

# Package ‘Rbonaut2’

December 5, 2016

**Type** Package

**Title** CLIP2's Rbonaut

**Version** 0.7

**Date** 2016-11-07

**Author** Cavorit

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

**Depends** data.table, RPostgreSQL, sfsmisc

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

**License** This package is private and internal of Cavorit Consulting GmbH

**LazyData** TRUE

**RoxygenNote** 5.0.1

## R topics documented:

Rbonaut2-package	2
adaptiv.BL16.fullRandom.modelPredict	2
adaptiv.BL16.fullRandom.modelTransform	4
adaptiv.BL16.modelPredict	6
askDB	7
augmentRAW	8
calcFiedler2016a	9
detectItemID	9
detectItemIDLIVE	10
detectItemResponse	11
erstelleRaschMatrixSkeleton	11
fillRaschMatrixSkeleton	12
getAdrWAlsListe	13
getFirstAdrW	14
getHW	14
getNachname	15
getSessionTimeStamp	16

getVorname . . . . .	16
gibZahlFuehrendeNullen . . . . .	17
implodeRaschMatrix4Quality . . . . .	18
isMultiTarget . . . . .	19
istFormatNachnameKommaVorname . . . . .	19
itemID2Params . . . . .	20
NormTree . . . . .	21
playedAngle . . . . .	21
plotFBN . . . . .	22
plotSeaShell . . . . .	22
readAUGMENTED . . . . .	23
readItemBank . . . . .	24
readRAW . . . . .	25
writeAUGMENTED . . . . .	25
writeRAW . . . . .	26
<b>Index</b>	<b>27</b>

---

Rbonaut2-package	<i>Rbonaut2</i>
------------------	-----------------

---

**Description**

CLIP2-Paket

**Author(s)**

Harald Fiedler (c) Cavorit

---

adaptiv.BL16.fullRandom.modelPredict
<i>adaptiv.BL16.fullRandom.modelPredict</i>

---

**Description**

modelPredict() fuer yhat-Modell eines voll randomisierten Samplers des BL32-Testraums mit Stop nach 16 Bällen

**Usage**

adaptiv.BL16.fullRandom.modelPredict(AnfrageDF)

## Arguments

DF

data.frame mit den Spalten:

- TestID: den Namen des Testformats. Beispielsweise: 'BL32'
- idS: die aktuelle Session-ID, beispielsweise: '002b6573-cf12-436d-bccd-0856b0bb0a25'
- idP: die Player-ID des jeweiligen Probanden, beispielsweise: 'fe553db4-bbde-43dd-a6a0-804b9e46c57'
- NamePlayer: Klarnamen des Spielers, beispielsweise: 'Mustermann, Tim'
- Birthday: Geburtsdatum des Spielers, beispielsweise: '2002-07-17'
- Team: Bezeichnung des Teams, z.B. "U17"
- SessionStart: der Zeitstempel für den Sessionsstart als String im Format "JJJJ-MM-TT HH:MM:SS", beispielsweise: '2014-03-22 13:42:03'
- adrB: eine Liste mit den Adressen der Ballkanonen, in der Reihenfolge ihrer Aktivierung, beispielsweise: [10, 45, 28] oder bei Sessionstart eine leere Liste '[]'.
- adrW: eine Liste von Listen mit den aufleuchtenden Zielfeldern. Beispielsweise: '[[4, 6], [21, 22], [50, 18]]' oder bei Sessionstart '[[[]]'
- adrCol: eine Liste von Listen mit der Farbe der Zielfelder aus [[adrW]]. Die Farben werden alphabetisch in Großbuchstaben durchnummeriert. Beispielsweise: "[['A', 'B'], ['A', 'B'], ['B', 'A']]" oder bei Sessionstart '[[[]]'
- adrOut: eine Liste mit den Adressen, wo der Ball tatsächlich raus ist. Beispielsweise: '[4, 23, 40]' oder bei Sessionstart '[]'
- FBt: eine Liste mit den Angaben über die Zeitdauer zwischen Ballkanoneneinwurf und Lichtschrankensignal beim Rausschießen des Balls (in Millisekunden), beispielsweise: '[2140, 2600, 8600]' oder bei Sessionstart '[]'

## Details

Es handelt sich hier um die modelPredict-Funktion für yhat-Architektur. Bei dem Dienst handelt es sich um einen reinen Item-Sampler. Solange die Session-History weniger als 16 Bälle aufweist, wird aus dem Item-Raum der BL32 ein Item gewählt.

## Value

Ein JSON-gültiger String mit den folgenden Informationen:

- TicketID ein stochastischer Identifikator für die REST-Request, z.B. "JTMfOgfTEiq6ZMxb"
- GameOver ein boolescher Wert der angibt, ob das Abbruchkriterium für die Testung erreicht wurde
- NextB Eine liste mit Informationen über den nächsten Ball
- Testergebnis Das Testergebnis, dass der Spieler auf der latenten Kompetenzdimension erhält. (Noch nicht implementiert, daher vorläufig NA)

## Author(s)

Harald Fiedler

---

 adaptiv.BL16.fullRandom.modelTransform

*adaptiv.BL16.fullRandom.modelTransform*


---

## Description

modelTransform() fuer yhat-Modell eines voll randomisierten Samplers des BL32-Testraums mit Stop nach 16 Bällen

## Usage

```
adaptiv.BL16.fullRandom.modelTransform(AnfrageJSONstring)
```

## Arguments

AnfrageJSONstring

String, der in R zu einer Liste umgewandelt werden kann mit folgenden Elementen:

- TestID: den Namen des Testformats. Beispielsweise: 'BL32'
- idS: die aktuelle Session-ID, beispielsweise: '002b6573-cf12-436d-bccd-0856b0bb0a25'
- idP: die Player-ID des jeweiligen Probanden, beispielsweise: 'fe553db4-bbde-43dd-a6a0-804b9e46c57'
- NamePlayer: Klarnamen des Spielers, beispielsweise: 'Mustermann, Tim'
- Birthday: Geburtsdatum des Spielers, beispielsweise: '2002-07-21'
- Team: Bezeichnung des Teams, z.B. "U17"
- SessionStart: der Zeitstempel für den Sessionsstart als String im Format "JJJJ-MM-TT HH:MM:SS", beispielsweise: '2014-03-22 13:42:03'
- adrB: eine Liste mit den Adressen der Ballkanonen, in der Reihenfolge ihrer Aktivierung, beispielsweise: [10, 45, 28] oder bei Sessionstart eine leere Liste '[]'.
- adrW: eine Liste von Listen mit den aufleuchtenden Zielfeldern. Beispielsweise: '[[4, 6], [21, 22], [50, 18]]' oder bei Sessionstart '[]'
- adrCol: eine Liste von Listen mit der Farbe der Zielfelder aus [[adrW]]. Die Farben werden alphabetisch in Großbuchstaben durchnummeriert. Beispielsweise: "[['A', 'B'], ['A', 'B'], ['B', 'A']]" oder bei Sessionstart '[]'
- adrOut: eine Liste mit den Adressen, wo der Ball tatsächlich raus ist. Beispielsweise: '[4, 23, 40]' oder bei Sessionstart '[]'
- FBt: eine Liste mit den Angaben über die Zeitdauer zwischen Ballkanoneneinwurf und Lichtschrankensignal beim Rausschießen des Balls (in Millisekunden), beispielsweise: '[2140, 2600, 8600]' oder bei Sessionstart '[]'

Hier ein Beispiel für eine gültige Anfrage für die Abfrage eines ersten Balls:

```
#' testJSONrequest <- '{
"TestID" : "BL32",
```

```
"idS" : "002b6573-cf12-436d-bccd-0856b0bb0a25",
"idP" : "fe553db4-bbde-43dd-a6a0-804b9e46c57",
"NamePlayer" : "Mustermann, Tim",
"Birthday" : "2002-07-21",
"Team" : "U14",
"SessionStart" : "2014-03-22 13:42:03",
"adrB" : [],
"adrW" : [],
"adrCol" : [],
"adrOut" : [],
"FBt" : []
}'
```

Für einen vierten Ball:

```
testJSONrequest <- '{
"TestID" : "BL32",
"idS" : "002b6573-cf12-436d-bccd-0856b0bb0a25",
"idP" : "fe553db4-bbde-43dd-a6a0-804b9e46c57",
"NamePlayer" : "Mustermann, Tim",
"Birthday" : "2002-07-21",
"Team" : "U14",
"SessionStart" : "2014-03-22 13:42:03",
"adrB" : [10, 45, 28],
"adrW" : [[4, 6], [21, 22], [50, 18]],
"adrCol" : [["A", "B"], ["A", "B"], ["B", "A"]],
"adrOut" : [4, 23, 40],
"FBt" : [2140, 2600, 8600]
}'
```

## Details

Es handelt sich hier um die modelTransform-Funktion für yhat-Architektur. Bei dem Dienst handelt es sich um einen reinen Item-Sampler. Solange die Session-History weniger als 16 Bälle aufweist, wird aus dem Item-Raum der BL32 ein Item gewählt.

## Value

data.frame mit den oben beschriebenen Spalten.

## Author(s)

Harald Fiedler

---

```
adaptiv.BL16.modelPredict
      adaptiv.BL16.modelPredict
```

---

## Description

modelPredict() fuer yhat-Modell eines adaptiven Samplers nach 5 Bällen des BL32-Testraums mit Stop nach 16 Bällen. Basiert auf dem Skript BL32MultiTargetSimTest.r (Teilcode wurde kopiert) Der Output Ergebnis wurde nach dem adaptiv.BL16.fullRandom.modelPredict.R implementiert

## Usage

```
adaptiv.BL16.modelPredict>AnfrageDF)
```

## Arguments

DF	<p>data.frame mit den Spalten:</p> <ul style="list-style-type: none"> <li>• TestID: den Namen des Testformats. Beispielsweise: 'BL32'</li> <li>• idS: die aktuelle Session-ID, beispielsweise: '002b6573-cf12-436d-bccd-0856b0bb0a25'</li> <li>• idP: die Player-ID des jeweiligen Probanden, beispielsweise: 'fe55db4-bbde-43dd-a6a0-804b9e46c57'</li> <li>• NamePlayer: Klarname des Spielers, beispielsweise: 'Mustermann, Tim'</li> <li>• Birthday: Geburtsdatum des Spielers, beispielsweise: '2002-07-17'</li> <li>• Team: Bezeichnung des Teams, z.B. "U17"</li> <li>• SessionStart: der Zeitstempel für den Sessionsstart als String im Format "JJJJ-MM-TT HH:MM:SS", beispielsweise: '2014-03-22 13:42:03'</li> <li>• adrB: eine Liste mit den Adressen der Ballkanonen, in der Reihenfolge ihrer Aktivierung, beispielsweise: [10, 45, 28] oder bei Sessionstart eine leere Liste '[]'.</li> <li>• adrW: eine Liste von Listen mit den aufleuchtenden Zielfeldern. Beispielsweise: '[[4, 6], [21, 22], [50, 18]]' oder bei Sessionstart '[[[]]'</li> <li>• adrCol: eine Liste von Listen mit der Farbe der Zielfelder aus [[adrW]]. Die Farben werden alphabetisch in Großbuchstaben durchnummeriert. Beispielsweise: "[['A', 'B'], ['A', 'B'], ['B', 'A']]" oder bei Sessionstart '[[[]]'</li> <li>• adrOut: eine Liste mit den Adressen, wo der Ball tatsächlich raus ist. Beispielsweise: '[4, 23, 40]' oder bei Sessionstart '[]'</li> <li>• FBt: eine Liste mit den Angaben über die Zeitdauer zwischen Ballkanoneneinwurf und Lichtschrankensignal beim Rausschießen des Balls (in Millisekunden), beispielsweise: '[2140, 2600, 8600]' oder bei Sessionstart '[]'</li> </ul>
----	---

## Details

Es handelt sich hier um die modelPredict-Funktion für yhat-Architektur. Bei dem Dienst handelt es sich um einen reinen Item-Sampler. Solange die Session-History weniger als 16 Bälle aufweist, wird aus dem Item-Raum der BL32 ein Item gewählt. Der erste Ball ist immer BL01, die nächsten 4 Bälle werden ohne Zurücklegen gezogen.

## Value

Ein JSON-gültiger String mit den folgenden Informationen:

- TicketID ein stochastischer Identifikator für die REST-Request, z.B. "JTMfOgfTEiq6ZMxb"
- GameOver ein boolescher Wert der angibt, ob das Abbruchkriterium für die Testung erreicht wurde
- NextB Eine liste mit Informationen über den nächsten Ball
- Testergebnis Das Testergebnis, dass der Spieler auf der latenten Kompetenzdimension erhält. (Noch nicht implementiert, daher vorläufig NA)
- Seashell\_output\_base64, Grafik im base64 im JSON format, wenn gameover.# Um es als html zu öffnen: `<!DOCTYPE html><html><body></body></html>`

## Author(s)

Harald Fiedler, Kateryna Budzyak

---

askDB

askDB

---

## Description

Fragt die DB ab

## Usage

askDB(Anfangsdatum, Enddatum)

## Arguments

Anfangsdatum	character der Länge 1 im Format "JJJJ-MM-DD", welches dann zu einem Datumsobjekt umgewandelt wird. Achtung: Zeitzone könnte ein paar Probleme aufwerfen.
Enddatum	character der Länge 1 im Format "JJJJ-MM-DD"

## Details

Diese Funktion fragt auf localhost einen DB-dump der fbn-Datenbank ab und ersetzt das Copy&Paste-Verfahren der shinyApp

**Value**

data.frame das dann von augmentRAW() weiterverarbeitet werden kann.

**Author(s)**

Harald Fiedler

---

augmentRAW

*augmentRAW*

---

**Description**

data.frame SQL wird angereichert

**Usage**

```
augmentRAW(SQL)
```

**Arguments**

SQL                      data.frame, dass per askDB() oder readRAW() eingelesen wurde

**Details**

Es werden Derivate gebildet und Punktzahlen eingebunden. Problemhafte Sessions werden eliminiert. Dazu zählen zwei Sessions aus dem November 2014 ohne adrW.

**Value**

data.frame

**Author(s)**

Harald Fiedler



---

calcFiedler2016a	<i>calcFiedler2016a</i>
------------------	-------------------------

---

**Description**

Berechnet für eine einzelne Session (DF) theta-Hat nach jedem Ball

**Usage**

```
calcFiedler2016a(SessionDF, ItemBank = readItemBank())
```

**Arguments**

SessionDF	data.frame mit einer Session
ItemBank	die ItemBank, wird per default mittels readItemBank() eingelesen.

**Details**

DF darf nur eine Session beinhalten. Dann berechnet die Funktion thetaHat nach jedem Ball und stellt dies am Ende in der Spalte Fiedler2016a zur Verfügung.

**Value**

data.frame mit der Spalte für die IRT-Points

**Author(s)**

Harald Fiedler

---

detectItemID	<i>detectItemID</i>
--------------	---------------------

---

**Description**

Liefert die ItemID eines Balls/Stimulus zurück

**Usage**

```
detectItemID(Stimulus)
```

**Arguments**

Stimulus	ein data.frame mit den Spalten isMulitTarg, 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**

charactger der Länge 1, z.B. c("BL03")

**Author(s)**

Harald Fiedler

---

detectItemIDLive

*detectItemIDLive*

---

**Description**

Liefert live die ItemID eines Balls

**Usage**

```
detectItemIDLive(adrB, adrW)
```

**Arguments**

adrB                numeric

adrW                numeric

**Details**

Liefert die ItemID eines Balls/Stimulus zurück für eine REST-Anfrage des simFBN

**Value**

charactger 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 isMulitTarg, 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

---

erstelleRaschMatrixSkeleton	<i>erstelleRaschMatrixSkeleton</i>
-----------------------------	------------------------------------

---

**Description**

erstellt eine NA-Matrix mit den Sessions als Zeilenindex und den Item-Namen als Spaltenindex

**Usage**

```
erstelleRaschMatrixSkeleton(DF, ItemIDNamen)
```

**Arguments**

DF	data.frame auf Ballebene. Eine Spalte muss "idS".
ItemIDNamen	character array mit den Itembezeichnungen, für die eine Rasch-Matrix erstellt werden soll.

**Details**

Achtung: eine Spalte des data.frame muss den Spaltennamen "idS" haben.

**Value**

Eine Matrix voller NA, mit colnames=Itembezeichnungen und rownames=unique(idS)

**Author(s)**

Harald Fiedler

**Examples**

```
rm(list=ls())
DF <- data.frame(c("SessionA", "SessionB"), c(22, 90), c(23, 18), c(10,12))
colnames(DF) <- c("idS", "It1", "It2", "It_von_wo_ganz_anders")
ItemIDNamen <- c("Item1", "Item2", "Item3")
print(DF)
erstelleRaschMatrixSkeleton(DF=DF, ItemIDNamen=ItemIDNamen)
```

---

fillRaschMatrixSkeleton

*fillRaschMatrixSkeleton*


---

**Description**

Füllt die NA-Matrix mit 0 und 1, wo es zutreffend ist.

**Usage**

```
fillRaschMatrixSkeleton(DF, RaschMatrixSkeleton)
```

**Arguments**

DF	data.frame auf Ballebene, etwa per SQL2DF erworben
RaschMatrixSkeleton	matrix , belabeled mit SessionIDs und ItemIDs, wird etwa aus erstelleRaschMatrixSkeleton() erworben.

**Value**

Eine RaschMatrix mit vielen NA, und wenigen 0 und einigen 1en.

**Author(s)**

Harald Fiedler

---

getAdrWAlsListe

*getAdrWAlsListe*

---

**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`*getHW*

---

**Description**

Hilfsfunktion von augmentRAW 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

---

getNachname	<i>getNachname</i>
-------------	--------------------

---

**Description**

Hilfsfunktion von augmentRWA(): 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**

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

---

getSessionTimeStamp	<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")
```

---

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

---

**Description**

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

**Usage**

```
getVorname(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**

```
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**

<code>k</code>	numeric (besser wäre integer, sonst wird das Ergebnis korumpiert)
<code>digits</code>	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)
gibZahlFuehrendeNullen(k=k, digits=9)
```

---

```
implodeRaschMatrix4Quality
```

```
implodeFilledRaschMatrixSkeleton4Quality
```

---

**Description**

Lässt alle Probanden/Sessions weg, die zu viele NA haben, um eine sinnvolle ItemAnylse durchzuführen.

**Usage**

```
implodeRaschMatrix4Quality(RaschMatrixSkeletonFilled, MissingToleranz = 0.1)
```

**Arguments**

RaschMatrixSkeletonFilled

matrix bestehend aus vielen NA und einigen 0 und 1en, wie man sie aus fill-RaschMatrixSkeleton() erwirbt

MissingToleranz

numeric der Länge 1. Wie viel Prozent fehlende Bälle werden höchstens erlaubt.  
Default ist 10 Prozent

**Value**

matrix bestehend aus 0en und 1en und ganz ganz wenigen NA. Die Spalten tragen Item-Namen, und die Zeilen die Probandennamen, hier: idS

**Author(s)**

Harald Fiedler

---

isMultiTarget	<i>isMultiTarget</i>
---------------	----------------------

---

**Description**

Hilfsfunktion von augmentRAW(): 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	<i>istFormatNachnameKommaVorname</i>
-------------------------------	--------------------------------------

---

**Description**

Hilfsfunktion von augmentRAW()

**Usage**

```
istFormatNachnameKommaVorname(Spielername)
```

**Arguments**

Spielername	String
-------------	--------

**Details**

In den SQL-Auszügen des FBN finden sich Spielername vom Format "Fiedler, Harald", aber auch "A\_TEST\_Forschung". Die Funktion testet komponentenweise, ob zwei Strings kommagetrennt gepastet sind.

**Value**

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

**Author(s)**

Harald Fiedler

**Examples**

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

---

itemID2Params

*itemID2Params*


---

**Description**

Gibt Informationen an `simFBN()` zurück, wie der nächste Ball gespielt werden soll

**Usage**

```
itemID2Params(ItemID)
```

**Arguments**

ItemID                    charakter der Länge 1, welches den Namen des Items angibt. Implementiert sind BL01:BL32

**Details**

Für eine genauere Beschreibung verweise ich auf das Markdown-Manual für BL32MultiTargetSimTest.md. Diese Funktion erstellt den Knoten "nextB"

**Value**

list

**Author(s)**

Harald Fiedler

**Examples**

```
itemID2Params("BL03")
```

---

NormTree

*NormTree ist eine verschachtelte Liste*


---

**Description**

NormTree |\_\_ Altersgruppen/Mannschaften |\_\_ RAW : data.frame mit SessionID, Fiedler2016a, FBt, FBq |\_\_ HIST : list of counts and breaks |\_\_ Kernel |\_\_ FBq |\_\_ FBt |\_\_ Level |\_\_ FBqFBt

**Usage**

```
data(NormTree)
```

**Format**

A nested list about 1194 players

---

playedAngle

*playedAngle*


---

**Description**

Winkel zwischen zwei Adressen.

**Usage**

```
playedAngle(adrA, adrB)
```

**Arguments**

adrA                numeric Adressen der Ausgangsfelder

adrB                numeric Adressen der Zielfelder

**Details**

Gibt den Winkel zwischen zwei FBN-Adressen

**Value**

numeric mit Winkel aus -170:180 wobei der Winkel positiv im Uhrzeigersinn gemessen wird

**Author(s)**

Harald Fiedler

**Examples**

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

---

plotFBN

*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)
```

---

plotSeaShell

*plotSeaShell*

---

**Description**

Plottet die SeaShellGrafik

**Usage**

```
plotSeaShell(x, A, B, TitelA, TitelB, developerMode = FALSE,  
            gameover = FALSE)
```

**Arguments**

x	numeric Der Item-Response-Wert des Probanden
A	numeric Die Item-Response-Werte von Gruppe A
B	numeric die Item-Response-Werte von Gruppe B
TitelA	character mit der Headline für Gruppe A
TitelB	character mit der Headline für Gruppe B
developerMode	boolescher Wert, der eine Augenscheinkontrolle des Kernels und der Histogramms erlaubt.
gameover	boolischer Wert, der bei TRUE base64 kodierte Image wiedergibt

**Details**

Es handelt sich um die extra für die Normgruppenvergleiche von Hoffenheim entwickelte Grafik

**Value**

gameover=FALSE hat die Funktion keinen Ausgabe-wert, sonst base64 codiertes Image

**Author(s)**

Harald Fiedler

---

readAUGMENTED

*readAUGMENTED*


---

**Description**

Liest die per writeAUGMENTED() gespeicherten Dateien ein

**Usage**

```
readAUGMENTED(Dateiname,
  Pfad = "~/Dropbox (Cavorit)/Cavorit/Forschungsprojekte/Hoffenheim/RAW/")
```

**Arguments**

Dateiname	character der Länge 1 mit Dateiname (ohne Endung). Der Pfad wird automatisch auf die Dropbox gesetzt, genauer in den Ordner RAW vom Ordner Hoffenheim
Pfad	CharacterString mit Pfadangabe zur Dropbox

**Author(s)**

Harald Fiedler

**Examples**

```
#Dateiname = "RAW-2015-04"  
#head(readAUGMENTED(Dateiname = Dateiname))
```

---

readItemBank	<i>readItemBank</i>
--------------	---------------------

---

**Description**

Liest die ItemBank ein

**Usage**

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

**Arguments**

file	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)
```



---

readRAW	<i>readRAW</i>
---------	----------------

---

**Description**

Liest die per writeRAW() gespeicherten Dateien ein

**Usage**

```
readRAW(Dateiname,
        Pfad = "~/Dropbox (Cavorit)/Cavorit/Forschungsprojekte/Hoffenheim/RAW/")
```

**Arguments**

Dateiname	character der Länge 1 mit Dateiname (ohne Endung). Der Pfad wird automatisch auf die Dropbox gesetzt, genauer in den Ordner RAW vom Ordner Hoffenheim
Pfad	CharacterString mit Pfadangabe zur Dropbox

**Author(s)**

Harald Fiedler

**Examples**

```
Dateiname = "RAW-2015-04"
#head(readRAW(Dateiname = Dateiname))
```

---

writeAUGMENTED	<i>writeAUGMENTED</i>
----------------	-----------------------

---

**Description**

Der per augmentRAW() erzeugte Datensatz (ein data.frame) wird als R-Objekt in der Dropbox abgespeichert.

**Usage**

```
writeAUGMENTED(DF, Dateiname)
```

**Arguments**

DF	data.frame der per augmentRAW() erzeugte Datensatz
Dateiname	character der Länge 1, gibt den Dateinamen ohne Endung an. Der Pfad ist hard coded zur Dropbox

**Author(s)**

Harald Fiedler

---

`writeRAW`*writeRAW*

---

**Description**

Der per askDB() erzeugte Datensatz (ein data.frame) wird als R-Objekt in der Dropbox abgespeichert.

**Usage**

```
writeRAW(SQL, Dateiname)
```

**Arguments**

SQL	data.frame der per askDB() erzeugte Datensatz
Dateiname	character der Länge 1, gibt den Dateinamen ohne Endung an. Der Pfad ist hard coded zur Dropbox

**Author(s)**

Harald Fiedler

# Index

## \*Topic **datasets**

NormTree, [21](#)

## \*Topic **package**

Rbonaut2-package, [2](#)

adaptiv.BL16.fullRandom.modelPredict,  
[2](#)

adaptiv.BL16.fullRandom.modelTransform,  
[4](#)

adaptiv.BL16.modelPredict, [6](#)

askDB, [7](#)

augmentRAW, [8](#)

calcFiedler2016a, [9](#)

detectItemID, [9](#)

detectItemIDLIVE, [10](#)

detectItemResponse, [11](#)

erstelleRaschMatrixSkeleton, [11](#)

fillRaschMatrixSkeleton, [12](#)

getAdrWALsListe, [13](#)

getFirstAdrW, [14](#)

getHW, [14](#)

getNachname, [15](#)

getSessionTimeStamp, [16](#)

getVorname, [16](#)

gibZahlFuehrendeNullen, [17](#)

implodeRaschMatrix4Quality, [18](#)

isMultiTarget, [19](#)

istFormatNachnameKommaVorname, [19](#)

itemID2Params, [20](#)

NormTree, [21](#)

playedAngle, [21](#)

plotFBN, [22](#)

plotSeaShell, [22](#)

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

Rbonaut2-package, [2](#)

readAUGMENTED, [23](#)

readItemBank, [24](#)

readRAW, [25](#)

writeAUGMENTED, [25](#)

writeRAW, [26](#)