

# Using the SCDA package (0.8) for analysing single and multiple case AB designs

Jürgen Wilbert, University Potsdam

## Content

Prerequisites.....	2
Introduction and an example .....	2
Descriptives .....	8
Percent non-overlapping data.....	9
Percent exceeding the median.....	9
Percent all non-overlapping data .....	10
Non-overlap of all Pairs .....	11
Percent exceeding the trend .....	12
Reliable Change Index .....	13
Linear trends.....	14
Piecewise linear model.....	15
Randomization tests.....	17
Plot AB-designs.....	18
Smoothing .....	21
Outlier.....	22
Fill in missing measurement times.....	23
Importing and exporting data .....	24
Selecting data .....	25
Random sample generator.....	25
Aggregate multiple cases into one case .....	28
*(Experimental) Power Analysis.....	29
Reference .....	31

This document gives an introduction to the SCDA (single-case data analysis) R-package. The current version can be reached via

<http://tinyurl.com/SCDA-Manual>

The SCDA package was designed for analysing and visualizing single-case AB data sets. These designs are commonly used for evaluating learning progress and curriculum-based measurements.

Furthermore, the SCDA package comprises a module for generating random single and multiple-baseline datasets under a variety of circumstances for conducting Monte-Carlo studies on the power of single-case analysing techniques.

## Prerequisites

Firstly, you have to install the R statistical program. Therefore, go to the R homepage (<http://www.r-project.org>) and choose the appropriate download file for your system.

The SCDA package is currently stored in a single R file and can be accessed via the internet at <http://tinyurl.com/SCDA-for-R>. You can implement and activate it into your R system using the `source` command:

```
source("http://tinyurl.com/SCDA-for-R")
```

You can reach a developmental version under (which is only recommended if you encounter an error):

```
source("http://tinyurl.com/SCDA-develop")
```

## Introduction and an example

The SCDA package offers several methods for coding single-case data. The simplest way is to use a vector of values, each representing one measurement.

In R, a vector is built in the following way:

```
y <- c(5, 7, 8, 5, 7, 12, 16, 18, 15, 14, 19)
```

`y` is the name of the variable. A chain of 11 values is ascribed to it by the arrow operator (`<-`). You can control the ascription by typing in the name of the variable and get the following result:

```
> y
[1] 5 7 8 5 7 12 16 18 15 14 19
```

For a graphical output of `y`, use the `plotSC` command. The parameter `B.start` informs the function about the start of the B-phase.

```
plotSC(y, B.start = 6)
```

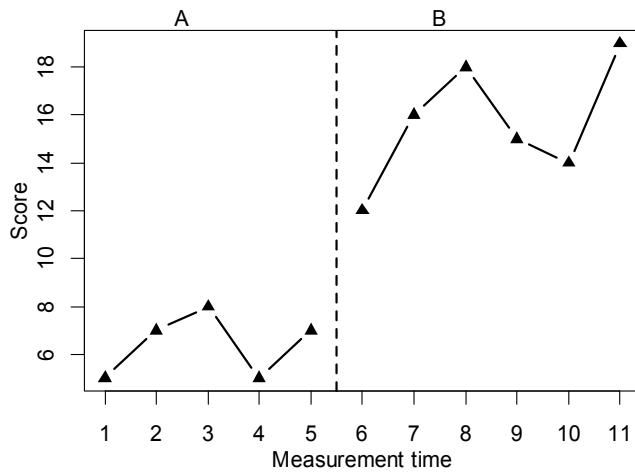


Figure. Simple plot

For a first overview, use the `describeSC` command:

```
> describeSC(y, B.start = 6)
Describe single-case data
```

	Person1	total
n A	5.00	5.00
n B	6.00	6.00
Mean A	6.40	6.40
Mean B	15.67	15.67
Mean dif	9.27	9.27
SD A	1.34	1.34
SD B	2.58	2.58
SD AB	5.24	5.24
SMD A	6.91	6.91
SMD B	3.59	3.59
SMD AB	1.77	1.77
Autocor A	-0.41	-0.41
Autocor B	-0.19	-0.19
Trend A	0.20	0.20
Trend B	0.74	0.74
Trend AB	1.40	1.40
PND	100.00	100.00
PEM	100.00	100.00
NAP	100.00	100.00
PAND	100.00	100.00

While this way of coding the data is handy and applicable to all functions in the SCDA package, the SCDA package internally uses a list of data frames to process single-case data. This allows for a higher flexibility in processing and portability of the data. A data frame is a spreadsheet like object to store data of different variables for a number of cases. The data frame for the above used example is:

phase	values	mt
A	5	1
A	7	2
A	8	3
A	5	4
A	7	5
B	12	6
B	16	7
B	18	8
B	15	9
B	14	10
B	19	11

The first column contains the phase of the measurement, the second the measured values, and the third the measurement times. To create a data frame object you type in:

```
person1 <- makeSCDF(c(5, 7, 8, 5, 7, 12, 16, 18, 15, 14, 19), B.start = 6)
```

If the `MT` variable is left out, it is internally built with values from 1...n (n is the number of measurements). The commands `plotSC(person1)` and `describeSC(person1)` would give the same results as before.

Alternatively, we can type the data into an external spreadsheet program like EXCEL or CALC, save them as a CSV file (comma-separated values), and load them into R:

	A	B	C	D
1	case	phase	values	mt
2	Charlotte	A	5	1
3	Charlotte	A	7	2
4	Charlotte	A	8	3
5	Charlotte	A	5	4
6	Charlotte	A	7	5
7	Charlotte	B	12	6
8	Charlotte	B	16	7
9	Charlotte	B	18	8
10	Charlotte	B	15	9
11	Charlotte	B	14	10
12	Charlotte	B	19	11
13	Theresa	A	3	1
14	Theresa	A	4	2
15	Theresa	A	3	3
16	Theresa	A	5	4
17	Theresa	B	7	5
18	Theresa	B	8	6
19	Theresa	B	7	7
20	Theresa	B	9	8
21	Theresa	B	8	9
22	Theresa	B	10	10
23	Theresa	B	12	11
24	Antonia	A	0	1

Figure. Using EXCEL to create and data frame for one or multiple single cases.

```
person1 <- readSC(file.choose())
```

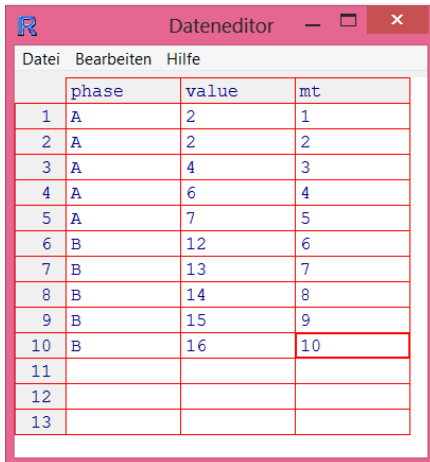
if you set up your computer system in a European country it may be necessary to use

```
person1 <- readSC(file.choose(), sep = ";", dec = ",")
```

instead, as the spreadsheet program writes semicolon separated files and a comma as a decimal sign using the CSV output format. Note that the spreadsheet you want to import must have variable names in the first row and the names of the cases in the first column. The first column is necessary even if you have only one case.

As a third alternative, we can use the built in data editor of R:

```
person1 <- edit(data.frame())
```



	phase	value	mt
1	A	2	1
2	A	2	2
3	A	4	3
4	A	6	4
5	A	7	5
6	B	12	6
7	B	13	7
8	B	14	8
9	B	15	9
10	B	16	10
11			
12			
13			

Figure: R's built in data editor.

Consider the case we have a multiple-baseline dataset of three persons. We may code these in the following way:

```
charlotte <- makeSCDF(c(5, 7, 8, 5, 7, 12, 16, 18, 15, 14, 19), 6)  
theresa <- makeSCDF(c(3, 4, 3, 5, 7, 8, 7, 9, 8, 10, 12), 5)  
antonia <- makeSCDF(c(9, 8, 8, 7, 5, 7, 13, 14, 15, 12, 16), 7)  
study1 <- list(charlotte, theresa, antonia)  
names(study1) <- c("Charlotte", "Theresa", "Antonia")
```

Firstly, we define three data frames. Then we combine the three data frames into a list with the name `study1`. For a nicer look, we added the optional `names` function to give each case an appropriate name that will be used subsequently in different SCDA functions (note that the name of the parameter `B.start` can be left out here because the value is placed exactly at the second position).

Now we can use the `plotSC` command to draw a graphic. To oversee the percent non-overlapping data we added a line representing the maximum of the A-phase using the `lines = "maxA"` parameter:

```
plotSC(study1, lines = "maxA")
```

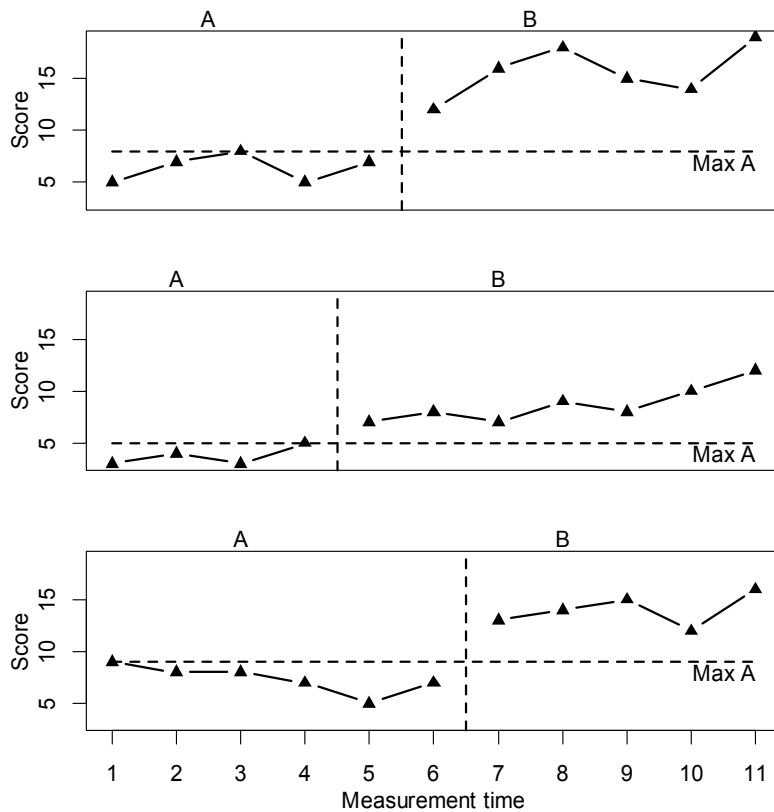


Figure. SCDA standard plot for a multiple baseline design.

We get an overview of the data with the `describeSC` command.

```
> describeSC(study1)
Describe single-case data

          charlotte theresa antonia  total  total(makesingleSC)
n A              5.00   4.00   6.00  15.00                15.00
n B              6.00   7.00   5.00  18.00                18.00
n AB             11.00  11.00  11.00  33.00                33.00
Missing A        0.00   0.00   0.00   0.00                0.00
Missing B        0.00   0.00   0.00   0.00                0.00
Missing AB       0.00   0.00   0.00   0.00                0.00
Mean A           6.40   3.75   7.33   5.83                0.00
Mean B          15.67   8.71  14.00  12.79                6.87
Mean dif         9.27   4.96   6.67   6.97                6.87
Min A            5.00   3.00   5.00   3.00               -2.33
Min B           12.00   7.00  12.00   7.00                3.25
Max A            8.00   5.00   9.00   9.00                1.67
Max B           19.00  12.00  16.00  19.00               12.60
SD A             1.34   0.96   1.37   1.24                1.17
SD B             2.58   1.80   1.58   2.03                2.69
SD AB           5.24   2.91   3.75   4.08                4.06
SMD A            6.91   5.19   4.88   5.66                5.85
SMD B            3.59   2.76   4.22   3.52                2.56
SMD AB           1.77   1.70   1.78   1.75                1.69
Autocor A       -0.41  -0.48   0.31  -0.19               -0.18
Autocor B       -0.19   0.26  -0.60  -0.18               -0.18
Trend A          0.20   0.50  -0.57   0.04               -0.16
Trend B          0.74   0.71   0.40   0.62                0.53
Trend AB         1.40   0.84   0.85   1.03                0.89
Trend dif        0.54   0.21   0.97   0.58                0.69
PND              100.00 100.00 100.00 100.00               100.00
```

PEM	100.00	100.00	100.00	100.00	100.00
NAP	100.00	100.00	100.00	100.00	100.00
PAND	100.00	100.00	100.00	100.00	100.00

For analysing the example, we compute the percent all non-overlapping data (Parker, Hagan-Burke, & Vannest, 2007) by the `pand` command.

```
> pand(study1)
Percent all non-overlapping data

PAND = 100 %
Phi = 1 ; Phi-square = 1

Number of persons: 3
Total measurements: 33
in phase A: 15
in phase B: 18
n Overlapping data per person: 0, 0, 0
n Overlapping data: 0
% Overlapping data: 0

2 x 2 Matrix
      % expected
      A      B      total
%    A  45    0    45
real B  0    55    55
total 45    55
```

Note. Matrix is corrected for ties (Wilbert, 2013)

Then we conduct a multiple-baseline randomization test:

```
> set.seed(1234) #only needed for replicating the following results exactly
> rand.test(study1, limit = 4, complete = TRUE)
```

Randomization test

Multiple-Baseline test for 3 cases.

```
Statistic: Mean B-A
Length A-phase 15
Length B-phase 18
Minimal length of phase: 4
Observed statistic = 6.965873
```

Based on all 64 possible combinations.

```
Distribution:
n = 64
M = 6.215873
SD = 0.428785
Min = 5.280952
Max = 6.965873

p = 0.015625
```

```
Shapiro-Wilk normality test: W = 0.965; p = 0.065 (hypothesis of normality
maintained)
```

```
z = 1.7491, p = 0.0401 (single sided)
```

Moreover, as a parametric approach, we compute a piecewise-linear-regression model (Beretvas & Chung, 2008; Huitema & Mckean, 2000):

```
> plm(study1)
Piecewise Regression Analysis

Multiple Baseline Design for 3 cases.

Regression model: B&L-B

F(3, 29) = 31.20; p = 0.000; R-Square = 0.763; adjusted R-Square = 0.739
```

	B	SE	t	p	R-Square
Intercept	0.483	1.203	0.401	0.691	
Trend	-0.157	0.351	-0.448	0.657	0.002
Level	5.457	1.588	3.437	0.002	0.096
Slope	0.685	0.444	1.543	0.134	0.019

All analyses suggest that the values in the B-phase of the study are higher than in the A-phase.

After this session, we save the data with the standard R procedure `save()` into a file.

```
save(study1, file = "study1.RData")
```

If you have problems finding the right folder on your hard disk, you can choose the `file.choose()` command instead of the file name.

```
save(study1, file = file.choose())
```

Later you can load the data with the `load` command.

```
load("study1.RData") OR load(file.choose())
```

Alternatively, you can export the data into a CSV file with `writeSC` and later read them again with `readSC`.

```
writeSC(study1, "study1.csv")
study1 <- readSC("study1.csv")
```

If you have chosen the right delimiter and decimal separator, you can open the `study1.csv` file into your spreadsheet program by a double-click.

## Descriptives

### Description

The `describeSC` command gives some important descriptives of your single-case data including trends, standardized mean differences, measurements for data overlap, and autocorrelation.

### Example

```
set.seed(1234)
dat <- rSC(5, B.start = c("rand", 5, 10), d.level = 1.0, d.slope = 0.1, trend =
0.05, rtt = 0.75, round = 0, random.names = TRUE)
describeSC(dat)
```

Describe single-case data

	Bradyn	August	Krish	Antoine	Mckayla	total	total(makesingleSC)
n A	5.00	7.00	7.00	7.00	8.00	34.00	34.00
n B	15.00	13.00	13.00	13.00	12.00	66.00	66.00
n AB	20.00	20.00	20.00	20.00	20.00	100.00	100.00
Missing A	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Missing B	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Missing AB	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mean A	56.40	42.71	61.57	68.29	53.50	56.49	0.00



Mean B	76.67	62.00	85.62	86.08	78.42	77.76	21.18
Mean dif	20.27	19.29	24.04	17.79	24.92	21.26	21.18
Min A	49.00	35.00	53.00	63.00	41.00	35.00	-12.50
Min B	61.00	51.00	69.00	72.00	71.00	51.00	3.71
Max A	63.00	52.00	68.00	79.00	69.00	79.00	15.50
Max B	89.00	78.00	98.00	98.00	90.00	98.00	36.50
SD A	5.64	5.71	5.38	5.59	9.37	6.52	6.27
SD B	10.15	8.74	10.44	8.48	7.19	9.08	9.26
SD AB	12.79	12.15	14.71	11.45	14.80	13.25	13.08
SMD A	3.59	3.38	4.47	3.18	2.66	3.46	3.38
SMD B	2.00	2.21	2.30	2.10	3.46	2.41	2.29
SMD AB	1.58	1.59	1.63	1.55	1.68	1.61	1.62
Autocor A	0.33	-0.44	-0.37	0.28	0.58	0.08	0.31
Autocor B	0.56	0.63	0.62	0.75	0.33	0.58	0.51
Trend A	3.30	1.29	1.00	1.11	2.88	1.91	1.74
Trend B	1.96	1.88	2.25	1.98	1.59	1.93	1.88
Trend AB	2.02	1.89	2.30	1.81	2.32	2.07	1.89
Trend dif	-1.34	0.59	1.25	0.88	-1.29	0.02	0.14
PND	86.67	84.62	100.00	76.92	100.00	89.64	69.70
PEM	100.00	100.00	100.00	100.00	100.00	100.00	100.00
NAP	97.33	97.80	100.00	96.70	100.00	98.37	97.42
PAND	90.00	85.00	100.00	90.00	100.00	93.00	92.00

## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`decreasing`: If you expect the data to be lower in the B phase, set `decreasing = TRUE` (default is `FALSE`).

## Percent non-overlapping data

### Description

The function `pnd` return the percentage non-overlapping data. If you expect a decrease in the B-phase, set the parameter `decreasing = TRUE`. Due to its error-proneness you should not use `PND` but `NAP` or `PAND` instead (see Parker & Vannest, 2009).

## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`decreasing`: If you expect the data to be lower in the B phase, set `decreasing = TRUE` (default is `FALSE`).

## Percent exceeding the median

### Description

The function `pem` return the percentage of the B-phase data exceeding the median of the A-phase data. Additionally, a  $X^2$  test against a 50/50 distribution is computed. Different measures of central tendency could be addressed for alternative analyses. If you expect a decrease in the B-phase, set the parameter `decreasing = TRUE`.

## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

decreasing: If you expect the data to be lower in the B phase, set `decreasing = TRUE` (default is `FALSE`).

`binom.test`: Computes a binomial test for a 50/50 distribution. By default set `TRUE`.

`chi.test`: By default set `FALSE`. If set `FALSE` the Chi<sup>2</sup> Test is skipped.

`FUN`: By default `median`. Alternatively, you could implement any other function for computing an average.

...: Additional arguments for the `FUN` parameter (e.g. `FUN = mean, trim = 0.1` will use a 10% trimmed mean instead of a median for analysing the data).

## Example

```
set.seed(1234)
dat <- rSC(5, d.level = 0.5)
pem(dat)
```

Percent Exceeding the Median

	PEM	binom.p	Chi	DF	p
Case 1	60.000	0.304	0.600	1	0.439
Case 2	86.667	0.004	8.067	1	0.005
Case 3	100.000	0.000	15.000	1	0.000
Case 4	20.000	0.996	5.400	1	0.020
Case 5	100.000	0.000	15.000	1	0.000

Alternative hypothesis: true probability > 50%

## Percent all non-overlapping data

### Description

The percent non-overlapping data is a measurement for estimating a level increase (or decrease) in performance after an intervention. It can be computed with the `pand` command. PAND allows for a Chi<sup>2</sup> statistical test (comparing real and estimated association a data points with phase A and B) and thereby allows an estimation of the effect size phi, which is identical to Pearsons *r* computed with dichotomous data. Phi<sup>2</sup> is thus the amount of explained variance. The original procedure for computing PAND as proposed by Parker and colleagues (Parker u. a., 2007) does not take ambivalent data points due to ties into account, as does the computation for the nonoverlap of all pairs index (Parker & Vannest, 2009). In the `pand` procedure, the frequency matrix is corrected for ties.

### Example

A simple example:

```
> set.seed(1234)
> dat <- rSC(1, d.level = 1.0)
> pand(dat)
```

Percent all non-overlapping data

```
PAND = 90 %
Phi = 0.733 ; Phi-square = 0.538
```

```
Number of persons: 1
Total measurements: 20
in phase A: 5
in phase B: 15
```

```
n Overlapping data per person: 2
n Overlapping data: 2
% Overlapping data: 10
```

```
2 x 2 Matrix
      % expected
      A      B      total
%    A  20     5       25
real B  5     70      75
total 25     75
```

Note. Matrix is corrected for ties (Wilbert, 2013)

Computing PAND when expecting a decrease in performance after intervention in a multiple-baseline design:

```
> set.seed(1234)
> dat <- rSC(3, B.start = c(5, 7, 9), d.level = -1.0)
> pand(dat, decreasing = TRUE)
```

Percent all non-overlapping data

```
PAND = 80 %
Phi = 0.524 ; Phi-square = 0.274
```

```
Number of persons: 3
Total measurements: 60
in phase A: 18
in phase B: 42
n Overlapping data per person: 2, 4, 6
n Overlapping data: 12
% Overlapping data: 20
```

```
2 x 2 Matrix
      % expected
      A      B      total
%    A  20     10      30
real B 10     60      70
total 30     70
```

Note. Matrix is corrected for ties (Wilbert, 2013)

## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`decreasing`: If you expect the data to be lower in the B phase, set `decreasing = TRUE` (default is `FALSE`).

`correction`: If set `TRUE`, a correction for ties conducted (default is `TRUE`).

## Non-overlap of all Pairs

### Description

The index Nonoverlapping of all pairs was developed by Parker and colleagues (Parker & Vannest, 2009). It is recommended as an overlapping measurement.

## Example

```
> set.seed(1234)
> dat <- rSC(1, d.level = -1.0)
> nap(dat, decreasing = TRUE)
```

Non-overlap of All Pairs

N persons = 1

NAP = 92 %

Rescaled NAP = 84 %

## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`decreasing`: If you expect the data to be lower in the B phase, set `decreasing = TRUE` (default is `FALSE`).

## Percent exceeding the trend

### Description

The function `pet` return the percentage of the B-phase data exceeding the predicted values on basis of the trend of the A-phase. A binomial test against a 50/50 distribution is computed. Furthermore, the percent of data points in the B-phase exceeding the upper (or lower) threshold of the 95% confidence interval of the predicted values is returned. If you expect a decrease in the B-phase, set the parameter `decreasing = TRUE`.

## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`decreasing`: If you expect the data to be lower in the B phase, set `decreasing = TRUE` (default is `FALSE`).

`ci`: The size of the confidence interval (by default `ci = .95`).

## Example

```
set.seed(1234)
dat <- rSC(5, d.slope = 0.2)
pet(dat)
```

Percent Exceeding the Trend

N persons = 5

	PET	binom.p	PET	CI
Case 1	93.333	0.000	0	
Case 2	93.333	0.000	0	
Case 3	100.000	0.000	100	
Case 4	46.667	0.696	0	
Case 5	100.000	0.000	100	

Binom.test: alternative hypothesis: true probability > 50%

PET CI: Percent of values greater than upper 95% confidence threshold (greater 1.645\*se above predicted value)

## Reliable Change Index

### Description

Computes different measures for a reliable change index (see Wise, 2004).

### Example

```
rci(byHeartSC[1])
```

Reliable Change Index

```
N persons = 1
Mean difference = 12.4
Standardized difference = 1.69056
```

Descriptives:

	n	mean	SD	SE
A-Phase	5	0.4	0.5477226	0.244949
B-Phase	15	12.8	5.6340800	2.519637

```
Reliability = 0.8
```

95 % confidenzintervals:

	Lower	Upper
A-Phase	-0.08009117	0.8800912
B-Phase	7.86160191	17.7383981

Reliable Change Indices:

	RCI
Jacobson et al.	50.62279
Christensen and Mendoza	35.79572
Hageman and Arrindell	17.92417

RCI for multiple single-cases with a graph:

```
rci(byHeartSC, graph = TRUE)
```

Reliable Change Index

```
N persons = 6
Data of cases are aggregated using the makesingleSC command.
```

```
Mean difference = 7.715789
Standardized difference = 1.284888
```

Descriptives:

	n	mean	SD	SE
A-Phase	30	7.397813e-18	0.5381962	0.2406887
B-Phase	95	7.715789e+00	5.7480120	2.5705891

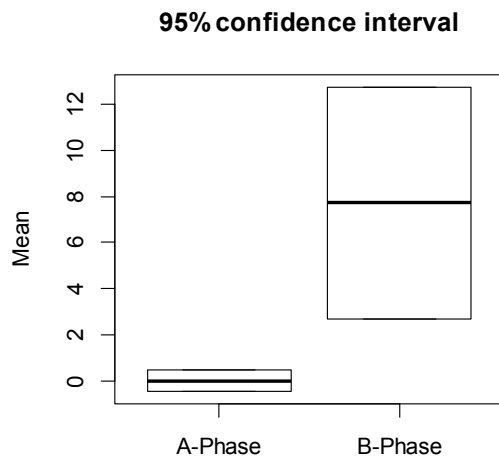
```
Reliability = 0.8
```

95 % confidenzintervals:

	Lower	Upper
A-Phase	-0.4717411	0.4717411
B-Phase	2.6775274	12.7540515

Reliable Change Indices:

	RCI
Jacobson et al.	32.05714



## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`rel`: Reliability necessary for computing standard errors. Default is `rel = 0.8`.

`ci`: Span of the confidence interval (default is `ci = 0.95` indicating a 95% confidence interval).

`graph`: If set `TRUE`, a boxplot with the mean of the A and B-phase and the confidence intervals is plotted (default is `FALSE`).

## Linear trends

### Description

If you want to get an overview of linear trends in the different phases of a single-case you can use the `trendSC` command. By default, it gives you the intercept and slope of a linear and a squared regression model of the measurement-time on the values. Models are computed for the A phase, the B-phase (with correcting the measurement-time to start at one), and the combined A and B-Phase. For a more advanced application, you can add further regression models using the R specific formula class, which will be applied again to the A-phase, the B-phase, and the combined A and B-phase.

## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`B.offset`: An offset for the first measurement-time (MT) of the B-phase. If set 0, the first MT of the B-phase is set to 1. The default is -1, so the MT of the B-phase begin with 0.

`model`: A string or a chain of (named) strings each depicting one regression model. This is a formula expression of the standard R class. The parameters of the model are `values`, `mt` and `phase`.

## Example

Simple example:

```
set.seed(1000)
dat <- rSC(1, d.slope = 0.5)
Trend in A and B-Phase

N persons = 1

      Intercept      B  Beta
Linear.AB    26.702  4.544 0.970
Linear.A     42.625  0.407 0.179
Linear.B     47.368  5.319 0.976
Squared.AB   43.811  0.213 0.984
Squared.A    43.674  0.016 0.042
Squared.B    60.055  0.363 0.967
```

Note. Measurement-times of the B-Phase start at 0

Advanced example:

In addition to the linear and squared regression model, two further models are computed: a) a cubic model, and b) the values predicted by the natural logarithm of the measurement time. Additionally, the data of eight single-cases are aggregated.

```
set.seed(1000)
dat <- rSC(8, d.slope = 0.3)
trendSC(dat, B.offset = 0, model = c("Cubic" = "values ~ I(mt^3)", "Log Time" =
"values ~ log(mt)"))

Trend in A and B-Phase

N persons = 8
Data of cases are aggregated using the makesingleSC command.
```

	Intercept	B	Beta
Linear.AB	-10.139	2.723	0.945
Linear.A	-1.625	0.542	0.162
Linear.B	-0.616	3.153	0.941
Squared.AB	0.253	0.127	0.952
Squared.A	-0.970	0.088	0.161
Squared.B	9.053	0.188	0.924
Cubic.AB	4.613	0.006	0.922
Cubic.A	-0.718	0.016	0.154
Cubic.B	12.948	0.012	0.884
Log Time.AB	-18.201	17.317	0.825
Log Time.A	-1.211	1.265	0.152
Log Time.B	-6.372	16.656	0.870

Note. Measurement-times of the B-Phase start at 1

## Piecewise linear model

### Description

The `plm` function computes a piecewise regression model (see Huitema & Mckean, 2000). A piecewise regression models a linear trend in the data, a level and a slope increase with the start of the intervention. Additionally, the `plm` function computes two effect-sizes by comparing restricted to full regression models (see Beretvas & Chung, 2008). Furthermore, an autoregression model can be

used to take into account autocorrelated data (Beretvas & Chung, 2008). For the latter, your R system must comprise an installation of the `nlme` package.

If your dataset contains multiple single-cases, `plm` internally standardizes all cases and puts the data into on new single-case (see the `makesingleSC` command). This new case is then used to compute a piecewise regression model.

## Example

Simple example:

```
set.seed(1000)
dat <- rSC(1, MT = 30, B.start = 11, d.level = 1.0, d.slope = 0.05, trend = 0.05)
plm(dat)
```

Piecewise Regression Analysis

F(3, 26) = 66.41; p = 0.000; R-Square = 0.885; adjusted R-Square = 0.871

	B	SE	t	p	R-Square
Intercept	44.253	3.401	13.013	0.000	
Trend	0.295	0.548	0.539	0.594	
Level	11.779	4.021	2.930	0.007	0.038
Slope	0.877	0.581	1.509	0.143	0.010

Separate regressions for phases

	Intercept	Slope
Phase A	44.25340	0.2954611
Phase B	59.28295	1.1724064

More complex example:

```
set.seed(2000)
dat <- rSC(3, MT = 30, B.start = 11, d.level = 1.0, d.slope = 0.05, trend = 0.05)
plm(dat, AR = 3)
Piecewise Regression Analysis
```

Multiple Baseline Design for 3 cases.

Correlated residuals up to autorregressions of lag 3 are modeled

F(3, 86) = 164.00; p = 0.000; R-Square = 0.851; adjusted R-Square = 0.846

	B	SE	t	p	R-Square
Intercept	-3.009	1.608	-1.871	0.065	
Trend	0.562	0.258	2.174	0.032	0.005
Level	11.304	1.889	5.984	0.000	0.040
Slope	0.407	0.275	1.482	0.142	0.002

Separate regressions for phases

	Intercept	Slope
Phase A	-3.009112	0.5617988
Phase B	25.710711	0.9692528

## Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the "Introduction and example" chapter for a description.

`model`: Regression model used for computation (see Huitema & Mckean, 2000). Default is `model = "B&L-B"`. Possible values are: `B&L-B`, `H-M`, `Mohr#1`, `Mohr#2`, `Manly`.



AR: Autoregression modelled up to the given lag. This function is only applicable when the nlme package is installed. By default, no correlated residuals are assumed (the nlme package can be easily installed with `install.packages("nlme")`).

## Randomization tests

### Description

The function `rand.test` computes a randomization test for single or multiple baseline single-case datasets. The function is based on an algorithm of the R-package SCRT (Bulté & Onghena, 2009, 2012) but rewritten and extended for the use in AB designs.

### Example

### Parameters and arguments

`data`, `B.start`: Specification of the data. Read the “Introduction and example” chapter for a description.

`stat`: Statistic that is computed for randomization. Default is `stat = "Mean B-A"`.

Table.

Arguments for the `stat` parameter of the `rand.test` function.

Argument	Result
Mean A-B	Computes the difference between the mean of A and the mean of B. This is used if a level decrease in the B-phase is expected.
Mean B-A	Computes the difference between the mean of B and the mean of A. This is used if a level increase in the B-phase is expected.
Mean  A-B	Computes the absolute value between the difference of the mean of the A and B-Phase.
Median A-B	The same as Mean A-B but based on the median.
Median B-A	The same as Mean B-A but based on the median.
B	Computes the difference between the slope parameter of a linear regression model for the B and A-phase. The procedure has not yet been tested for practicality, so it is experimental.
Bdecrease	Complementary to the BETA argument. Computes the difference between the slopes of the A and the B-phase. This is useful when a decrease in the slope due to an intervention is expected.
B plm level	Computes piecewise regression models and takes the B factors of the level predictor as the critical statistic.
B plm slope	Computes piecewise regression models and takes the B factors of the slope predictor as the critical statistic.

`number`: Size of the randomization distribution. The exactness of the p-value could not exceed  $1/\text{number}$  (i.e., `number = 100` results in p-values with an exactness of one percent). Default is `number = 500`. For a faster processing set `number = 100`. For a more precise p-value set `number = 1000`.

`complete`: If `TRUE`, the distribution is based on a complete permutation of all possible starting combinations. This setting overwrites the `number` Argument. The default setting is `FALSE`.

`limit`: Minimal length of the A- und B-phase in the randomization sample. Default is `limit = 5`.

`exclude.equal`: If set `FALSE`, random distribution value that are equal to the observed distribution are counted as values of the distribution of the null-hypothesis. That is, they decrease the probability of rejecting the null-hypothesis (increase the p-value). Default is `exclude.equal = FALSE`. This should be set to `TRUE` if you analyse one single-case design (not a multiple baseline data set) to get a sufficient test-power but be aware it increases the chance of an alpha-error.

`graph`: If `TRUE`, a histogram of the resulting distribution is plotted. Default is `HIST = FALSE`.

`output` and `data.check`: Functions for speed optimization in a Monte-Carlo analysis. Defaults are `output = "c"` and `data.check = TRUE`.

## Plot AB-designs

### Description

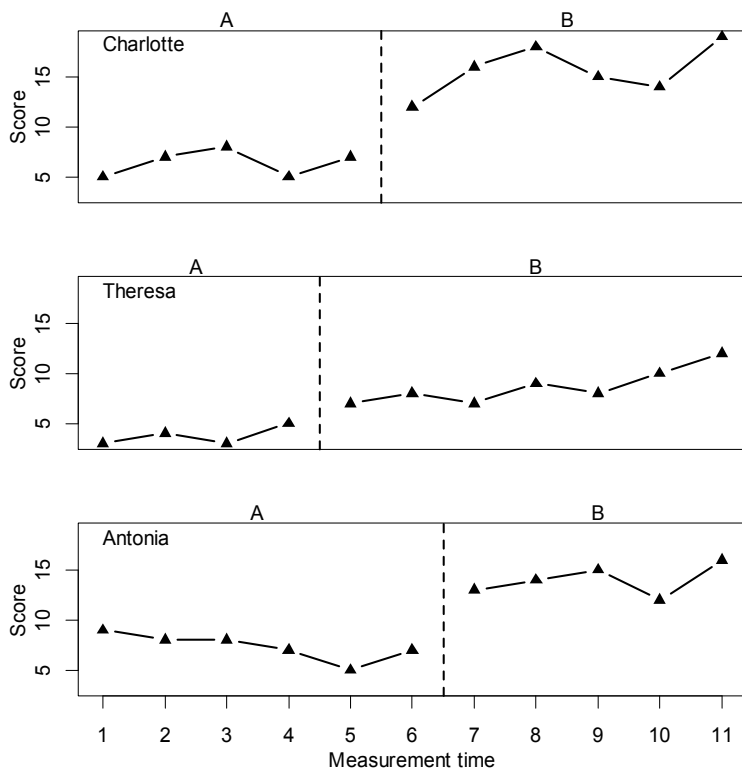
The `plotSC` command draws a nice plot of one or multiple single-cases. By default, `plotSC` draws a multiple single-case plot if the given data contains more than one case.

You can export the plot in several graphic formats including `jpeg`, `tiff`, `pdf`, and as a vector graphic either using the clipboard or writing it into a file. Windows users can easily use the file-menu of the plot window that comes up when drawing a plot to export the graphic.

### Example

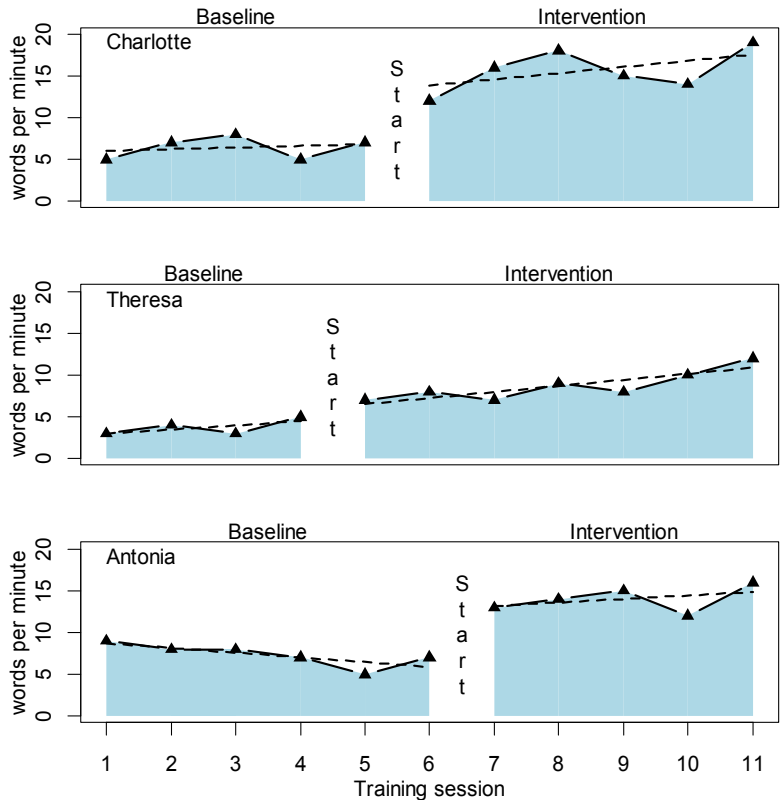
The default settings of the `plotSC` command using the data from the introduction chapter:

```
plotSC(study1)
```



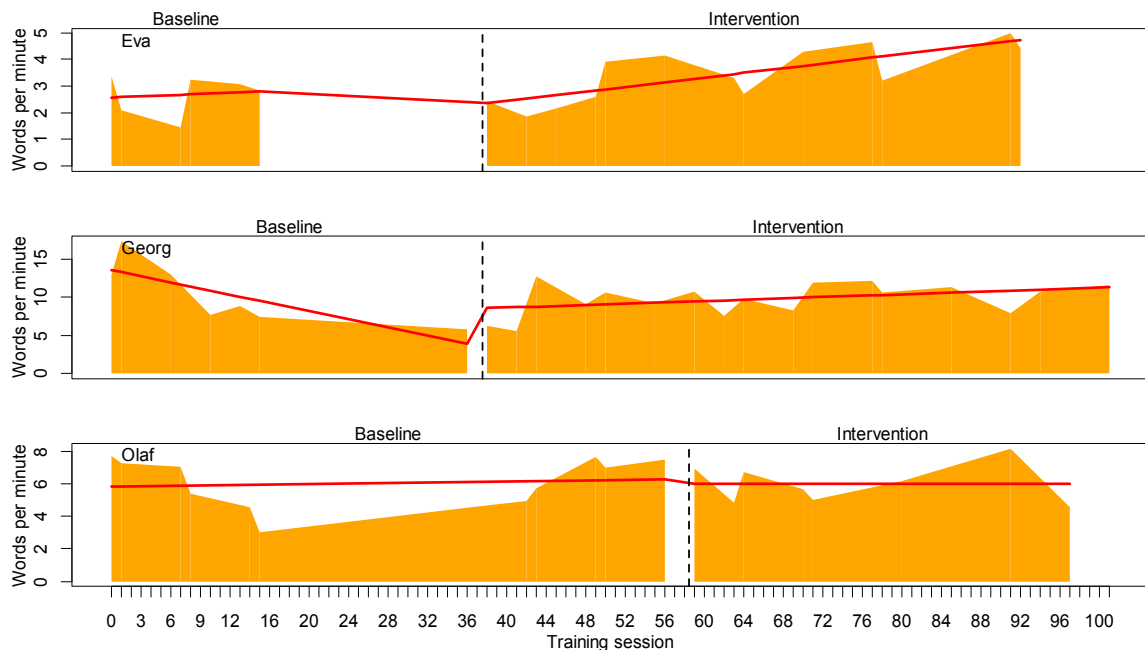
An example of a customized setup with the same data:

```
plotSC(study1, ylim = c(0,20), fill = "light blue", xlab = "Training session", ylab = "words per minute", text.ABlag = "Start", phase.names = c("Baseline", "Intervention"), lines = "trend")
```



A third example with the data of the study from Grosche (2011):

```
plotSC(Grosche2011SC, fill = "orangel", type = "n", ylim = c(0,NA), xlab = "Training session", ylab = "Words per minute", phase.names = c("Baseline", "Intervention"), lines = c("plm", lty = "solid", col = "red", lwd = 3))
```



## Parameters and arguments

`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`phase.names`: SCDA uses the default names “A” and “B” for the two experimental phases. You can change these with the `phase.names` parameter. Example: `phase.names = c("Baseline", "Treatment")`.

`case.names`: SCDA tries to extract the case names from the given data. If that is not possible, no case names are included in the plot. If you want to set the case names explicitly, you can use the `case.names` parameter. Example: `case.names = c("Antonia", "Charlotte", "Theresa")`.

`lines`: Adds one or more lines or curves to the plot for a better estimation of statistical parameters. The argument is either given as a single character string `lines = "median"` or a chain of character strings `lines = c("median", "trend")`. Additionally, some of the procedures can be refined by a single argument e.g., `lines = c("mean" = 0.10)` adds 10% trimmed mean lines

Table: Arguments for the parameter `lines`.

Argument	Result
<code>median</code>	A median line for the A and B-phase.
<code>mean</code>	A mean line for the A and B-phase. Default is a 10% trimming. Other trimming values can be set by a second parameter e.g., <code>lines = c(mean = 0.2)</code> draws a 20% trimmed mean line.
<code>trend</code>	A trend line for the A and B-phase.
<code>trendA</code>	A trend line for the A-phase.
<code>maxA</code> OR <code>pnd</code>	A line for the maximum of the A-phase.
<code>medianA</code>	A line for the median of the A-phase.
<code>meanA</code>	A line for the 10% trimmed mean of the A-phase. Trimming can be changed in the following way <code>lines = c(meanA = 0.15)</code> .
<code>plm</code> OR <code>piecisereg</code>	A regression line for a piecewise linear regression model.
<code>plm.ar</code>	A regression line for a piecewise autoregression model. The lag is specified by <code>plm.ar = 2</code> . Where the number is the maximum order of the autoregression analysis.
<code>movingMean</code>	Draws a moving mean curve. The lag can be specified by <code>lines = c(movingMean = 2)</code> . Default is a lag 1 curve.
<code>movingMedian</code>	Draws a moving median curve. The lag can be specified by <code>lines = c(movingMedian = 3)</code> . Default is a lag 1 curve.
<code>loreg</code>	A non-parametric local regression line. The proportion of data influencing each data point can be specified by <code>lines = c(loreg = 0.66)</code> . The default is 0.5.
<code>lty</code> , <code>lwd</code> , <code>col</code>	Specifies line type <code>lty</code> , line width <code>lwd</code> , and line colour <code>col</code> . Default is a black dashed line with the width 2. E.g., to draw a solid red thick mean line type <code>lines = c(mean, lty = "solid", lwd = 4, col = "red")</code> . Some of the

	possible line types are "solid", "dashed", and "dotted".
--	--

`fill.col`: If set, the area beneath the plot line is coloured in the given colour (e.g., `fill.col = "grey"`). Use the standard R command `colours()` to get a list of possible colours.

`textAB.lag`: By default, SCDA draws a vertical line to separate the A and B-phase. Alternatively, you could print a character string between the two phases using the `textAB.lag` parameter (e.g., `textAB.lag = "Start"`).

Some standard parameters of the R plot function that are useful here:

`xlim` and `ylim`: Lower and upper limit for the x- and y-axis of the plot (e.g., `ylim = c(0, 20)` sets the y-axis to a scale from 0 to 20). In multiple single cases plots you can use `ylim = c(0, NA)` to scales the y-axis from 0 to the maximum of each case. If the `xlim` and `ylim` parameter is not set, SCDA automatically computers a proper scale.

`xlab` and `ylab`: Set the labels for the x and y-axis. The defaults are "Measurement time" and "Score" (e.g., `xlab = "Sessions"`).

`type`: Sets the plot type. "l" draws lines, "p" points, "b" draws lines and points, and "n" draws nothing onto the plot (default of `plotSC` is "b"). The "n" argument can be useful in combination with the `lines` argument (e.g., `type = "n", lines = "loreg"`).

`lwd`: Width of the plot lines. Default is 2.

`pch`: Point type. Default is 17. (e.g., `pch = 16` draws filled circles; `pch = "A"` draws As).



Figure: Some possible values for the `pch` parameter of the `plotSC` function.

## Smoothing

### Description

This command provides three different procedures to smooth the data (i.e., to eliminate noise).

A moving average function (either mean based or median based) replaces step-by-step each data point by the average of the surrounding data. With a local regression function, each data point is regressed by its surrounding values.

### Example

### Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the "Introduction and example" chapter for a description.

`FUN`: Function to compute the smoothing. Default is a lag 1 moving Median function.

Table: Arguments for the parameter `FUN`.

Argument	Result
<code>movingMean</code>	Computes the moving mean.
<code>movingMedian</code>	Computes the moving median.
<code>localRegression</code>	Computes non-parametric local regressions.

`intensity`: In the case of `movingMean` or `movingMedian` it is the lag (i.e., the range of values around the target value) that is used for computing the average (default is 1). In case of `localRegression` it is the proportion of data influencing each data point (default is 0.2).

## Outlier

### Description

Identifies and drops outliers for each case of your data file. The function returns a new data file without the outlier data. Criteria for outlier identification can be standard deviations, confidence intervals, and Cook's distance (based on a piecewise linear regression model).

### Example

To "clean" the original data from Grosche (2011) without the outliers and using Cook's distance greater  $4/n$  as a criteria, type:

```
new.data <- outlierSC(Grosche2011SC, criteria = c("Cook", "4/n"))
```

```
Outlier analysis for single case data
```

```
Criteria: Cook's distance based on piecewise regression exceeds 4/n
```

```
Eva : Dropped 1 data
Georg : Dropped 3 data
Olaf : Dropped 2 data
```

The object `new.data` contains the "cleaned" data and can be used now for further analyses (e.g., `describeSC(new.data)`).

### Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the "Introduction and example" chapter for a description.

`criteria`: Specifies criteria for dropping data. Default is `criteria = c("SD", 2)` i.e, drop all values exceeding two standard deviations.

Table: Arguments for the parameter `criteria`.

Argument	Result
<code>SD</code>	Standard deviations.
<code>CI</code>	Confidence interval. 0.95 represents a 95% confidence interval.

Cook	Cook's distance based on the piecewise-linear-regression model. To exclude cases where cook exceeds 4/n set: <code>criteria = c("Cook", "4/n").</code>
------	---

## Fill in missing measurement times

### Description

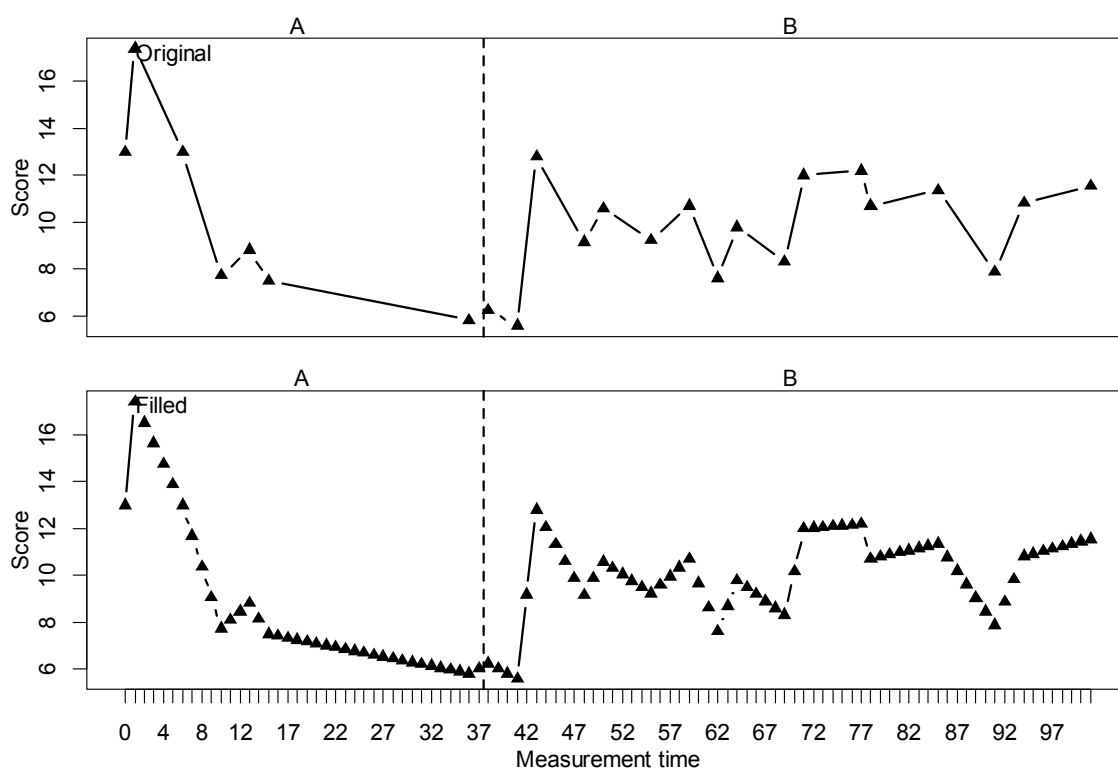
The command `fillmissingSC()` fills in missing measurements if a single-case or multiple single-cases by interpolation.

### Example

In the study of Grosche (2011) measurements couldn't be realized each single week for all participants. During the course of 100 weeks, about 20 measurements per person at different times were administered. Here is a way to fill in the missing data:

```
new.data <- fillmissingSC(Grosche2011SC)
```

```
plotSC(list(Original = Grosche2011SC[[2]], Filled = new.data[[2]]))
```



### Parameters and Arguments

`data`, `B.start` and `MT`: Specification of the data. Read the "Introduction and example" chapter for a description.

## Importing and exporting data

### Description

If you want to export or import your data for the use with other programs (like EXCEL or CALC) you can use the `readSC` and `writeSC` commands.

### Example

```
# create an example dataset with three cases
dat <- rSC(3, MT = 10, B.start = c(5,6,7), d.level = 1.0)
# name the cases
names(dat) <- c("Klara", "James", "Samantha")
# write the cases into a csv file.
writeSC(dat, "example.csv")
```

The resulting data file can be imported into a spreadsheet program and looks something like this:

	A	B	C	D
1	case	phase	values	mt
2	Klara	A	55.448105420411	1
3	Klara	A	58.1268075064462	2
4	Klara	A	61.3359031271745	3
5	Klara	A	55.9736623272424	4
6	Klara	B	69.8555789675726	5
7	Klara	B	77.2385402707659	6
8	Klara	B	61.3270393374128	7
9	Klara	B	62.3479421686947	8
10	Klara	B	70.1356485086755	9
11	Klara	B	70.5010707844293	10
12	James	A	41.2601835980338	1
13	James	A	58.5888556224445	2
14	James	A	50.6158290949696	3
15	James	A	54.411736249002	4
16	James	A	61.410419902144	5
17	James	B	61.4321016357799	6
18	James	B	71.8943231161831	7
19	James	B	66.9898510890472	8
20	James	B	69.0927435527971	9
21	James	B	71.4369743810074	10
22	Samantha	A	64.2871259707484	1
23	Samantha	A	52.6919464123854	2
24	Samantha	A	57.7375137080534	3
25	Samantha	A	58.4525980264936	4
26	Samantha	A	65.0316065362981	5
27	Samantha	A	66.3446879669641	6
28	Samantha	B	66.8442390844992	7
29	Samantha	B	60.1710852461412	8
30	Samantha	B	77.4565451577345	9
31	Samantha	B	72.0004809085326	10
32				

Vice versa, a data file of the given structure can be exported as a CSV file and then be imported with

```
dat <- readSC("example.csv")
```

```
describeSC(dat)
```

```
Describe single-case data
```

```
      James   Klara Samantha total
n A      5.00   4.000   6.00 15.00
n B      5.00   6.000   4.00 15.00
Mean A   53.26  57.721  60.76 57.25
Mean B   68.17  68.568  69.12 68.62
Mean dif 14.91  10.847   8.36 11.37
SD A     7.86   2.674   5.32  5.69
SD B     4.25   5.902   7.37  5.98
SD AB    9.86   7.287   7.24  8.22
```



SMD A	1.90	4.056	1.57	2.51
SMD B	3.51	1.838	1.13	2.16
SMD AB	1.51	1.488	1.15	1.38
Autocor A	-0.29	-0.269	0.12	-0.15
Autocor B	-0.38	-0.076	-0.19	-0.21
Trend A	3.61	0.479	1.37	1.82
Trend B	1.72	-0.487	3.28	1.50
Trend AB	2.91	1.503	1.71	2.04
PND	100.00	83.333	75.00	86.11
PEM	100.00	100.000	75.00	91.67
NAP	100.00	95.833	87.50	94.44
PAND	100.00	80.000	80.00	86.67

Note that most windows systems in Germany are set up with a ";" as a separator and a "." as a decimal sign. If you have a system like this, you can use the `sep = ";"`, `dec = "."` parameters in the `writeSC` and `readSC` command to specify this condition.

## Parameters and Arguments

`dat`: Data file to be exported (only for the `writeSC` command)

`filename`: Name of the file to be exported or imported

`sort.labels`: If set `TRUE` resulting list is sorted by label names (alphabetically, increasing). Default is `FALSE`.

`sep`: Variable separator (default is ",")

`dec`: Decimal sign (default is ".")

## Selecting data

### Description

If you want to select a certain subset of the data, you can use the `selectSCData` command.

### Example

The example dataset `byHeartSC` contains single-case data of students learning vocabulary by heart in 20 sessions with additional five pre-training measurements.

If you want to plot the first to the 20<sup>th</sup> measurement type:

```
newData <- selectSCData(byHeartSC, select = 1:20)
plotSC(newData)
```

## Parameters and Arguments

## Random sample generator

### Description

SCDA offers a module for generating random single- and multiple-baseline samples.

### Example

```
set.seed(1234)
study <- rSC(3, d.level = 0.7, d.slope = 0.1, concise = TRUE)
plm(study, AR = 2)
```

Piecewise Regression Analysis

Multiple Baseline Design for 3 cases.

Correlated residuals up to autorregressions of lag 2 are modeled

F(3, 56) = 63.43; p = 0.000; R-Square = 0.773

	B	SE	t	p	R-Square
Intercept	0.117	0.328	0.357	0.723	
Trend	-0.040	0.098	-0.411	0.683	
Level	0.732	0.360	2.036	0.046	0.018
Slope	0.178	0.100	1.776	0.081	0.015

As a more advanced example, that requires some knowledge in R-programing see the following Monte-Carlo study on the influence of trends on the power of different analysing techniques.

```
op <- par(lwd = 3, mfrow = c(2,1))
for(d in c(0,1)){
  it <- seq(0, 0.2, length = 6)
  study <- list()
  for(i in 1:length(it))
    study[[i]] <- rSC(n = 150, MT = 20, B.start = 10, d.level = d, trend = it[i],
rtt = 0.8, concise = TRUE)

  p.rand <- lapply(study, function(x) unlist(lapply(x, function(x)
rand.test(list(x), number = 100, output = "p", exclude.equal = TRUE,data.check =
FALSE))))
  p.t <- lapply(study, function(z) unlist(lapply(z, function(y) t.test(values~phase,
data = y, alternative = "less", var.equal = TRUE)$p.value)))
  p.pand <- lapply(study, function(z) unlist(lapply(z, function(y) pand(y)$PAND)))
  p.nap <- lapply(study, function(z) unlist(lapply(z, function(y) nap(y, data.check
= FALSE)[[1]])))
  p.plm <- lapply(study, function(z) unlist(lapply(z, plm.mt)))

  y <- unlist(lapply(p.rand,function(x) mean(x<=0.05)))
  plot(it, y, xlab = "trend in d/mt", ylab = "% significant", type = "l", ylim =
c(0,1))
  y <- unlist(lapply(p.t,function(x) mean(x<=0.05)))
  lines(it,y, lty = 2)
  y <- unlist(lapply(p.pand,function(x) mean(x>80)))
  lines(it,y, lty = 3)
  y <- unlist(lapply(p.nap,function(x) mean(x>80)))
  lines(it,y, lty = 4)
  y <- unlist(lapply(p.plm,function(x) mean(x<=0.05)))
  lines(it,y, lty = 5)

  legend("bottomright", legend = c("Rand Test", "T-Test", "PAND>80%", "NAP>80%",
"PLM"), lty = 1:5)
}

par(op)
```

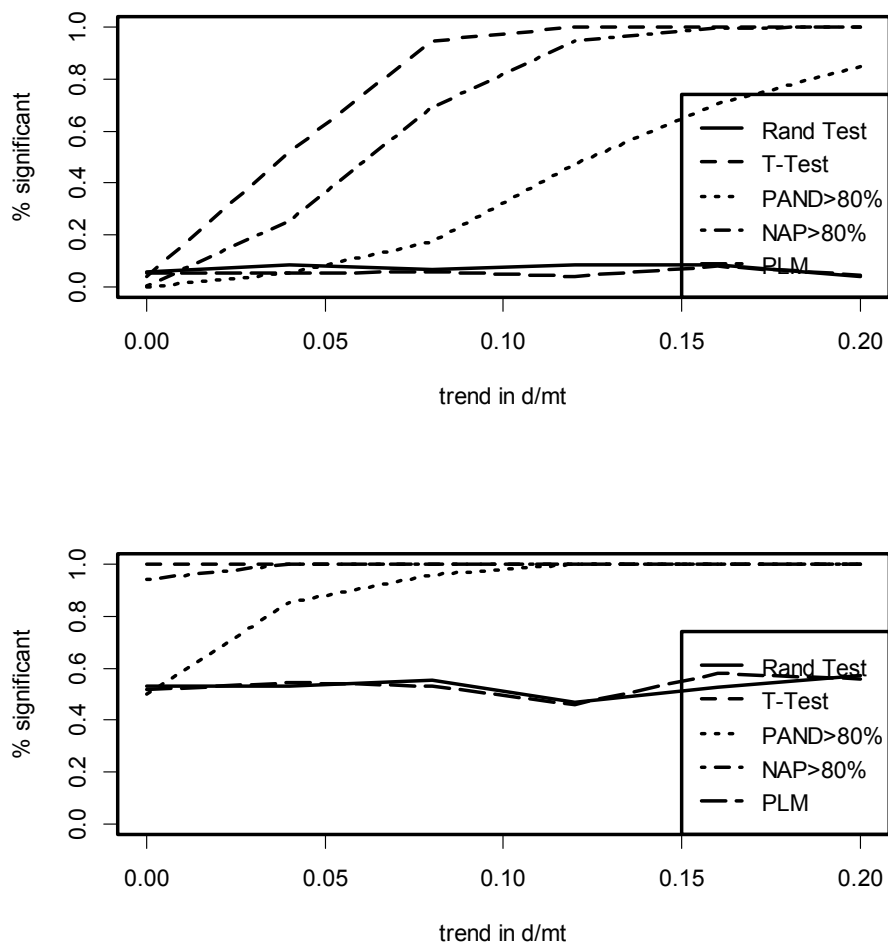


Figure. Alpha-error (upper plot) and testpower (lower plot) of different statistical techniques for analysing single-case data with an increasing trend.

## Parameters and Arguments

`n`: Number of studies to be created (default is `n = 1`).

`MT`: Measurement times of single cases in each study (default `MT = 20`).

`B.start`: Starting values of the B-phase. A single value (e.g., `B.start = 6`) defines the `B.start` for all studies and cases. A vector of starting values is given with the chain command (e.g., `B.start = c(6, 7, 8)`). A value between 0 and 1 is interpreted as a proportion (e.g., `B.start = c(0.3, 0.5, 0.8)`) are starting point at 30%, 50% and 80% of the length defined in `MT`). For randomized starting point, you can use `B.start = c("rand", 5, 10)` which computes random values for each study with values between 5 and 10 or `B.start = c("rand", 0.25, 0.75)` which computes random starting points between 25% and 75% of the length defined in `MT`.

`cases`: Number of cases per study. E.g., `n = 10, cases = 3, B.start = c(7,9,11)` creates 10 multiple-baseline designs with 3 persons (Starting point of the B-phase at 7, 9, 11, respectively) each.

`trend`: Defines a trend in `d` per measurement across all measurements. E.g., `trend = 0.1` is an increase of 0.1 standard deviations per measurement.

`m` and `s`: Mean and standard deviation of the sample distribution the data are drawn from.

`d.level`: A single level increase with the beginning of the B-Phase in standard deviations.

`d.slope`: Amount of increase in standard deviations per measurement time starting with the Beginning of the B-phase (e.g., `d.slope = 0.1` generates an incremental increase of 0.1 standard deviations for each measurement of the B-phase).

`rtt`: Reliability of the underlying simulated measurements (default is `rtt = 0.8`).

`concise`: If set `TRUE`, no information about the generated sample is given (default is `FALSE`).

`check`: If set `TRUE`, the generated samples mean, sd, and rtt is analysed (default is `FALSE`).

`round`: Rounds the measurement values to a specific decimal position (e.g., `round = 2`).

`extreme.p`: Probability of extreme values (e.g., `extreme.p = 0.05` gives a 5% probability of an extreme value; default = 0).

`extreme.d`: Range of an extreme value in d (e.g., `extreme.d = c(-7, -6)` results in extreme values within a range of -7 and -6 standard deviations). Default is `c(-4, -3)`. Caution: the first value must be smaller than the second, otherwise the procedure will fail.

`random.names`: Just a gimmick. If set `TRUE` cases have random first names. The selection of names is drawn from a list of the 2000 most popular names for births in the U.S.A. 2012 (1000 names for boys and 1000 names for girls). Default is `FALSE`.

## Aggregate multiple cases into one case

### Description

The `makesingleSC` command combines multiple single cases into one single case which can be used for further analysis.

The algorithm works the following way:

- 1) All measurements of each single-case are mean-centred with respect to the mean of the A-phase of each single-case.
- 2) The values of the A-phases of all single-cases are combined and ordered ascending with respect to the measurement times.
- 3) The values of the B-phases are combined and ordered as well.
- 4) The measurements of the B-phases are attached behind the A-phase measurements. The measurement times of the B-phase are shifted to start one measurement time after the A-phase.

### Parameters and Arguments

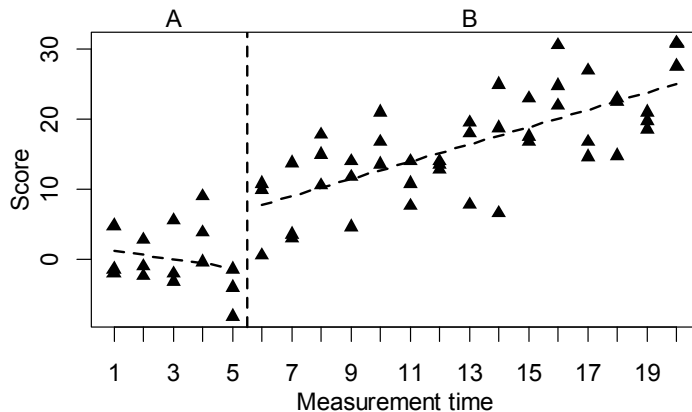
`data`, `B.start` and `MT`: Specification of the data. Read the “Introduction and example” chapter for a description.

`scale`: If set `TRUE`, values are standardized to the deviation of the A-phase. Default is `FALSE`.

`type`: By default values with the same measurement-time are added. If `type` is set to `mean` or `median`, values of the same MT are replaced with their mean or median. Default is `add`.

## Example

```
study1 <- rSC(3, d.level = 0.8, d.slope = 0.1, round = 0)
new.sc <- makesingleSC(study1)
plotSC(new.sc, type = "p", lines = "trend")
```



## \*(Experimental) Power Analysis

### Description

The `power.testSC` command conducts a Monte-Carlo study on the test-power and alpha-error of a randomization-test and a piecewise-regression model. The distribution values of the Monte-Carlo sample is either defined or automatically estimated based on the data of an actual study.

### Parameters and Arguments

`n`: Size of Monte-Carlo study (default is `n = 100`).

`MT`: Measurement times of single cases random study (default `MT = 20`).

`B.start`: Starting values of the B-phase. A single value (e.g., `B.start = 6`) defines the `B.start` for all studies and cases. A vector of starting values is given with the chain command (e.g., `B.start = c(6, 7, 8)`). A value between 0 and 1 is interpreted as a proportion (e.g., `B.start = c(0.3, 0.5, 0.8)`) are starting point at 30%, 50% and 80% of the length defined in `MT`). For randomized starting point, you can use `B.start = c("rand", 5, 10)` which computes random values for each study with values between 5 and 10 or `B.start = c("rand", 0.25, 0.75)` which computes random starting points between 25% and 75% of the length defined in `MT`.

`cases`: Number of cases per study. E.g., `n = 10, cases = 3, B.start = c(7,9,11)` creates 10 multiple-baseline designs with 3 persons (Starting point of the B-phase at 7, 9, 11, respectively) each.

`trend`: Defines a trend in `d` per measurement across all measurements. E.g., `trend = 0.1` is an increase of 0.1 standard deviations per measurement.

`m` and `s`: Mean and standard deviation of the sample distribution the data are drawn from.

`d.level`: A single level increase with the beginning of the B-Phase in standard deviations.

`d.slope`: Amount of increase in standard deviations per measurement time starting with the Beginning of the B-phase (e.g., `d.slope = 0.1` generates an incremental increase of 0.1 standard deviations for each measurement of the B-phase).

`rtt`: Reliability of the underlying simulated measurements (default is `rtt = 0.8`).

`extreme.p`: Probability of extreme values (e.g., `extreme.p = 0.05` gives a 5% probability of an extreme value; default = 0).

`extreme.d`: Range of an extreme value in `d` (e.g., `extreme.d = c(-7, -6)` results in extreme values within a range of -7 and -6 standard deviations). Default is `c(-4, -3)`. Caution: the first value must be smaller than the second, otherwise the procedure will fail.

`limit`: Minimal length of the A- und B-phase in the randomization sample. Default is `limit = 5`.

`exclude.equal`: If set FALSE, random distribution value that are equal to the observed distribution are counted as values of the distribution of the null-hypothesis. That is, they decrease the probability of rejecting the null-hypothesis (increase the p-value). Default is `exclude.equal = "auto"` (FALSE for multiple-baseline designs and TRUE for single baseline designs).

`alpha`: Alpha level used to calculate the proportion of significant tests. Default is 0.05.

`stat`: Defines the statistics the power analysis is computed for. The default `stat = c("rand.test", "plm")` computes a power-analysis for the randomization and the plm analysis.

`rand.test.stat`: Defines the statistic a randomization test is based on. The first values stipulates the statistic for the level-effect computation and the second value for the slope-effect computation. Default is `rand.test.stat = c("Mean B-A", "B")`.

## Example

```
> power.testSC(fillmissingSC(Grosche2011SC))
Compute Monte-Carlo power-analyses with the following parameters:
```

```
Sample studies      100
Cases per sample   3
M                   5.537086
SD                  3.039698
MT                  93
B.start             39 39 60
rtt                 0.8
d level             0.3274
d slope             0.0197
d trend             -0.0079
Extreme.p           0
Extreme.d           -4 -3
Alpha level         0.05
Exclude equal       FALSE
Limit               5
```

```
Test-Power in percent:
```

	Power	Alpha-error
Rand-Test: Mean B-A	12	0
Rand-Test: B	0	1
PLM: Level	85	4
PLM: Slope	100	6

## Reference

- Beretvas, S., & Chung, H. (2008). An evaluation of modified  $R^2$ -change effect size indices for single-subject experimental designs. *Evidence-Based Communication Assessment and Intervention*, 2, 120–128.
- Bulté, I., & Onghena, P. (2009). Randomization tests for multiple-baseline designs: An extension of the SCRT-R package. *Behavior Research Methods*, 41(2), 477–485. doi:10.3758/BRM.41.2.477
- Bulté, I., & Onghena, P. (2012). *SCRT: Single-Case Randomization Tests*. Abgerufen von <http://cran.r-project.org/web/packages/SCRT/index.html>
- Grosche, M. (2011). Effekte einer direkt-instruktiven Förderung der Lesegenauigkeit. *Empirische Sonderpädagogik*, 3(2), 147–161.
- Huitema, B. E., & Mckean, J. W. (2000). Design specification issues in time-series intervention models. *Educational and Psychological Measurement*, 60(1), 38–58.
- Parker, R. I., Hagan-Burke, S., & Vannest, K. (2007). Percentage of All Non-Overlapping Data (PAND) An Alternative to PND. *The Journal of Special Education*, 40(4), 194–204.
- Parker, R. I., & Vannest, K. (2009). An improved effect size for single-case research: Nonoverlap of all pairs. *Behavior Therapy*, 40(4), 357–367.
- Wise, E. A. (2004). Methods for analyzing psychotherapy outcomes: A review of clinical significance, reliable change, and recommendations for future directions. *Journal of Personality Assessment*, 82(1), 50–59.