Institut zur Qualitätsentwicklung im Bildungswesen
Forschungsdatenzentrum
WISSENSCHAFTLICHE EINRICHTUNG DER LÄNDER
AN DER HUMBOLDT-UNIVERSITÄT ZU BERLIN E.V.

Replikationsanalysen mit eatRep

Sebastian Weirich und Benjamin Becker

Stand: 30.12.2020

POSTANSCHRIFT Humboldt-Universität zu Berlin | Institut zur Qualitätsentwicklung im Bildungswesen
Unter den Linden 6 | 10099 Berlin
Tel: +49 [30] 2093 46500 | Fax: +49 [30] 2093 46599
IQBOffice@iqb.hu-berlin.de | www.iqb.hu-berlin.de

KULTUSMINISTER KONFERENZ

Bibliographische Informationen

Weirich, S. & Becker, B. (2019). *Replikationsanalysen mit eatRep.* Berlin: IQB – Institut zur Qualitätsentwicklung im Bildungswesen. http://dx.doi.org/10.5159/IQB_Tutorial_Replikationsanalyse_v2

Replikationsanalysen mit eatRep

Sebastian Weirich und Benjamin Becker

30. Dezember 2020

Einleitung

In diesem Dokument sollen einige der Replikationsanalysen demonstriert werden, die bspw. für Berichte der IQB-Bildungstrendstudien mit dem R-Paket `eatRep` gerechnet werden können. Es geht dabei nicht um Itemkalibrierung oder das Ziehen der "Plausible Values", sondern um Analysen, die auf dem Gesamtanalysedatensatz (GADS) beruhen, also etwa die Bestimmung der Mittelwerte oder Kompetenzstufenbesetzungen der einzelnen Länder. Prinzipiell handelt es sich dabei um die Bestimmung deskriptiver Statistiken, Häufigkeitsverteilungen oder Parameter linearer bzw. logistischer Regressionen, für die drei Besonderheiten zu berücksichtigen sind:

1. Der Datensatz stellt keine (echte) Zufallsstichprobe aus der Population dar, da die Schülerinnen und Schüler nicht proportional zur Häufigkeit gezogen wurden, mit der sie in der Population vertreten sind. So ist in der Population der Anteil an Schülerinnen und Schülern aus Nordrhein-Westfalen größer als der Anteil von Schülerinnen und Schülern in Berlin; in der Stichprobe ist es jedoch umgekehrt.

2. Die Ziehungseinheiten (sampling units) in der Stichprobe sind Klassen, keine Individuen. Da sich Schüler innerhalb einer Klasse ähnlicher sind als Schüler verschiedener Klassen, ist die Stichprobe homogener als es eine echte Zufallsstichprobe gleicher Größe wäre.

3. Die interessierenden Variablen (Kompetenzen) sind latent, d.h. nicht direkt beobachtbar. Zudem enthalten viele (auch direkt beobachtete) Variablen fehlende Werte. Die Daten sind daher imputiert.

Das Paket `eatRep` ermöglicht die Bestimmung von (adjustierten) Mittelwerten und Mittelwertsdifferenzen, Häufigkeitsverteilungen, Perzentilen und Regressionen unter Berücksichtigung der geschachtelten und/oder imputierten Stichprobe. Auch Trendanalysen können gerechnet werden. Die oben genannten Besonderheiten werden in dem Paket wie folgt berücksichtigt:

1. Optionale Einbeziehung von individuellen Personengewichten

2. Optionale Verwendung von Replikationsmethoden (Bootstrap, Jackknife oder "Balanced repeated replicate"-Methoden)

3. Poolen der Ergebnisse entsprechend den Regeln von Rubin (1987).

Auf die theoretischen Grundlagen dieser Verfahren soll hier nicht eingegangen werden – sie werden in einer Vignette näher erläutert. Stattdessen sollen einige typische Analysen beispielhaft demonstriert wer-

den. Das Paket eignet sich ferner auch, wenn die Stichprobe nicht geschachtelt, sondern nur imputiert, bzw. nur geschachtelt und nicht imputiert ist. Mit anderen Worten, die drei oben genannten Methoden (Einbeziehung von Gewichten, Replikationsverfahren, Poolen) können unabhängig voneinander im Paket angewendet werden.

0. Installation des Pakets

Generell wird empfohlen, mit R Versionen 4.0 oder höher zu arbeiten. `eatRep` liegt ab der Version 0.13.4 auf CRAN und kann über folgende Befehlszeile installiert werden:

```
install.packages("eatRep")
```

Eine sehr kurze englischsprachige Einführung mit Literaturreferenzen erhält man über folgende Eingabe:

```
package?eatRep
```

1. Beispieldatensatz

Das Paket `eatRep` enthält einen Beispieldatensatz namens `lsa` (steht für "Large-Scale Assessment"), der in seiner Struktur eng an den Gesamtanalysedatensatz (GADS) der Bildungstrendstudien angelehnt und nur im Umfang reduziert ist. Sofern das Paket installiert ist, kann der Datensatz geladen und seine Struktur angezeigt werden:

```
library(eatRep)
data(lsa, package="eatRep")
str(lsa, give.attr = FALSE)
```

```
## 'data.frame':    90216 obs. of  22 variables:
##  $ year     : num  2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
##  $ idstud   : Factor w/ 7518 levels "P0001","P0002",..: 2362 175 1283 783 728 2122 732 2429
2899 2191 ...
##  $ wgt      : num  2.51 5.19 6.1 4.71 4.42 ...
##  $ jkzone   : num  18 86 79 5 3 8 3 20 13 12 ...
##  $ jkrep    : num  1 0 1 0 1 1 1 1 0 0 ...
##  $ imp      : num  3 3 2 2 2 2 1 2 3 2 ...
##  $ nest     : num  1 2 1 1 2 1 2 1 2 2 ...
##  $ country  : Factor w/ 3 levels "LandA","LandB",..: 2 3 2 3 3 2 3 2 1 2 ...
##  $ sex      : Factor w/ 2 levels "female","male": 2 2 1 2 2 2 1 2 1 1 ...
##  $ ses      : num  24.8 28.5 23.5 64.4 70.3 ...
##  $ mig      : logi  FALSE TRUE FALSE FALSE FALSE FALSE ...
##  $ domain   : Factor w/ 2 levels "listening","reading": 1 1 1 1 1 1 1 1 1 1 ...
##  $ score    : num  342 317 286 327 360 ...
##  $ comp     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ failMin  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ passReg  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ passOpt  : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ leScore  : num  1.21 1.21 1.21 1.21 1.21 ...
##  $ leComp   : num  0.00582 0.00582 0.00582 0.00582 0.00582 ...
##  $ leFailMin: num  0.00494 0.00494 0.00494 0.00494 0.00494 ...
##  $ lePassReg: num  0.00601 0.00601 0.00601 0.00601 0.00601 ...
##  $ lePassOpt: num  0.00141 0.00141 0.00141 0.00141 0.00141 ...
```

Wie der Gesamtanalysedatensatz in den Bildungstrendstudien ist auch der Beispieldatensatz im Langformat, das bedeutet, er besitzt mehr Zeilen als es Personen gibt: Wenn eine bestimmte Variable (z.B. der Migrationshintergrund `mig`) mehrfach imputiert wurde, tritt sie in dem Datensatz *nicht* in verschiedenen Variablen (z.B. `mig_Imp_1`, `mig_Imp_2`, `mig_Imp_3`, etc.) auf, sondern nur einmal als `mig`. Die verschiedenen Imputationen sind stattdessen untereinander angeordnet, und es gibt eine explizite Variable `imp`, die anzeigt, um die wievielte Imputation es sich handelt. Ebenfalls tritt eine Variable, die sich auf mehrere Kompetenzbereiche *und* mehrere Imputationen *und* mehrere Jahrgänge bezieht (z.B. der Kompetenzwert `score`), nicht mehrfach auf (also `lesen_2010_score_Imp_1`, `lesen_2010_score_Imp_2`, …, `hoeren_2010_score_Imp_1`, `hoeren_2015_score_Imp_1`, …), sondern die Variable `score` existiert nur einmal. `imp` gibt an, um die wievielte Imputation von `score` es sich handelt, und `domain` gibt an, um welchen Kompetenzbereich es sich handelt. Um Datensätze vom Lang- ins Wideformat und umgekehrt zu transformieren, kann das Paket `reshape2` verwendet werden. Alternativ bietet sich die Funktion `wideToLong` aus dem Paket `eatTools` an. Die Funktionsweise wird in Abschnitt 1.1 dieses Tutorials kurz erkäutert. Ferner ist das R-Paket `eatGADS` darauf zugeschnitten, aus dem sehr großen und folglich sehr unhandlichen Gesamtanalysedatensatz einen Teildatensatz mit den für die konkrete Analyse relevanten Variablen (auf der Lehrer- oder Schülerebene) zu erzeugen.

Der hier verwendete fiktive Beispieldatensatz enthält folgende Variablen:

- `year`: in welchem Jahr fand die hypothetische Erhebung statt? (2010 oder 2015)
- `idstud`: individuelle Personen-ID; es gibt insgesamt 7518 verschiedene Personen
- `wgt`: Fallgewicht des Schülers
- `jkzone`: Jackknife-Zone
- `jkrep`: Jackknife-Replicate
- `imp`: Nummer der Imputation (1, 2, oder 3)
- `nest`: Nummer der genesteten Imputation; für den Bildungstrend irrelevant
- `country`: fiktives Bundesland, aus dem der Schüler stammt (LandA, LandB, LandC)
- `sex`: Geschlecht des Schülers (male, female)
- `ses`: Sozioökonomischer Status
- `mig`: Indikator für das Vorliegen eines Migrationshintergrunds
- `domain`: Kompetenzbereich (listening oder reading)
- `score`: Punktwert auf der Bildungsstandards-Skala, den der Schüler in der jeweiligen Imputation in dem jeweiligen Kompetenzbereich erreicht hat
- `comp`: Kompetenzstufe für die jeweilige Imputation und den jeweiligen Kompetenzbereich
- `failMin`: Indikator: Hat der Schüler den Mindeststandard verfehlt?
- `passReg`: Indikator: Hat der Schüler den Regelstandard mindestens erreicht?

- `passOpt`: Indikator: Hat der Schüler den Optimalstandard erreicht?
- `leScore`: Linkingfehler für die `score`-Variable
- `leComp`: Linkingfehler für die Kompetenzstufen-Variable
- `leFailMin`: Linkingfehler für die Indikatorvariable, ob der Schüler den Mindeststandard verfehlt hat
- `lePassReg`: Linkingfehler für die Indikatorvariable, ob der Schüler den Regelstandard mindestens erreicht hat
- `lePassOpt`: Linkingfehler für die Indikatorvariable, ob der Schüler den Optimalstandard erreicht hat

Obschon mit über 90.000 Beobachtungen recht groß, bildet der Beispieldatensatz aus ökonomischen Gründen nur einen Ausschnitt des tatsächlichen GADS ab: So gibt es hier nur 3 (statt 15) Imputationen, nur 3 (statt 16) Bundesländer, und nur 2 (statt 8) Kompetenzbereiche. Teilweise sind für die Variablen im Datensatz auch Attribute definiert, die Variablen- und/oder Wertelabels enthalten:

```
attributes(lsa[,"year"])

## $varLabel
## [1] "year of assessment"
```

Da genestete Imputationen für den IQB-Bildungstrend keine Rolle spielen, kann der Datensatz nochmal auf nur die erste genestete Imputation reduziert werden:

```
bt <- lsa[which(lsa[,"nest"] == 1),]
```

1.1 Exkurs: Umformen von Wideformat-Datensätzen in das Langformat

Von öffentlichen Institutionen bereitgestellte Datensätze (z.B. PISA) liegen häufig im Wideformat vor – jede Zeile des Datensatzes entspricht dabei genau einer Person. Bevor diese Daten mit `eatRep` ausgewertet werden können, müssen sie in ein Langformat umgeformt werden. Diese Prozedur ist nur notwendig, wenn es sich um imputierte Datensätze handelt – in Wideformat-Datensätzen sind verschiedene Imputationen derselben Variable als zusätzliche Spalten enthalten, es gibt jedoch keine explizite Variable `imp` oder `imputation`. In Langformat-Datensätzen sind verschiedene Imputationen derselben Variable untereinander angeordnet, und eine Variable namens `imp` oder `imputation` gibt an, um die wievielte Imputation es sich jeweils handelt.

Um Datensätze umzuformen, sind in R verschiedene Prozeduren bereits implementiert, etwa in den Paketen `reshape2`, `tidyr` oder `data.table`. Darüber hinaus enthält das Paket `eatTools` ab Version 0.1.17 die Funktion `wideToLong` zur Umformung von Wideformat-Datensätzen in das von `eatRep` benötigte Langformat. Die Funktionsweise soll hier anhand eines typischen Wideformat-Beispieldatensatzes `data.timss3` aus dem Paket `BIFIEsurvey` illustriert werden. Der Datensatz kann analog wie der in `eatRep` enthaltene Beispieldatensatz geladen werden:

```
data(data.timss3, package="BIFIEsurvey")
str(data.timss3, give.attr = FALSE)
```

```
## 'data.frame':    4668 obs. of  20 variables:
##  $ IDSTUD : num  4e+08 4e+08 4e+08 4e+08 4e+08 ...
##  $ TOTWGT : num  17.5 17.5 17.5 17.5 17.5 ...
##  $ JKZONE : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ JKREP  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ female : num  1 0 1 1 1 1 1 1 0 0 ...
##  $ books  : num  3 3 5 3 3 2 4 3 3 4 ...
##  $ lang   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ migrant: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ scsci  : num  NA 2 2 1 2 3 2 2 1 1 ...
##  $ likesc : num  2 4 2 1 1 1 2 2 NA 1 ...
##  $ ASMMAT1: num  543 522 456 512 506 ...
##  $ ASSSCI1: num  600 512 497 584 533 ...
##  $ ASMMAT2: num  557 533 462 510 563 ...
##  $ ASSSCI2: num  578 519 545 614 568 ...
##  $ ASMMAT3: num  506 557 445 531 530 ...
##  $ ASSSCI3: num  570 554 528 569 564 ...
##  $ ASMMAT4: num  524 511 473 497 488 ...
##  $ ASSSCI4: num  560 506 550 597 483 ...
##  $ ASMMAT5: num  578 546 457 528 583 ...
##  $ ASSSCI5: num  607 565 546 623 578 ...
```

Der Datensatz enthält 4668 Zeilen entsprechend 4668 Personen. Von Interesse sind dabei folgende Variablen:

- `IDSTUD`: individuelle Personen-ID
- `TOTWGT`: Fallgewicht des Schülers
- `JKZONE`: Jackknife-Zone
- `JKREP`: Jackknife-Replicate
- `female`: Indikator für das Geschlecht des Schülers (1 = weiblich; 0 = männlich)
- `books`: Likert-Variable der Anzahl der Bücher im Haushalt
- `lang`: Wie oft wird zuhause die Instruktionssprache des Tests gesprochen?
- `migrant`: Indikator für den Migrationshintergrund
- `ASMMAT1`: Erste Imputation (bzw. erster plausible value) der Mathematikkompetenz
- `ASMMAT2`: Zweite Imputation (bzw. zweiter plausible value) der Mathematikkompetenz
- `ASMMAT3`: Dritte Imputation (bzw. dritter plausible value) der Mathematikkompetenz
- `ASMMAT4`: Vierte Imputation (bzw. vierter plausible value) der Mathematikkompetenz
- `ASMMAT5`: Fünfte Imputation (bzw. fünfter plausible value) der Mathematikkompetenz
- `ASSSCI1`: Erste Imputation (bzw. erster plausible value) der Naturwissenschaftskompetenz
- `ASSSCI2`: Zweite Imputation (bzw. zweiter plausible value) der Naturwissenschaftskompetenz
- `ASSSCI3`: Dritte Imputation (bzw. dritter plausible value) der Naturwissenschaftskompetenz
- `ASSSCI4`: Vierte Imputation (bzw. vierter plausible value) der Naturwissenschaftskompetenz
- `ASSSCI5`: Fünfte Imputation (bzw. fünfter plausible value) der Naturwissenschaftskompetenz

Obwohl auch die TIMSS-Daten imputiert sind, fehlt in diesem Datensatz eine explizite Imputationsvariable; wieviele Imputationen verwendet wurden, geht nur indirekt aus der Variablenbezeichnung hervor. Man erkennt dafür recht schnell, welche Variablen imputiert sind (z.B. `ASMMAT`; sie existiert fünfmal) und welche nicht (z.B. `female`, existiert nur einmal). Für die Umformung des Datensatzes ist es wichtig, dass die Anzahl der Imputationen (falls imputiert wurde) für sämtliche Variablen gleich ist. Sofern eine Variable 5 mal, eine andere 15 mal imputiert wurde, kann die Funktion nicht verwendet werden. Für den Aufruf müssen alle Variablen, die für die spätere Analyse benutzt werden sollen, angegeben werden – Variablen, die nicht von Interesse sind, können ignoriert werden. Im Funktionsaufruf muss zwischen den nicht-imputierten und den imputierten Variablen unterschieden werden:

```
library(eatTools)
timssLong <- wideToLong(datWide = data.timss3,
             noImp = c("IDSTUD", "TOTWGT", "JKZONE", "JKREP", "female"),
             imp = list ( mat = c("ASMMAT1", "ASMMAT2", "ASMMAT3", "ASMMAT4", "ASMMAT5"),
                          nawi = c("ASSSCI1", "ASSSCI2", "ASSSCI3", "ASSSCI4", "ASSSCI5")))
```

Die nicht imputierten Variablen müssen dabei als einfache Zeichenkette spezifiziert werden; die imputierten Variablen als eine Liste aus mindestens einer Zeichenkette (oder, wie hier im Beispiel: mehreren Zeichenketten). Die Listenstruktur definiert dabei, dass die Variable `mat` aus den fünf Imputationen `ASMMAT1`, `ASMMAT2`, `ASMMAT3`, `ASMMAT4` und `ASMMAT5` besteht. Der resultierende Datensatz `timssLong` besitzt nun das dasselbe Format wie der oben beschriebene Beispieldatensatz `lsa` und kann mit den entsprechenden `eatRep`-Funktionen ausgewertet werden. Falls Datensätze mit vielen Imputationen (z.B. 15) umgeformt werden müssen, kann der oben beschriebene Funktionsaufruf mühsam sein, wenn jede der `ASMMAT`- und `ASSSCI`-Variablen 15 mal hintereinander aufgelistet werden muss. Eine deutlich einfachere Variante ist dann, die Zeichenkette über den `paste0`-Befehl zu erzeugen:

```
timssLong <- wideToLong(datWide = data.timss3,
             noImp = c("IDSTUD", "TOTWGT", "JKZONE", "JKREP", "female"),
             imp = list ( mat = paste0("ASMMAT",1:5),
                          nawi = paste0("ASSSCI",1:5)))
```

Alternativ kann die Umformung auch mit der Funktion `melt` aus dem Paket `data.table` erfolgen:

```
library(data.table)
timssLong2<- data.table::melt(setDT(data.timss3),
               id = c("IDSTUD", "TOTWGT", "JKZONE", "JKREP", "female"),
               measure = patterns("^ASMMAT", "^ASSSCI"),
               value.name = c("mat", "nawi"), variable.name="imp")
```

2. Hauptfunktionen des Pakets “eatRep”

Das Paket enthält vier Hauptfunktionen entsprechend der vier im Bildungstrend gebräuchlichsten Analysen:

1. `repMean`: Berechnet Mittelwerte, Standardabweichungen sowie Differenzen von Mittelwerten oder Standardabweichungen.

2. `repTable`: Berechnet Häufigkeitsverteilungen und Differenzen von Häufigkeitsverteilungen; relevant bspw. zur Bestimmung der Anteile, wieviel Prozent der Schülerinnen und Schüler den Mindeststandard verfehlt, Regelstandard erreicht oder Optimalstandard erreicht haben.
3. `repQuantile`: zur Bestimmung von Quantilen, Perzentilen etc.
4. `repGlm`: Lineare und Log-lineare Regressionsmodelle.

2.1 Einfache Mittelwertsanalyse: ohne Trend, ohne Gruppendifferenzen, ohne cross-level Differenzen

Für das erste Beispiel soll angenommen werden, dass für einen Zeitpunkt (das Jahr 2010) und einen Kompetenzbereich ("reading") die Mittelwerte und Standardabweichungen der verschiedenen Länder einschließlich ihrer Standardfehler bestimmt werden sollen. Dazu muss der Datensatz "bt" (der Daten für 2010 *und* 2015 sowie für reading *und* listening enthält) nochmal reduziert werden – dies geschieht in R über sogenannte Subsetting-Prozeduren:

```
bt2010     <- bt[which(bt[,"year"] == 2010),]
bt2010read <- bt2010[which(bt2010[,"domain"] == "reading"),]
```

Für diesen reduzierten Datensatz wird nun die Funktion "repMean" aufgerufen:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = "country", dependent = "score",
           progress = FALSE)

## 1 analyse(s) overall according to: 'group.splits = 1'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
```

Als Gruppierungsvariable (Argument `groups`) wird hier die Ländervariable "country" verwendet. Das bewirkt, dass die interessierenden Statistiken (Mittelwerte, Standardabweichungen) separat für alle drei Länder bestimmt werden. Wichtig: die Personen im Datensatz müssen dabei in der Gruppierungsvariable genestet sein. Für "country" ist das der Fall: jede Person gehört genau zu einem Land. Für eine andere mögliche Gruppierungsvariable, den Kompetenzbereich (Variable "domain") ist das jedoch nicht der Fall, da ein und dieselbe Person Aufgaben mehrerer Kompetenzbereiche bearbeitet haben kann. Um zu prüfen, ob die Personen in einer Gruppierungsvariablen genestet sind, kann die Funktion "isNested" aus dem "lme4"-Paket genutzt werden:

```
lme4::isNested(bt2010[,"idstud"], bt2010[,"country"])

## [1] TRUE

lme4::isNested(bt2010[,"idstud"], bt2010[,"domain"])

## [1] FALSE
```

Möchte man die interessierenden Statistiken separat für beide Kompetenzbereiche ausgeben lassen, sollte man die entsprechende Variable `domain` nicht als Gruppierungsvariable verwenden – die entsprechende Funktion gibt dann eine Warnmeldung aus. Alternativ könnte man den Gesamtdatensatz in zwei Teildatensätze für "reading" und "listening" aufspalten und die Analyse für jeden Teildatensatz separat berechnen lassen. Einfacher geht es jedoch über einen Schleifenaufruf. Dieses Vorgehen wird im Abschnitt 2.5 dieses Tutorials demonstriert. Um die Ergebnisse in ein übersichtliches Format zu bringen, das auch nach Excel exportiert werden kann, muss die Reportingfunktion `report()` aufgerufen werden.

```
resReading <- report ( results, add = list( kb="reading"))
```

Das Argument "add" der Funktion `report()` bietet dabei die Möglichkeit, den Output um zusätzliche Spalten zu ergänzen. Die Funktion "weiß" ja nicht, dass die Analyse sich auf den Kompetenzbereich "lesen" bezieht; diese Information würde also andernfalls im Output nicht enthalten sein. Gegebenenfalls kann der Output um beliebig viele zusätzliche Spalten erweitert werden, z.B.:

```
resReading <- report ( results, add = list( kb="reading", year = "2010"))
```

Für die obenstehende Analyse wurde nur eine Gruppierungsvariable ("country"), ein Kompetenzbereich ("reading") und keinerlei Gruppenvergleiche spezifiziert. Der Output der Reportingfunktion ist daher noch sehr übersichtlich.

Die Ergebnisse können auch für mehrere Gruppierungsvariablen getrennt berechnet werden: Sollen die Mittelwerte und Standardabweichungen separat für jedes Land und jede Migrationsgruppe bestimmt werden, müssen beim Aufruf von `repMean()` zwei Gruppierungsvariablen definiert werden:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = c("country", "mig"), dependent = "score",
           progress = FALSE)

## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

Üblicherweise wird für den IQB-Bildungstrend – wenn die Mittelwerte nur für die jeweiligen Länder bestimmt werden sollen – zusätzlich der Gesamtmittelwert benötigt (der Mittelwert für Gesamtdeutschland). Es wäre möglich, die oben beschriebene Analyse zweimal auszuführen, und dabei einmal die Gruppierungsvariable "country" zu spezifizieren, und einmal keine Gruppierungsvariable anzugeben. Einfacher bzw. sparsamer ist es jedoch, nur einen Modellaufruf dafür zu verwenden:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = "country", group.splits = 0:1,
           dependent = "score", progress = FALSE)

## 2 analyse(s) overall according to: 'group.splits = 0 1'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                     NA
## 2               2               1           country                   NA
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

Das Argument `group.splits` definiert die Hierarchieebenen, für die die Berechnungen ausgeführt werden sollen. Hierarchieebenen geben an, wie stark (nach Gruppen) ausdifferenziert eine Analyse ist: Es gibt dabei immer eine Hierarchieebene mehr, als es Gruppierungsvariablen gibt; eine Gruppierungsvariable entspricht zwei Ebenen, zwei Gruppierungsvariablen drei Ebenen, und keine Gruppierungsvariable nur einer – der "nullten" Ebene. Auf der "nullten" Ebene findet keine Ausdifferenzierung statt; der Mittelwert (oder was auch immer) wird stets für die Gesamtgruppe berechnet. Bei einer Gruppierungsvariablen (z.B. `country`) gibt es zwei Ebenen: die Gesamtgruppe ("nullte" Ebene: alle Länder gemeinsam) und die erste Ebene: separiert nach Teilgruppen (LandA vs. LandB vs. LandC). Bei zwei Gruppierungsvariablen (Bsp. `country` und `mig`) gibt es drei Ebenen: die Gesamtgruppe ("nullte" Ebene: keine Ausdifferenzierung, also alle Länder und alle Migrationsgruppen gemeinsam), die "erste" Ebene, in der die Analysen jeweils nach einer Gruppierungsvariable ausdifferenziert sind (LandA vs. LandB vs. LandC unter Einbeziehung aller Migrationsgruppen; sowie Migration vs. nicht-Migration unter Einbeziehung aller Länder), und die "zweite" Ebene, in der die Analysen *nach beiden Gruppierungsvariablen ausdifferenziert* werden, also Migranten in LandA vs. Migranten in LandB, vs. Migranten in LandC vs. Nicht-Migranten in LandA vs. Nicht-Migranten in LandB, vs. Nicht-Migranten in LandC.

`group.splits` ist ein numerischer Vektor, der nun die Hierachriebenen enthält, für die Mittelwerte berechnet werden sollen. Wird das Argument nicht spezifiziert, wird standardmäßig nur die höchste Hierarchieebene berechnet – also bei einer Gruppierungsvariablen nur die Ergebnisse differenziert für diese Variable (ohne Gesamtgruppe), und bei zwei Gruppierungsvariablen differenziert nach beiden Gruppierungsvariablen (also ohne Gesamtgruppe und ohne Ausdifferenzierung nach nur einer Gruppierungsvariablen).

Gibt man bei nur einer Gruppierungsvariablen `group.splits = c(0,1)` oder `group.splits = 0:1` an, so wird zusätzlich die "nullte" Ebene (Gesamtgruppe) berechnet. Gibt man bei zwei Gruppierungsvariablen `group.splits = 1:2` an, so wird die "erste" und die "zweite", nicht aber die unterste Ebene ("nullte" Ebene oder Gesamtgruppe) berechnet. Um pauschal alle Ebenen zu berechnen, kann man `group.splits = 0:x` angeben, wobei "x" die Anzahl der Gruppierungsvariablen ist.

2.2 Mittelwertsanalyse: Gruppendifferenzen

Mitunter sollen im IQB-Bildungstrend nicht nur Mittelwerte getrennt nach bestimmten Gruppierungsvariablen berechnet werden, sondern es soll auch geprüft werden, ob sich die Mittelwerte bestimmter Gruppen unterscheiden – ob also ihre Differenz signifikant verschieden von Null ist. So etwa im Kapitel für "Geschlechtsbezogene Disparitäten", wo es darum geht, ob sich Jungen und Mädchen in ihren erreichten Kompetenzen signifikant unterscheiden. Differenzen werden aber auch in Kapiteln wie "Mittelwerte und Streuungen der erreichten Kompetenzen im Ländervergleich" benötigt, wenn etwa angegeben werden soll, ob sich der Mittelwert eines bestimmten Landes signifikant vom deutschen Gesamtmittelwert unterscheidet. Beide hier angegebenen Beispiele unterscheiden sich insofern, als es sich im ersten Fall (Jungen vs. Mädchen) um Differenzen *innerhalb einer Hierarchieebene* handelt, im zweiten Fall (LandA vs. Gesamtmittelwert) um Differenzen zwischen zwei Hierarchieebenen (das erkennt man auch daran, dass "LandA" selbst Teil der Gesamtpopulation ist, gegen dessen Wert es verglichen werden soll).

Differenzen innerhalb einer Hierarchieebene sollen daher im Folgenden als "Gruppendifferenzen", Differenzen zwischen zwei Hierarchieebenen als "cross-level Differenzen" bezeichnet werden. Um Gruppendifferenzen bestimmen zu können, muss immer mindestens eine Gruppierungsvariable angegeben sein. Folgender Syntaxaufruf erlaubt die Bestimmung von Geschlechterunterschieden in der Gesamtpopulation:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = "sex", group.differences.by = "sex",
           dependent = "score", progress = FALSE)

## 1 analyse(s) overall according to: 'group.splits = 1'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

Das Argument `group.differences.by` spezifiziert diejenige Gruppierungsvariable, für die Differenzen bestimmt werden sollen. Was in `group.differences.by` steht, muss daher eine Teilmenge dessen sein, was in `groups` steht. In `group.differences.by` darf dabei jedoch immer nur eine Variable stehen. Technisch funktioniert die Bestimmung von Gruppendifferenzen so, dass für die in `group.differences.by` angegebene Gruppierungsvariable alle paarweisen Kontraste bestimmt und dafür Differenzen berechnet werden. Für eine dichotome Variable mit nur zwei Ausprägungen (Jungen, Mädchen) gibt es nur einen Kontrast (Jungen vs. Mädchen). Für eine polytome Variable mit drei Ausprägungen (LandA, LandB, LandC) gibt es drei Konstraste (LandA vs. LandB, LandA vs. LandC, LandB vs. LandC). Durch eine polytome Variable mit vier Ausprägungen werden sechs Kontraste definiert, usw. Werden in `groups` mehrere Variablen definiert, kann in `group.differences.by` angegeben werden, für welche dieser Variablen Gruppendifferenzen bestimmt werden sollen. Sollen beispielsweise Geschlechterdifferenzen für jedes Land separat berechnet werden:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = c("country", "sex"),
           group.differences.by = "sex", dependent = "score", progress = FALSE)

## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

Sollen Geschlechterdifferenzen für jedes Land separat *und* zusätzlich für die Gesamtgruppe berechnet werden:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = c("country", "sex"), group.splits = 0:2,
           group.differences.by = "sex", dependent = "score", progress = FALSE)

## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                   <NA>
## 2               2               1           country                 <NA>
## 3               3               1               sex                  sex
## 4               4               2     country + sex                  sex
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

Sollen umgekehrt Länderdifferenzen für jede Geschlechtsgruppe separat und zusätzlich für die Gesamtgruppe berechnet werden:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = c("country", "sex"), group.splits = 0:2,
           group.differences.by = "country", dependent = "score", progress = FALSE)

## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                   <NA>
## 2               2               1           country              country
## 3               3               1               sex                 <NA>
## 4               4               2     country + sex              country
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

2.3 Mittelwertsanalyse: cross-level Differenzen

Im einfachsten Fall – das heißt, mit nur einer Gruppierungsvariablen und also nur zwei Hierarchieebenen (der "nullten" und der "ersten" Ebene) – werden cross-level Differenzen als Differenz einer bestimmten Gruppe zur Gesamtgruppe verstanden. Bei zwei oder mehr Gruppierungsvariablen und demzufolge drei oder mehr Hierarchieebenen können cross-level Differenzen für eine bestimmte Hierarchieebene jeweils in Relation zu *allen* übergeordneten Hierarchieebenen bestimmt werden. Das allerdings wird schnell recht unübersichtlich. Stellt man sich das Beispiel der Gruppierungsvariablen `country` (3-stufig) und `mig` (2-stufig) vor, so könnten folgende 23 cross-level Differenzen bestimmt werden:

Ebene 2 vs. Ebene 1:

- Migranten in LandA vs. Migranten
- Migranten in LandB vs. Migranten
- Migranten in LandC vs. Migranten
- Nicht-Migranten in LandA vs. Nicht-Migranten
- Nicht-Migranten in LandB vs. Nicht-Migranten
- Nicht-Migranten in LandC vs. Nicht-Migranten
- Migranten in LandA vs. LandA
- Migranten in LandB vs. LandB
- Migranten in LandC vs. LandC
- Nicht-Migranten in LandA vs. LandA
- Nicht-Migranten in LandB vs. LandB
- Nicht-Migranten in LandC vs. LandC

Ebene 1 vs. Ebene 0:

- Migranten vs. Gesamtgruppe
- Nicht-Migranten vs. Gesamtgruppe
- LandA vs. Gesamtgruppe
- LandB vs. Gesamtgruppe
- LandC vs. Gesamtgruppe

Ebene 2 vs. Ebene 0:

- Migranten in LandA vs. Gesamtgruppe
- Migranten in LandB vs. Gesamtgruppe
- Migranten in LandC vs. Gesamtgruppe
- Nicht-Migranten in LandA vs. Gesamtgruppe
- Nicht-Migranten in LandB vs. Gesamtgruppe
- Nicht-Migranten in LandC vs. Gesamtgruppe

Jede cross-level Differenz setzt dabei zwei Hierarchieebenen zueinander in Beziehung. Hierachieebenen sind zueinander benachbart, wenn sich sich in ihrer Differenz genau um den Wert 1 unterscheiden. Der Vergleich von Ebene 2 vs. Ebene 1 bzw. Ebene 1 vs. Ebene 0 betrachtet benachbarte Ebenen, der Vergleich Ebene 2 vs. Ebene 0 jedoch nicht. Allgemein gilt, dass cross-level Differenzen nur dann bestimmt werden können, wenn im Funktionsaufruf `group.splits` als ein Vektor von wenigstens zwei Elementen spezifiziert wurde. Um die Menge der möglichen cross-level Differenzen nicht ausufern zu lassen, gibt es die Möglichkeit, über ein zusätzliches Argument `cross.differences` zu definieren, für welche Paare von Hierarchieebenen die Differenzen bestimmt werden sollen.

Dazu ein Beispiel: Es sollen wieder die beiden Gruppierungsvariablen `country` (3-stufig) und `mig` (2-stufig) betrachtet werden. Mittelwerte und Standardabweichungen sollen jeweils für alle drei Hierarchieebenen berechnet werden. Gruppendifferenzen sollen für die Ländervariable berechnet werden (also LandA vs. LandB, LandA vs. LandC, sowie LandB vs. LandC). Cross-level Differenzen sollen stets in Relation zur Gesamtgruppe berechnet werden, also für Ebene 1 vs. Ebene 0 und Ebene 2 vs. Ebene 0. Der Syntaxaufruf dafür lautet folgendermaßen:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = c("country", "sex"), group.splits = 0:2,
           group.differences.by = "country", cross.differences = list(c(0,1), c(0,2)),
           dependent = "score", progress = FALSE)

## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                   <NA>
## 2               2               1           country               country
## 3               3               1               sex                   <NA>
## 4               4               2     country + sex               country
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

`cross.differences` verlangt eine Liste von Vektoren (minimal ein Vektor). Jeder Vektor darf genau zwei Ziffern enthalten, die die beiden Hierarchieebenen angeben, für die cross-level Differenzen berechnet werden sollen. Der Einfachheit halber können mit dem Aufruf `cross.differences = TRUE` sämtliche cross-level Differenzen ausgegeben werden, wohin bei `cross.differences = FALSE` keinerlei cross-level Differenzen berechnet werden.

Die Argumente `group.differences.by` und `cross.differences` erlauben in Kombination auch, cross-level Differenzen von Gruppendifferenzen zu bestimmen – etwa wenn geprüft werden soll, ob sich die Differenz "Jungen vs. Mädchen" in "LandA" bedeutsam von der Differenz "Jungen vs. Mädchen" in der Gesamtgruppe unterscheidet:

```
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = c("country", "sex"), group.splits = 0:2,
           group.differences.by = "sex", cross.differences = TRUE, dependent = "score",
           progress = FALSE)

## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                   <NA>
## 2               2               1           country                 <NA>
## 3               3               1               sex                  sex
## 4               4               2     country + sex                  sex
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## Compute cross level differences using 'wec' method. Assume heteroscedastic variances.
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 39 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 46 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 39 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

Das Outputobjekt `res` ist nun schon umfangreicher. cross-level Differenzen von Gruppendifferenzen sind in der Spalte `comparison` durch die Angabe `crossDiff_of_groupDiff` gekennzeichnet.

2.4 Anmerkungen technischer Natur

Anders als bei Gruppenvergleichen (z.B. "Jungen vs. Mädchen") sind die Gruppen bei cross-level Differenzen nicht unabhängig – so ist ja beispielsweise `LandA` in der Gesamtgruppe enthalten. Anstelle des *t*-Tests für unabhängige Gruppen können die cross-level Differenzen über "weighted effect coding" oder Replikations- beziehungsweise Bootstrapverfahren bestimmt werden. Die Standardmethode in `eatRep` ist ab Version 0.10.0 "weighted effect coding" (wec). Alternative Methoden können über das Argument `crossDiffSE` gewählt werden. In früheren Versionen von `eatRep` war die Methode `old` das Standardverfahren. Hierbei wurde ein konventioneller *t*-Tests durchgeführt, der jedoch aufgrund der nicht berücksichtigten Abhängigkeit zu konservativ ausfällt.

2.5 Mittelwertsanalyse: Trends

Prinzipiell sind (IQB-Bildungs-)Trends ja auch nichts anderes als Gruppendifferenzen. Die Personen sind in der Trendvariablen genestet (jede Person gehört zu genau einem Messzeitpunkt), und es ist grundsätzlich egal, ob man in einem Datensatz Jungen gegen Mädchen (jeweils zum gleichen Zeitpunkt) vergleicht, oder Testteilnehmer von 2015 gegen Testteilnehmer von 2010. Der wesentliche Unterschied besteht darin, dass im ersten Fall (Jungen gegen Mädchen zum gleichen Messzeitpunkt) die zugrundeliegende Itemstichprobe für beide Personengruppen identisch ist, was im zweiten Fall (2015 vs. 2010) nicht zwangsläufig so sein muss. Auch ist es bei Vergleichen über die Zeit möglich – selbst wenn exakt dieselben Items verwendet wurden –, dass diese Items in die Jahre gekommen sind und 2015 (geringfügig) anders funktionieren als 2010. Sofern dieses "Andersfunktionieren" zufallsbedingt und nicht systematisch ist, spielt es vor allem für den Standardfehler der Differenzschätzung eine Rolle. Für die Bestimmung des Standardfehlers von Trendvergleichen wird daher das zufallsbedingte "Andersfunktionieren" der jeweiligen Itemstichproben als Linkingfehler berücksichtigt. Das ist bei normalen Gruppenvergleichen nicht üblich.

Technisch funktioniert die Bestimmung von Trends folgendermaßen: Die Analyse wird für beide Trendgruppen separat durchgeführt, also in unserem Beispiel für 2010 und 2015. Anschließend werden für jede Kombination aus der (oder den) Gruppierungsvariablen bspw. für Mittelwerte die Differenzen $\overline{m}_{2015} - \overline{m}_{2010}$ berechnet. Der Standardfehler jeder Differenz (bzw. der Standardfehler des Trends) berechnet sich:

$$SE_{trend} = \sqrt{SE_{2010}^2 + SE_{2015}^2 + SE_{link}^2} \quad (1)$$

Trends können dabei für einfache Mittelwerte, Gruppendifferenzen und cross-level Differenzen bestimmt werden. Zur Veranschaulichung soll die zuletzt durchgeführte Analyse noch einmal – diesmal jedoch mit zusätzlicher Bestimmung von Trends – durchgeführt werden. Der bislang für alle Analysen be-

nutzte Teildatensatz `bt2010read` kann nicht länger verwendet werden, da er nur die Daten für das Jahr 2010 enthält. Ein Datensatz für den Kompetenzbereich “Lesen” mit den Daten von 2010 und 2015 kann als Teildatensatz von `bt` erzeugt werden. Im Funktionsaufruf wird nun zusätzlich die Trendvariable `trend = "year"` sowie der Linkingfehler `linkErr = "leScore"` definiert. Wird kein Linkingfehler angegeben, wird er standardmäßig auf 0 gesetzt.

```
btread  <- bt[which(bt[,"domain"] == "reading"),]
results <- repMean(datL = btread, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = c("country", "sex"), group.splits = 0:2,
           group.differences.by = "country", cross.differences = list(c(0,1), c(0,2)),
           dependent = "score", trend = "year", linkErr = "leScore", progress = FALSE)

##
## Trend group: '2010'
## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                    <NA>
## 2               2               1           country             country
## 3               3               1               sex                 <NA>
## 4               4               2     country + sex             country
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                    <NA>
## 2               2               1           country             country
## 3               3               1               sex                 <NA>
## 4               4               2     country + sex             country
##
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
##
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.

## Note: No linking error was defined. Linking error will be defaulted to '0'.
```

```
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.

## Note: No linking error was defined. Linking error will be defaulted to '0'.

res     <- report(results, add = list(kb="reading"))
```

2.6 Schleifen über nicht-genestete (Gruppierungs-)variablen

Die Argumente `groups` und `group.splits` erlauben, Analysen für verschiedene Gruppen und Hierarchieebenen mit nur einem Funktionsaufruf durchzuführen. Alternativ könnte man auch die Funktion `repMean` einmal für die Gesamtgruppe, einmal getrennt nach Ländern, und einmal getrennt nach Geschlechtsgruppen ausführen (was allerdings umständlich wäre, insbesondere bei mehrfach gestuften Gruppierungsvariablen, z.B. 16 Ländern). Das Argument `groups` verlangt jedoch, dass die Personen der Stichprobe in der oder den Gruppierungsvariable(n) genestet sind. Für den Kompetenzbereich ist das nicht der Fall – die Variable `domain` kann hier also nicht als Gruppierungsvariable benutzt werden. (Gleiches würde für eine potenzielle Variable `Schulfach` gelten.) Da im IQB-Bildungstrend die Analysen häufig für viele verschiedene Fächer und/oder Kompetenzbereiche durchzuführen sind, würde das bedeuten, für jeden Kompetenzbereich einen separaten Funktionsaufruf definieren zu müssen. Alternativ gibt es die Möglichkeit, über eine Schleife die Analyse in einem Funktionsaufruf für mehrere Kompetenzbereiche auszuführen. Im Beispieldatensatz `bt` soll das anhand der zwei darin enthaltenen Kompetenzbereiche “listening” und “reading” demonstriert werden (bisher wurde in dem Teildatensatz `btread` bzw. `bt2010read` nur der Kompetenzbereich “reading” isoliert betrachtet). Inhaltlich wird auf dasselbe Beispiel wie zuvor zurückgegriffen, also eine Analyse mit Gruppen- und cross-level Differenzen und Trendschätzung.

```
results <- by ( data = bt, INDICES = bt[,"domain"], FUN = function ( teildatensatz) {
           repMean(datL = teildatensatz, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
                    repInd = "jkrep", imp="imp", groups = c("country", "sex"),
                    group.splits = 0:2, group.differences.by = "country",
                    cross.differences = list(c(0,1), c(0,2)), dependent = "score",
                    trend = "year", linkErr = "leScore", progress = FALSE) } )

##
## Trend group: '2010'
## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                   <NA>
```

```
## 2               2               1           country              country
## 3               3               1               sex                 <NA>
## 4               4               2     country + sex              country
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                   <NA>
## 2               2               1           country              country
## 3               3               1               sex                 <NA>
## 4               4               2     country + sex              country
##
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
##
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.

## Note: No linking error was defined. Linking error will be defaulted to '0'.

##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.

## Note: No linking error was defined. Linking error will be defaulted to '0'.

##
## Trend group: '2010'
## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
```

```
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                   <NA>
## 2               2               1           country              country
## 3               3               1               sex                 <NA>
## 4               4               2     country + sex              country
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 4 analyse(s) overall according to: 'group.splits = 0 1 2'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by
## 1               1               0                                   <NA>
## 2               2               1           country              country
## 3               3               1               sex                 <NA>
## 4               4               2     country + sex              country
##
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
##
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.

## Note: No linking error was defined. Linking error will be defaulted to '0'.

##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.

## Note: No linking error was defined. Linking error will be defaulted to '0'.
```

Um den eigentlichen Funktionsaufruf `repMean` wird eine `by`-Schleife gelegt. Sie definiert für den Gesamtdatensatz `bt` eine Filtervariable namens “domain”. Der Datensatz wird also in soviele Teildatensätze

zerlegt, wie es Kompetenzbereiche gibt (hier sind es nur zwei). Für jeden dieser Teildatensätze ("listening" und "reading") wird nun der einmal definierte Funktionsaufruf ausgeführt. Das Objekt "results" ist nun eine Liste, die aus zwei Elementen besteht. Das erste Element heißt "listening" und enthält die Ergebnisse für "listening". Das zweite Element heißt "reading" und enthält die Ergebnisse für "reading". Für die Reportingfunktion muss nun ebenfalls eine Schleife über diese beiden Analysen definiert werden, die nun getrennt aufbereitet werden sollen. Hier wird auch deutlich, wozu das Argument `add` benötigt wird. Im ersten Schritt soll die Aufbereitung ohne die Verwendung einer Schleife demonstriert werden:

```
names(results)

## [1] "listening" "reading"

resultsListening <- report(results[["listening"]], add = list ( kb = "listening"))
resultsReading   <- report(results[["reading"]], add = list ( kb = "reading"))
alleResults1     <- rbind ( resultsListening, resultsReading)
```

Unter Verwendung einer Schleife ist der Syntaxaufruf sparsamer, insbesondere, wenn mehr als zwei Kompetenzbereiche in der Analyse betrachtet wurden:

```
alleResults2     <- lapply(names(results), FUN = function ( x ) {
                          report(results[[x]], add = list(kb=x))})
alleResults2     <- do.call("rbind", alleResults2)
```

2.7 Adjustierte Mittelwerte

Ab Version 0.13.0 erlaubt das Paket eatRep auch die Berechnung sogenannter "fairer" oder adjustierter Mittelwerte. Auf die theoretischen Hintergründe dieser Adjustierungsmethoden soll hier nicht eingegangen werden; grob vereinfacht kann man sich die Logik der Adjustierung in etwa so vorstellen: ein Vergleich der mittleren Kompetenzen zweier Bundesländer, etwa Bayern und Bremen, ist insofern schwer interpretierbar, als dass die Zusammensetzung der Schülerschaft (etwa was Migrationshintergrund, SES, etc. angeht) in beiden Ländern unterschiedlich ist. Aufgrund unterschiedlicher demografischer Hintergründe bestehen zwischen den Schülerpopulationen also bereits *vor der Einschulung* erhebliche Unterschiede. Für die Bestimmung adjustierter Mittelwerte "wird nun so getan", als bestünden diese Unterschiede nicht. Dazu wählt man eine (prinzipiell willkürliche) Referenz, beispielsweise Gesamtdeutschland. Anschließend kann berechnet werden, wie die Mittelwerte für Bremen und Bayern ausfallen würden, wenn die Zusammensetzung ihrer Schülerschaft genauso wäre wie die Zusammensetzung der Schülerschaft in Gesamtdeutschland. Dabei kann unterschieden werden, für welche demografischen Variablen adjustiert werden soll: für Migrations- oder Sprachhintergrund, Hisei, Pared, oder mehrerer dieser Variablen.

Im Folgenden sollen adjustierte Ländermittelwerte für den Kompetenzbereich "reading" im Jahr 2010 berechnet werden, wobei für die Variablen `Geschlecht`, `Migrationshintergrund` und `SES` adjustiert wird. Wichtig ist dabei, dass alle Adjustierungsvariablen numerisch sein müssen. Im Falle polytomer Gruppierungsvariablen (bspw. Sprache zuhause: "deutsch", "nicht deutsch", "deutsch und eine andere Sprache")

müssten hier zuvor dichotome Indikatorvarablen definiert werden. Im folgenden Syntaxbeispiel werden die beiden nicht-numerischen Adjustierungsvariablen `sex` und `mig` zuvor numerisch konvertiert:

```
sapply(bt2010read[,c("sex", "mig", "ses")], class)

##       sex       mig       ses
##  "factor" "logical" "numeric"

bt2010read[,"sexnum"] <- car::recode(bt2010read[,"sex"], "'male'=0; 'female'=1",
                         as.factor = FALSE)
bt2010read[,"mignum"] <- as.numeric(bt2010read[,"mig"])
results <- repMean(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
           repInd = "jkrep", imp="imp", groups = "country", group.splits = 0:1,
           cross.differences = TRUE, adjust = c("sexnum", "mignum", "ses"),
           dependent = "score", progress = FALSE)

## 2 analyse(s) overall according to: 'group.splits = 0 1'.
##
##   analysis.number hierarchy.level groups.divided.by group.differences.by adjust
## 1               1               0                                     NA  FALSE
## 2               2               1           country                   NA   TRUE
##
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res <- report(results, add = list( kb="reading"))
```

Mehrere Dinge müssen dabei bedacht werden: Cross-level Differenzen können für adjustierte Mittelwerte vorerst nur mit der Methode `old` bestimmt werden. In der Gesamtgruppe findet keine Adjustierung statt, bzw. da ja eben *für die Zusammensetzung der Hintergrundvariablen in der Gesamtgruppe* adjustiert wird, wären hier der adjustierte und der unadjustierte Mittelwert identisch.

Sollen zusätzlich Trends berechnet werden, kann nun *nicht* so ohne weiteres der oben stehende Syntaxaufruf um die Argumente zur Trendbestimmung erweitert werden, wie das für die unadjustierten Mittelwerte der Fall war. Wenn in Trendanalysen beispielsweise der Mittelwert eines Landes aus dem Jahr 2010 mit dem Mittelwert desselben Landes aus dem Jahr 2015 verglichen wird, ist die Referenz ja nicht länger Gesamtdeutschland. Während unadjustierte Mittelwerte in der Regel unabhängig von der jeweiligen Fragestellung berechnet werden können, hängt die Adjustierung immer von der konkreten Fragestellung ab: für Trendanalysen muss anders adjustiert werden als für Analysen innerhalb eines Messzeitpunkts.

Oben wurde die Fragestellung für Analysen innerhalb eines Messzeitpunkts skizziert: wie würden die Mittelwerte für Bremen und Bayern ausfallen, wenn die Zusammensetzung ihrer Schülerschaft genauso wäre wie die Zusammensetzung der Schülerschaft in Deutschland? Für Trendanalysen könnte die Fragestellung lauten: Wie wäre der Mittelwertsunterschied zwischen 2010 und 2015 *für ein bestimmtes Bundesland*, wenn die Zusammensetzung der Schülerschaft über die Jahre konstant geblieben wäre? Im Paket `eatRep` wird momentan noch nicht zwischen verschiedenen Fragestellungen differenziert. Um adjustierte Trends zu bestimmen, muss daher das, was bislang immer als Trendvariable behandelt wurde,

nämlich `year`, nun als Gruppierungsvariable behandelt werden. Sollen adjustierte Trends für verschiedene Gruppen (bspw. verschiedene Länder) berechnet werden, muss die Aufteilung in länderspezifische Teildatensätze von hand bzw. über eine Schleife erfolgen. Die Berücksichtigung des Linkingfehlers, die sonst innerhalb der Trendfunktion geschieht, muss nun ebenfalls von hand erfolgen. In der folgenden Syntax wird daher die Schleife von Hand konstruiert (über die `by`-Funktion), und der Standardfehler des Trends wird von Hand berechnet – als die Wurzel der Summe der quadrierten Standardfehler für 2010, 2015 und des Links:

```
btread  <- bt[which(bt[,"domain"] == "reading"),]
btread[,"sexnum"] <- car::recode(btread[,"sex"], "'male'=0; 'female'=1", as.factor = FALSE)
btread[,"mignum"] <- as.numeric(btread[,"mig"])
btread[,"year"] <- as.integer(btread[,"year"])
results <- by(data = btread,  INDICES = btread[,"country"], FUN = function(sub.dat) {
           res <- repMean(datL = sub.dat, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
                  repInd = "jkrep", imp="imp", groups = c("year"),
                  adjust = c("sexnum", "mignum", "ses"), dependent = "score",
                  progress = FALSE)
           res <- report(res, add = list( kb="reading",
                                          country= as.character(sub.dat[1,"country"])))
           res[,"trend"]   <- diff(res[,"est"])
           res[,"trendSE"] <- sqrt(sum(res[,"se"]^2) + unique(sub.dat[,"leScore"])^2)
           return(res)})

## 1 analyse(s) overall according to: 'group.splits = 1'.
## Assume unnested structure with 3 imputations.
## Create 67 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 1'.
## Assume unnested structure with 3 imputations.
## Create 98 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 1'.
## Assume unnested structure with 3 imputations.
## Create 65 replicate weights according to JK2 procedure.

results <- do.call("rbind", results)
```

3. Häufigkeitsanalysen (repTable)

Einfache univariate Häufigkeitsanalysen polytomer Variablen, so wie sie in den Bildungstrendstudien des IQB verwendet werden, sind im Grunde auch bloß Mittelwertsanalysen dichotomer Indikatoren dieser polytomen Variablen. In vielen (nicht allen) Fällen bräuchte man eine Funktion wie `repTable` gar nicht; man könnte bspw. für die fünffach gestufte Kompetenzstufenvariable (1, 2, 3, 4, 5) fünf Indikatorvariablen definieren und deren Häufigkeit jeweils mit `repMean` bestimmen. Der wichtigste Unterschied zwischen Häufigkeits- und Mittelwertsanalysen besteht in der zugrundeliegenden Statistik für Gruppendifferenzen: Für Mittelwertsanalysen wird i.d.R. ein einfacher t-Test für unabhängige Stichproben durchgeführt (“Ist der Mittelwert für Jungen signifikant verschieden von dem Mittelwert für Mäd-

chen?"). Für Häufigkeitsanalysen wird üblicherweise der χ^2-Test verwendet ("Unterscheidet sich die Kompetenzstufenverteilung für Jungen von der Kompetenzstufenverteilung für Mädchen?"). Man kann theoretisch auch bei Häufigkeitsanalysen für jede einzelne Kompetenzstufe 1, 2, 3, 4, 5 in einem separaten Test prüfen, ob die Häufigkeit dieser bestimmten Stufe für Jungen signifikant verschieden von der Häufigkeit für Mädchen ist. Zu beachten ist, dass diese fünf Vergleiche nicht unabhängig voneinander sind und hier ggf. eine Bonferroni-Korrektur angewendet werden sollte. Im Folgenden sollen beide Verfahren kurz demonstriert werden:

```
### erstes Beispiel: Gruppenvergleiche mit chi^2-Test: pruefe fuer jedes Land,
### ob die Kompetenzstufenverteilung fuer Jungen und Maedchen unterschiedlich ist
freq1 <- repTable(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
         repInd = "jkrep", imp="imp", groups = c("country", "sex"),
         group.differences.by = "sex", cross.differences = FALSE, dependent = "comp",
         chiSquare = TRUE, progress = FALSE)

## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res1  <- report(freq1, add = list( kb = "reading"))
```

Im zweiten Beispiel werden Gruppenvergleiche anhand fünf separater *t*-Tests ausgeführt. Dabei wird für jedes Land und jede einzelne Kompetenzstufe geprüft, ob die empirische Häufigkeit dieser Kompetenzstufe innerhalb jedes Landes zwischen Geschlechtergruppen variiert. Technisch werden dabei für eine fünfstufige Kompetenzstufenvariable fünf dichotome Indikatorvariablen erzeugt, und `repMean` wird für jede dieser fünf abhängigen Variablen aufgerufen.

```
freq2 <- repTable(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
         repInd="jkrep", imp="imp", groups = c("country", "sex"), group.differences.by = "sex",
         cross.differences = FALSE, dependent = "comp", chiSquare = FALSE, progress = FALSE)

## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
```

```
res2  <- report(freq2, add = list( kb = "reading"))
```

Cross-level Differenzen und Trends können mit `repTable` analog wie mit `repMean` bestimmt werden.

3.1 Schleifen über mehrere abhängige Variablen

In Abschnitt 2.5 wurde demonstriert, wie eine `by`-Schleife genutzt werden kann, um mit einem Funktionsaufruf die Analyse für bspw. verschiedene Kompetenzbereiche simultan ausführen zu lassen. Nach demselben Prinzip können auch mehrere Analysen für verschiedene abhängige Variablen berechnet werden. Im Bildungstrend ist das u.a. für das Kapitel über Kompetenzstufenbesetzungen relevant. Hier sollen ja die Häufigkeiten für verschiedene Kriterien (bzw. abhängige Variablen) berichtet werden, nämlich "Mindeststandard verfehlt", "Regelstandard (mindestens) erreicht", "Optimalstandard erreicht". Je nachdem, ob die Kompetenzbereiche als verschiedene Variablen im Datensatz aufgelistet sind, oder in einer einzigen Variable (z.B. `criterion_reached`) auftreten, wobei dann eine zusätzliche Variable definiert, um welches Kriterium es sich handelt, bietet sich dafür eine `lapply`- oder eine `by`-Schleife an:

```
### abhaengige Variablen definieren
AVs   <- c("failMin", "passReg", "passOpt")
freq3 <- lapply(AVs, FUN = function (av) {
         repTable(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
            repInd = "jkrep", imp="imp", groups = c("country", "sex"),
            group.differences.by = "sex", cross.differences = FALSE,
            dependent = av, progress = FALSE) })

## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
```

Die Aufbereitung der Ergebnisse erfolgt (analog wie in 2.5) ebenfalls über eine `lapply`-Schleife:

```
alleResults3     <- do.call("rbind", lapply(freq3, report))
```

3.2 Verschachtelte Schleifen

Beide Schleifen (über nicht-genestete Gruppierungsvariablen und über verschiedene abhängige Variablen) können auch kombiniert werden. In diesem Beispiel wäre für zwei Kompetenzbereiche und 3 abhängige Variablen eine (zweifach) geschachtelte Schleife für insgesamt $3 \times 2 = 6$ Analysen zu definieren:

```
AVs   <- c("failMin", "passReg", "passOpt")
freq4 <- lapply(AVs, FUN = function (av) {
         f4 <- by ( data = bt2010, INDICES = bt2010[,"domain"], FUN = function (sub.dat) {
               repTable(datL = sub.dat, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
                         repInd = "jkrep", imp="imp", groups = c("country", "sex"),
                         group.differences.by = "sex", cross.differences = FALSE,
                         dependent = av, progress = FALSE)})})

## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
```

Die Aufbereitung ist aufgrund der zweifach geschachtelten Struktur etwas aufwändiger:

```
alleResults4     <- lapply(freq4, FUN = function (av) {
                          do.call("rbind", lapply(names(av), FUN = function ( x ) {
                                  report(av[[x]], add = list(kb=x))}))})
alleResults4     <- do.call("rbind", alleResults4)
```

Eine Kombination von zwei Schleifen (über nicht-genestete Gruppierungsvariablen und über verschiedene abhängige Variablen) ist auch bei Trendanalysen möglich. Hier muss jedoch beachtet werden, dass jeder abhängigen Variable gegebenenfalls eine eigene spezifische Variable für den Linkingfehler zugeordnet ist. Um über verschiedene AVs eine Schleife zu definieren, kann daher nicht `lapply` verwendet werden – geeigneter ist hier `apply` oder `mapply`. Am einfachsten ist es, wenn man die jeweilige AV und ihren Linkingfehler in einem data.frame ablegt und die Schleife über die Zeilen des data.frame definiert.

```
### zweifach geschachtelte Schleife mit Trendanalyse
AVs   <- data.frame ( av = c("failMin", "passReg", "passOpt"),
                      le = c("leFailMin", "lePassReg", "lePassOpt"),
                      stringsAsFactors = FALSE)
freq5 <- apply(AVs, MARGIN = 1, FUN = function (abhVars) {
         f4 <- by ( data = bt, INDICES = bt[,"domain"], FUN = function (sub.dat) {
               repTable(datL = sub.dat, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
                        repInd = "jkrep", imp="imp", groups = c("country", "sex"),
                        group.differences.by = "sex", cross.differences = FALSE,
                        trend = "year", dependent = abhVars[["av"]],
                        linkErr = abhVars[["le"]], progress = FALSE)})})

##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
##
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
##
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
##
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
```

```
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
##
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
##
##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 2'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.
```

Für die Aufbereitung wird analog vorgegangen:

```
alleResults5     <- lapply(freq5, FUN = function (av) {
                          do.call("rbind", lapply(names(av), FUN = function ( x ) {
                                  report(av[[x]], add = list(kb=x))}))})
alleResults5     <- do.call("rbind", alleResults5)
```

4. Analyse von Streubreiten (repQuantile)

Die Bestimmung von Quartilen oder Perzentilen erfolgt im Grunde genauso wie die Bestimmung von Mittelwerten, mit zwei wesentlichen Unterschieden: Zum einen können mit `repQuantile` (noch) keine Gruppendifferenzen bestimmt werden, das Argument `group.differences.by` fehlt ersatzlos. Zum zweiten müssen in einem zusätzlichen Argument namens `probs` die Wahrscheinlichkeiten bestimmt werden, an denen die Verteilung abgeschnitten werden soll. Das folgende Beispiel demonstriert dies für den Kompetenzbereich “lesen” beziehungsweise “reading” sowie die Jahre 2010 und 2015, wobei – analog zu den IQB-Bildungstrendstudien – das 5., 10., 25., 75., 90. und 95. Perzentil berechnet werden soll:

```
btRead <- bt[which(bt[,"domain"] == "reading"),]
quan   <- repQuantile(datL = btRead, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
               repInd = "jkrep", imp="imp", groups = "country", trend = "year",
               dependent = "score", linkErr = "leScore",
               probs = c(0, .05, .1, .25, .75, .90, .95, 1), progress = FALSE )

##
## Trend group: '2010'
## 1 analyse(s) overall according to: 'group.splits = 1'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.
##
##
## Trend group: '2015'
## 1 analyse(s) overall according to: 'group.splits = 1'.
## Assume unnested structure with 3 imputations.
## Create 75 replicate weights according to JK2 procedure.

res    <- report(quan, add = list(domain = "reading"))
```

5. Regressionsanalysen (repGlm)

Mit `repGlm` können lineare und logistische Regressionsmodelle spezifiziert werden – also weitgehend alles, was auch `glm` erlaubt. Für Trendanalysen kann (noch) kein Linkingfehler berücksichtigt werden. Die Aufbereitungsfunktion `report` erlaubt bei Bedarf, die Ergebnisse in einer ähnlichen Form darzustellen, wie die `summary`-Funktion für `glm`. Dazu muss in der Reportingfunktion das Argument `printGlm` auf `TRUE` gesetzt werden. Im ersten Beispiel soll für jedes Land separat die Leseleistung (metrisch) auf Geschlecht, sozio-ökonomischen Status und die Interaktion aus beiden regrediert werden:

```
reg1   <- repGlm(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
               repInd = "jkrep", imp="imp", groups = "country",  formula = score~sex*ses,
               family=gaussian(link="identity"), progress = FALSE)

## 1 analyse(s) overall according to: 'group.splits = 1'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res1   <- report(reg1, add = list(domain = "reading"), printGlm = TRUE)

##        Trend group: 'noTrend'.
##             groups: country = LandA
##             domain: reading
## dependent Variable: score
##
##     parameter     est     se t.value p.value
## 1 (Intercept) 438.021 11.907  36.786   0.000
## 2         ses   1.444  0.187   7.709   0.000
## 3     sexmale -29.416 16.544  -1.778   0.076
## 4 sexmale:ses   0.269  0.248   1.085   0.278
##
```

```
##             R-squared: 0.134; SE(R-squared): NA
## Nagelkerkes R-squared: NaN; SE(Nagelkerkes R-squared): NA
## 1203 observations and 1199 degrees of freedom.
## ----------------------------------------------------------------
##             groups: country = LandB
##             domain: reading
## dependent Variable: score
##
##     parameter     est     se t.value p.value
## 1 (Intercept) 367.323 13.060  28.126   0.000
## 2         ses   2.255  0.241   9.358   0.000
## 3     sexmale  -4.027 16.196  -0.249   0.804
## 4 sexmale:ses  -0.167  0.338  -0.495   0.621
##
##             R-squared: 0.222; SE(R-squared): NA
## Nagelkerkes R-squared: NaN; SE(Nagelkerkes R-squared): NA
## 1263 observations and 1259 degrees of freedom.
## ----------------------------------------------------------------
##             groups: country = LandC
##             domain: reading
## dependent Variable: score
##
##     parameter     est     se t.value p.value
## 1 (Intercept) 413.788 13.914  29.738   0.000
## 2         ses   2.081  0.250   8.323   0.000
## 3     sexmale -12.438 16.359  -0.760   0.447
## 4 sexmale:ses  -0.238  0.301  -0.791   0.429
##
##             R-squared: 0.169; SE(R-squared): NA
## Nagelkerkes R-squared: NaN; SE(Nagelkerkes R-squared): NA
## 1243 observations and 1239 degrees of freedom.
```

Im zweiten Beispiel wird ein logistisches Regressionsmodell definiert. Abhängige Variable ist der Indikator, inwiefern der Regelstandard (mindestens) erreicht wurde (Variable `passReg`). Die Variable "country" soll als Regressor verwendet werden:

```
reg2   <- repGlm(datL = bt2010read, ID="idstud", wgt="wgt", type="jk2", PSU="jkzone",
               repInd = "jkrep", imp="imp", formula = passReg~country*ses,
               family=binomial(link="logit"), progress = FALSE)

## 1 analyse(s) overall according to: 'group.splits = 0'.
## Assume unnested structure with 3 imputations.
## Create 62 replicate weights according to JK2 procedure.

res2   <- report(reg2, add = list(domain = "reading"), printGlm = TRUE)

##        Trend group: 'noTrend'.
##             domain: reading
## dependent Variable: passReg
##
##          parameter    est    se t.value p.value
## 1      (Intercept) -0.816 0.239  -3.419   0.001
```

```
## 2     countryLandB -0.996 0.290  -3.440   0.001
## 3 countryLandB:ses  0.008 0.006   1.474   0.141
## 4     countryLandC -0.135 0.354  -0.382   0.703
## 5 countryLandC:ses  0.005 0.008   0.590   0.555
## 6              ses  0.032 0.005   6.280   0.000
##
##             R-squared: 0.134; SE(R-squared): NA
## Nagelkerkes R-squared: 0.125; SE(Nagelkerkes R-squared): NA
## 3709 observations and 3703 degrees of freedom.
```

Standardmäßig werden in allen Regressionsanalysen sowohl das R^2 (für lineare Regressionsmodelle) als auch Nagelkerkes R^2 (logistische Regressionsmodelle) ausgegeben. Abhängig vom gewählten Modell sollte immer nur einer von beiden Werten interpretiert werden.