnen
MaximaleToleranz numeric der Länge 1, gibt an, wie viel sL.x von sL.y bzw. sR.x von sR.y abweichen darf, damit das Item in DF identifiziert wird mit dem Item aus der Itembank. Dabei stammt *.x aus DF und *.y aus der ItemBank. Default ist 10.

Details

Aus einem DF (erzeugt mittels SQL2DF(SQL = SQL)) und der ItemBank wird ein Merge erzeugt. Dabei nutze ich AW, RW, HW und vA als Key. In einem zweiten Schritt wird die Identifikation der Items gelöscht, wenn zwischen dem tatsächlichen sL bzw. sR und dem in der ItemBank hinterlegten sL und sR eine zu große Diskrepanz entsteht.

Value

DF wird um die Spalten ItemID, ICCa, ICCb, ICCc und ICCd angereichert.

Author(s)

Harald Fiedler

Examples

```
Pfad <- system.file("extdata", package="Rbonaut2", "Footbonaut_Datenabfrage_RicoWehrle.csv")
SQL <- read.csv(file=Pfad, sep=";", header=TRUE, encoding="utf8", stringsAsFactors = FALSE)
DF <- SQL2DF(SQL = SQL)
ItemBank=readItemBank()
F14 <- getItemICC(DF=DF, ItemBank=readItemBank(), MaximaleToleranz=15)
head(F14)
```

getNachname

getNachname

Description

Hilfsfunktion von SQL2DF(): Gibt aus einem Spielernamen den Vornamen

Usage

```
getNachname(Spielernamen)
```

Arguments

Spielernamen character Vektor von beliebiger Länge

Details

Spielernamen können in SQL-Abfragen des FBN beispielsweise "Dogan, Isa" sein. Es wird "Isa" zurückgegeben.

Value

character Vektor der gleichen Länge wie der an die Funktion übergebene Vektor

Author(s)

Harald Fiedler

Examples

```
Spielername <- c("Fiedler, Harald", "Mayer, Jan", "A-Team")
getNachname(Spielername = Spielername)
```

<code>getSessionTimeStamp</code>	<i>getSessionTimeStamp</i>
----------------------------------	----------------------------

Description

Hilfsfunktion von SQL2DF(): ermittelt Sessionstart

Usage

```
getSessionTimeStamp(DatumString)
```

Arguments

DatumString String, etwa "2015-08-27 18:59:25.328383+02"

Details

Macht aus 2015-08-27 18:59:25.328383+02 den String 18:59:25

Value

Ein String, etwas "18:59:25"

Author(s)

Harald Fiedler

Examples

```
DatumString <- c("2015-08-27 18:59:25.328383+02", "2015-08-27 18:59:25.328383+02", "2015-08-27 18:59:25.328383+02")
```

<code>getVorname</code>	<i>getVorname</i>
-------------------------	-------------------

Description

Hilfsfunktion von SQL2DF(): Gibt aus einem Spielername den Vornamen

Usage

```
getVorname(Spielername)
```

Arguments

Spielername character Vektor von beliebiger Länge

Details

Spielernamen können in SQL-Abfragen des FBN beispielsweise "Dogan, Isa" sein. Es wird "Isa" zurückgegeben.

Value

character Vektor der gleichen Länge wie der an die Funktion übergebene Vektor

Author(s)

Harald Fiedler

Examples

```
Spielersname <- c("Fiedler, Harald", "Mayer, Jan", "A-Team")
getVorname(Spielersname = Spielersname)
```

```
gibZahlFuehrendeNullen
```

```
gibZahlFuehrendeNullen
```

Description

Hilfsfunktion von SQL2DF: aus c(3) mach c("003")

Usage

```
gibZahlFuehrendeNullen(k, digits = 3)
```

Arguments

k	numeric (besser wäre integer, sonst wird das Ergebnis korrumpiert)
digits	numeric der Länge 1, das die Wortlänge bezeichnet. "0004" erhält man beispielsweise mit digits=4

Details

Wenn man idX <- 1:31 nutzt, um einen Index idB zu erstellen, erhält man einen eindeutigen Schlüssel. Allerdings verhält sich die lexikografische Sortierung nicht, wie man es vielleicht möchte. So würde auf die idB=1 nicht etwa idB=2 folgen, sondern idB=11. Daher macht es Sinn, bei der Konvertierung einer Ziffer oder Zahl in ein Character eine gewisse Anzahl an Nullen voranzustellen. So wird etwa aus der Zahl 2 das Wort "002" gemacht, wodurch die lexikografische Sortierung wieder so funktioniert, wie man es gerne hätte.

Value

Ein Vektor mit der gleichen Länge wie k, dessen Elemente Worte sind. Sie example.

Author(s)

Harald Fiedler

Examples

```
k = c(2, 7, 17, 299)
gibZahlfuehrendeNull(k=k, digits=9)
```

`isMultiTarget`*isMultiTarget*

Description

Hilfsfunktion von SQL2DF(): ist adrW multitargetting?

Usage

```
isMultiTarget(adrW)
```

Arguments

adrW character Array, etwa c("22, 33, 44, 55", "11, 22222, 11111", "99")

Details

Sagt, ob 11, 21, 16 oder 23 unter adrW abgespeichert wurde

Value

boolescher Vektor

Author(s)

Harald Fiedler

`istFormatNachnameKommaVorname`*istFormatNachnameKommaVorname*

Description

Hilfsfunktion von SQL2DF()

Usage

```
istFormatNachnameKommaVorname(Spielername)
```

Arguments

Spielername String

Details

In den SQL-Auszügen des FBN finden sich Spielernamen vom Format "Fiedler, Harald", aber auch "A_TEST_Forschung". Die Funktion testet komponentenweise, ob zwei Strings kommasetrennt gepastet sind.

Value

Boolescher Wert, der angibt, ob das Format Name, Vorname (mutmaßlich) vorliegt

Author(s)

Harald Fiedler

Examples

```
Spielernamen <- c("Fiedler, Harald", "Mayer, Jan", "A-Team")
istFormatNachnameKommaVorname(Spielernamen = Spielernamen)
```

isUTF8	<i>isUTF8</i>
--------	---------------

Description

schätzt, ob UTF8 Codierung vorliegt

Usage

```
isUTF8(file, echo = TRUE)
```

Arguments

file	character der Länge 1 der angibt, ob schätzungsweise UTF8 vorliegt
echo	boolean der Länge 1 sagt, ob der Systemoutput angezeigt werden soll

Details

Diese Funktion funktioniert nur auf Mac, wobei hierzu per `*brew install moreutils*` installiert sein muss. Sie gibt `character(0)` zurück, wenn kein utf8-nonkonformes Zeichen gefunden wurde, und ansonsten eine Liste mit Angaben zu invaliden Zeichen

Value

siehe `*details*`

Author(s)

Harald Fiedler

Examples

```
isUTF8(file="~/Desktop")
```

playedAngle	<i>playedAngle</i>
-------------	--------------------

Description

Winkel zwischen zwei Adressen

Usage

```
playedAngle(adrA, adrB, Clockwise = FALSE)
```

Arguments

Clockwise	boolean der angibt, ob die Winkel im Uhrzeigersinn oder im mathematisch positiven Sinn gemessen werden. Default ist der mathematisch positive Sinn.
A	numeric Adressen der Ausgangsfelder
B	numeric Adressen der Zielfelder

Details

Gibt den Winkel zwischen zwei FBN-Adressen

Value

numeric mit Winkel aus 0:180

Author(s)

Harald Fiedler

Examples

```
adrA=10  
adrB=18  
plotFBN()  
playedAngle(adrA=adrA, adrB=adrB, Clockwise = FALSE)
```

plotFBN	<i>plotFBN</i>
---------	----------------

Description

plotFBN

Usage

```
plotFBN(Adresses = TRUE)
```

Arguments

Adresses	boolescher Wert der angibt, ob die Fensteradressen mit eingegeben werden sollen.
----------	--

Details

Zeichnet schematisch den Footbonaut

Author(s)

Harald Fiedler

Examples

```
plotFBN(Adresses=FALSE)
```

readItemBank

readItemBank

Description

Liest die ItemBank ein

Usage

```
readItemBank(file = NA)
```

Arguments

Pfad	character der Länge 1, der den Pfadname zu einer .csv-Datei darstellt. Die Datei muss eine gültige ItemBank im Sinne des 4-PL-Modells sein. Als Default-Wert für den Pfad fungiert ein Pfad zu einer Pakte-Datei, die in der Lib installiert wurde (was der eigentliche Clou dieser Funktion ist).
------	--

Details

Es wird die ItemBank im 1:4PL-Modell eingelesen.

Value

data.frame für das 4PL-Modell

Author(s)

Harald Fiedler

Examples

```
ItemBank <- readItemBank()  
head(ItemBank)
```

`SQL2DF`*SQL2DF*

Description

SQL zu data.frame

Usage

`SQL2DF(SQL)`

Arguments

`SQL` data.frame, dass per read.csv eingelesen wurde

Details

Mit shinySQL erhalten wir von CGoal SQL-Abfragen händisch als .csv-Files zurück. Diese werden in ein data.frame umgewandelt

Value

data.frame

Author(s)

Harald Fiedler

Examples

```
message("Ich lade den R-Paket-internen RAW-Datensatz: Footbonaut_Datenabfrage_RicoWehrle.csv")
Pfad <- system.file("extdata", package="Rbonaut2", "Footbonaut_Datenabfrage_RicoWehrle.csv")
SQL <- read.csv2(file=Pfad, sep = ",", stringsAsFactors = FALSE, encoding = "utf8")
DF <- SQL2DF(SQL=SQL)
head(DF)
```

Index

*Topic **package**

Rbonaut2-package, [2](#)

detectItemID, [2](#)

detectItemResponse, [3](#)

getAdrWAlisListe, [3](#)

getFirstAdrW, [4](#)

getHW, [5](#)

getItemICC, [5](#)

getNachname, [6](#)

getSessionTimeStamp, [7](#)

getVorname, [7](#)

gibZahlFuehrendeNullen, [8](#)

isMultiTarget, [9](#)

istFormatNachnameKommaVorname, [9](#)

isUTF8, [10](#)

playedAngle, [11](#)

plotFBN, [11](#)

Rbonaut2 (Rbonaut2-package), [2](#)

Rbonaut2-package, [2](#)

readItemBank, [12](#)

SQL2DF, [13](#)