

# Package ‘ACE’

April 7, 2025

**Title** Absolute Copy Number Estimation from Low-coverage Whole Genome Sequencing

**Description** Uses segmented copy number data to estimate tumor cell percentage and produce copy number plots displaying absolute copy numbers.

**Version** 1.25.0

**Author** Jos B Poell

**Maintainer** Jos B Poell <j.poell@amsterdamumc.nl>

**Depends** R (>= 3.4)

**Imports** Biobase, QDNAseq, ggplot2, grid, stats, utils, methods, grDevices, GenomicRanges

**biocViews** CopyNumberVariation, DNASeq, Coverage, WholeGenome, Visualization, Sequencing

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**URL** <https://github.com/tgac-vumc/ACE>

**RoxygenNote** 6.1.1

**Suggests** knitr, rmarkdown, BiocStyle

**VignetteBuilder** knitr

**NeedsCompilation** no

**git\_url** <https://git.bioconductor.org/packages/ACE>

**git\_branch** devel

**git\_last\_commit** 0696f37

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2025-04-07

## Contents

ACE-package . . . . .	2
ACEcall . . . . .	3
analyzegenomiclocations . . . . .	6
compresstemplate . . . . .	7
copyNumbersSegmented . . . . .	9
correlationmatrix . . . . .	9
forcesegmentsontemplate . . . . .	11
getadjustedsegments . . . . .	12
linkvariants . . . . .	14
loopsquaremodel . . . . .	16
objectsampletotemplate . . . . .	18
postanalysisloop . . . . .	19
runACE . . . . .	22
segmentstotemplate . . . . .	24
singlemodel . . . . .	26
singleplot . . . . .	28
squaremodel . . . . .	30
squaremodelsummary . . . . .	33
templatefromequalsegments . . . . .	34
twosamplecompare . . . . .	36
<b>Index</b>	<b>40</b>

---

ACE-package

*ACE package*

---

## Description

ACE is developed to analyze (low-coverage) whole genome sequencing data, infer absolute copies of chromosomal segments, estimate tumor cell percentage, and create visually appealing and highly interpretable copy number profiles. For preprocessing it utilizes the [QDNaseq package](#), but in principle any segmented copy number data can be used. The core function, [runACE](#), performs the whole pipeline from mapped reads to suggested copy number profiles. Functions such as [singlemodel](#) and [singleplot](#) enable further inspection and customization of individual samples. For further instructions, please consult the vignette for a walkthrough, or the individual function documentation for detailed function specifics.

## Details

ACE basically starts from segmented copy number data. To arrive at this point from mapped sequencing reads, it runs a set of default QDNaseq functions to obtain [QDNaseqCopyNumbers](#) objects with segmented data. For copy number plots, ACE uses [ggplot2](#). The following functions comprise the main functionality of ACE:

[runACE](#) Perform model fitting on all samples in a directory

[singlemodel](#) Perform model fitting on a single sample

`singleplot` Create an absolute copy number plot of a single sample

`squaremodel` Perform model fitting on a single sample using both cellularity and ploidy as variable

`getadjustedsegments` Use obtained model parameters to calculate adjusted segment values

`linkvariants` Link variant data to copy number data to estimate the number of mutant copies

`analyzegenomiclocations` Look up adjusted segment values of specific genomic locations

`postanalysisloop` Use obtained model parameters for all samples in a QDNAseq-object to print final output, e.g. copy number plots, adjusted segment data, mutation data linked to adjusted copy number data

The following functions provide some more "niche" functionality

`ACEcall` Call losses and gains for all segments and visualize in copy number profile

`twosamplecompare` Compare segments and copy number profile of two samples and plot overlay

`squaremodelsummary` Return or print a summary of a squaremodel

`loopsquaremodel` Perform squaremodel fitting on all samples in a QDNAseq-object

`forcesegmentsontemplate` Custom resegmentation with user-defined segment information

## License

This package is licensed under GPL.

## Author(s)

Jos B. Poell

---

ACEcall

*Categorize and plot subclonal, single, and double gains / losses*

---

## Description

ACEcall is the calling algorithm for ACE that utilizes the absolute copy number scaling to distinguish subclonal, single, and double losses and gains. For a segment to be "called", its segment mean needs to deviate from the general tumor ploidy both based on statistical significance and a large enough difference (both parameters can be set by the user).

## Usage

```
ACEcall(template, QDNAseqobjectsample = FALSE, cellularity = 1,
         ploidy = 2, standard, plot = TRUE, title, pcutoff,
         qcutoff = -3, subcutoff = 0.2, trncname = FALSE,
         cap = 12, bottom = 0, chrsubset, onlyautosomes = TRUE,
         sgc = c())
```

**Arguments**

template	Object. Either a data frame as created by <code>objectsampletotemplate</code> , or a QDNaseq-object
QDNaseqobjectsample	Integer. Specifies which sample to analyze from the QDNaseq-object. Required when using a QDNaseq-object as template. Default = FALSE
cellularity	Numeric. Used for rescaling bin and segment values. Printed on graph. Default = 1
ploidy	Integer. Assume the median of segments has this absolute copy number. Default = 2
standard	Numeric. Forces the given ploidy to represent this raw value. When omitted, the standard will be calculated from the data. When using parameters obtained from <code>squaremodel</code> , specify <code>standard = 1</code>
plot	Logical. Specifies whether the plot is created. When FALSE, only the data frame will be returned. Default = TRUE
title	Character string. Overwrites the automatically generated title
pcutoff	Numeric. Specifies the cutoff for statistical significance, without multiple testing correction, required to call a segment. Provide the desired log <sub>10</sub> -transformed p-value cutoff. When omitted (default), the q-value cutoff will be used instead.
qcutoff	Numeric. Specifies the cutoff for statistical significance, with multiple testing correction, required to call a segment. Provide the desired log <sub>10</sub> -transformed q-value cutoff. Default = -3
subcutoff	Numeric. Specifies the difference from the general tumor ploidy required to call a segment. Also used as margin for the other calls. Default = 0.2
trncname	Logical. In case of a QDNaseq object, the name of the sample is retrieved from the object and used as title. If set to TRUE, <code>trncname</code> truncates the sample name from the first instance of "_" in the name. You can also specify the regular expression here, e.g. <code>trncname = "-.*"</code> truncates the name from the first dash. Default = FALSE
cap	Integer. Influences your output copy number graphs. The upper limit of the y-axis is set at this number. When set to "max", it sets the cap to the maximum absolute copynumber value, rounded up. Bins and segments that exceed the cap are represented by a special mark. Recommended use between 8 and 16. Default = 12
bottom	Integer. Similar to cap, but for the lower limit of the y-axis. When set to "min", it sets the bottom to the minimum absolute copynumber value, rounded down. Bins and segments that subceed the bottom are represented by a special mark. Default = 0
chrsubset	Integer vector. Specify the chromosomes you want to plot. It will always take the full range of chromosomes in your subset, so specifying <code>chrsubset = c(4, 8)</code> will give the same plot as <code>chrsubset = 4:8</code> . When using a subset, ACEcall will not plot the cellularity and error on the plot.
onlyautosomes	Logical or integer. You can fill in an integer to specify how many autosomes your species has. When TRUE, ACEcall defaults to 22 (human) autosomes.

	When FALSE, ACEcall will also plot whichever other chromosomes are specified in the template, e.g. "X", "Y", "MT"
sgc	Integer or character vector. Specify which chromosomes occur with only a single copy in the germline

### Details

The color code in the plot is as follows (in parentheses the call in the data frame):

- **black** - not called (0)
- **gold** - subclonal gain (0.5)
- **turquoise** - subclonal loss (-0.5)
- **dark orange** - single copy gain (1)
- **blue** - single copy loss (-1)
- **red** - double copy gain (2)
- **dark blue** - double copy or full loss (-2)
- **purple** - amplification (3)

### Value

ACEcall returns a data frame that is similar to the input template, but it is supplemented with adjusted copy numbers, adjusted segment values, the segment mean and associated standard error, and the log<sub>10</sub> p-value and q-value, which signifies the probability that if the true segment mean equals the general tumor ploidy (rounded to an integer), the resulting or a more extreme segment mean would be found. The q-value is the result of a Benjamini-Hochberg correction of the p-value, taking into account all tested segments. Output is restricted to the specified chromosomal subset. If the argument `plot = TRUE`, ACEcall will also return an absolute copy number plot (a ggplot2-object) with the segments color-coded to specify the calls.

### Note

For biological reasons, a segment is called a double loss when it is highly likely that some cells do not harbor a single copy of the segment of interest. Therefore, a segment with exactly one copy in a triploid tumor will be called as a single loss, not a double loss. Note that ACEcall compares segments to integer copy numbers. It therefore needs to round the given ploidy to an integer as well. It uses the `round` function for this.

### Author(s)

Jos B. Poell

### See Also

[singleplot](#)

**Examples**

```
## simulated data assuming each chromosome comprises 100 bins
s <- jitter(c(1, 1, 0.8, 1.2, rep(1, 5), 1.4, rep(1, 13)), amount = 0)
n <- c(100, 100, 40, 60, rep(100, 5), 100, rep(100, 13))
bin <- 1:2200
chr <- rep(1:22, each = 100)
start <- rep(0:99*1000000+1, 22)
end <- rep(1:100*1000000, 22)
copynumbers <- jitter(rep(s,n), amount = 0.05)
segments <- rep(s, n)
template <- data.frame(bin = bin, chr = chr, start = start, end = end,
  copynumbers = copynumbers, segments = segments)
ACEcall(template, cellularity = 0.4, title = "sim")
## using segmented data from a QDNAseq-object
data("copyNumbersSegmented")
## for derivations of the parameters for this fit,
## see documentation on squaremodel
ACEcall(copyNumbersSegmented, QDNAseqobjectsample = 2,
  cellularity = 0.41, ploidy = 2.08, standard = 1)
```

---

analyzegenomiclocations

*Retrieve adjusted copy number information for specific genomic locations*

---

**Description**

analyzegenomiclocations searches an adjusted segment data frame for the specified genomic locations and reports the associated segment mean as "Copynumbers". If frequency of a variant is given (in percentage), mutant copies are also calculated. Make sure the specified cellularity is the same number as was used to create the adjusted segment data frame.

**Usage**

```
analyzegenomiclocations(segmentdf, Chromosome, Position,
  Frequency, cellularity, sgc = c())
```

**Arguments**

segmentdf	Data frame. Output of getadjustedsegments
Chromosome	Vector. Specifies the chromosome(s)
Position	Numeric vector. Specifies the base position(s) of interest
Frequency	Numeric vector. Optional. Used (in conjunction with cellularity) to calculate mutant copies. Enter a percentage.
cellularity	Numeric. Only required when calculating mutant copies. Use same number as used with getadjustedsegments
sgc	Integer or character vector. Specify which chromosomes occur with only a single copy in the germline

**Details**

The formula that calculates mutant copies only works if the variant is not present in normal tissue. In other words, make sure variants are not of germline origin. If you are interested in a heterozygous germline variant, you can calculate the "mutant copies" by subtracting  $(1/\text{cellularity} - 1)$ .

**Value**

Returns a data frame with Chromosome, Position, Copynumbers. When Frequency is given as an argument, also returns Mutant\_copies.

**Note**

Chromosome, Position, and Frequency can be single values or vectors. In the latter case, they need to be of equal length. Make sure the position you enter matches with the genome build used for alignment / binning of sequence reads.

When analyzing genomic locations on sex chromosomes of a male individual, make sure to specify `sgc = c("X", "Y")`

**Author(s)**

Jos B. Poell

**See Also**

[getadjustedsegments](#), [linkvariants](#)

**Examples**

```
## using segmented data from a QDNAseq-object
data("copyNumbersSegmented")
segmentdf <- getadjustedsegments(copyNumbersSegmented,
  QDNAseqobjectsample = 2, cellularity = 0.39)
analyzegenomiclocations(segmentdf, Chromosome = 5, Position = 26365569)
analyzegenomiclocations(segmentdf, Chromosome = 5, Position = 26365569,
  Frequency = 25, cellularity = 0.39)
```

---

compresstemplate

*Reduce the size of a template data frame*

---

**Description**

compresstemplate combines bins to reduce the size of a template data frame.

**Usage**

```
compresstemplate(template, factor = 20, funtype = "median")
```

### Arguments

template	Data frame as created by <a href="#">objectsampletotemplate</a>
factor	Integer. Number of bins to combine into a single bin. Default = 20
funtype	Character string. Specifies the function used to combine bins. Either "mean" or "median". Default = "median"

### Details

compresstemplate can be used when the number of features in a template is too large for downstream analyses. Bins are combined, resulting in larger bin sizes and a reduced number of features.

### Value

Returns a template data frame with increased bin size and reduced number of features compared to the input data frame.

### Note

Segmentation is slightly affected by this function due to the fact that the breakpoints will usually fall somewhere within a newly created bin. In case of funtype "mean" the segment value of this bin will be different from both adjacent bins and therefore create a 1-bin-segment. This "oversegmentation" is greatly reduced when using funtype "median" (especially when argument factor is and odd number).

### Author(s)

Jos B. Poell

### See Also

[objectsampletotemplate](#), [segmentstotemplate](#)

### Examples

```
## segmented data from a QDNaseq-object
data("copyNumbersSegmented")
template <- objectsampletotemplate(copyNumbersSegmented, index = 1)
ct <- compresstemplate(template, factor = 10)
length(template$bin)
length(ct$bin)
```



---

copyNumbersSegmented *Segmented data of two tumor samples*

---

### Description

A QDNAseq-object with segmented copy number data of two tumor samples: 'sample1' and 'sample2'

### Usage

```
data("copyNumbersSegmented")
```

### Format

A QDNAseqCopyNumbers S4 object with 3113 features (bins), each represented by copy number values and segment values of two samples

copyNumbersSegmented@assayData\$copynumber copy number data saved in a matrix, rownames are bins (chr:start-end), colnames are sampleNames

copyNumbersSegmented@assayData\$segmented segment values saved in a matrix, the rownames are bins (chr:start-end), colnames are sampleNames

### Value

A QDNAseqCopyNumbers S4 object

### Examples

```
data("copyNumbersSegmented")
## ploidyplotloop(copyNumbersSegmented, currentdir = ".")
singlemodel(copyNumbersSegmented, QDNAseqobjectsample = 1)
singleplot(copyNumbersSegmented, 1, cellularity = 0.79)
model2 <- singlemodel(copyNumbersSegmented, QDNAseqobjectsample = 2)
singleplot(copyNumbersSegmented, 2, cellularity = with(model2,
  minima[which(rerror==min(rerror))]))
```

---

correlationmatrix *Create a correlation matrix of all samples in a QDNAseq-object*

---

### Description

correlationmatrix runs the R stats cor function on the segmented data in a QDNAseqobject and returns the correlation matrix. The adjusted version equalizes the segments for each individual comparison, either through [twosamplecompare](#) or [templatefromequalsegments](#)

**Usage**

```
correlationmatrix(object, trncname = FALSE)

correlationmatrixadjusted(object, trncname = FALSE,
  equalsgments = FALSE, funtype = 'mean')
```

**Arguments**

object	QDNAseq-object
trncname	Logical. The name of the sample is retrieved from the object and used as title. If set to TRUE, trncname truncates the sample name from the first instance of "_" in the name. You can also specify the regular expression here, e.g. trncname = "-.*" truncates the name from the first dash. Default = FALSE
equalsgments	Logical or integer. If FALSE, the function will simply take all combined break-points and "resegment" both samples accordingly. When an integer is given, the function will create artificial segments that are all comprised of as many bins as entered with this argument, or 20 when set to TRUE. Default = FALSE
funtype	Character string. Specifies the function used to calculate new segment values. Alternative is "median". Default = "mean"

**Details**

Calculating and plotting correlation of segments between samples can be helpful to examine similarity / dissimilarity among samples.

**Value**

Returns a matrix with sample names defining both rows and columns and cells containing the pearson correlation of segment values of all bins of the intersecting samples.

**Note**

It is possible to visualize a correlation matrix with the R stats heatmap function. Although it is undoubtedly possible to adjust the function to give an interpretable plot, there are packages that facilitate this greatly. An example is the [corrplot](#) function from the eponymous package.

**Author(s)**

Jos B. Poell

**See Also**

[twosamplecompare](#), [templatefromequalsgments](#)

**Examples**

```
## using segmented data from a QDNaseq-object
data("copyNumbersSegmented")
correlationmatrix(copyNumbersSegmented)
correlationmatrixadjusted(copyNumbersSegmented)
```

---

forcesegmentsontemplate

*Custom resegmentation with user-defined segment information*

---

**Description**

forcesegmentsontemplate forces a user-defined segmentation pattern on a template. The input containing the segment information requires the chromosome name, start position, and end position of the segments. This can be provided manually as a data frame or as a data frame such as obtained by [getadjustedsegments](#). This means you can also use this function to force the segment pattern of one sample onto another sample. It is possible to retain the break points of the original input template.

**Usage**

```
forcesegmentsontemplate(segmentinput, template, QDNaseqobjectsample = FALSE,
                        combinesegments = FALSE, funtype = 'mean')
```

**Arguments**

segmentinput	Data frame containing segment information. Requires columns with chromosome name, start position, and end position. The function looks for column names containing "chr", "start", and "end" to find the required information
template	Object. Either a data frame as created by <a href="#">objectsamplletotemplate</a> , or a QDNaseq-object. Note that segmented data is not required for this function, unless combining segments
QDNaseqobjectsample	Integer. Specifies which sample to use as input from the QDNaseq-object. Required when using a QDNaseq-object as template. Default = FALSE
combinesegments	Logical. When TRUE, segment break points from the input template are retained. Default = FALSE
funtype	Character string. Specifies which function to use to calculate segment values. An alternative would be 'median'. Default = 'mean'

**Details**

This function only changes (or adds) the segments column of the input template. Copynumbers values are always retained, even if they fall out of the range of the segmentinput.

**Value**

Returns a template data frame with the columns bin, chromosome, copynumbers, and segments

**Author(s)**

Jos B. Poell

**See Also**

[getadjustedsegments](#), [twosamplecompare](#)

**Examples**

```
data("copyNumbersSegmented")
template <- objectsampletotemplate(copyNumbersSegmented, index = 1)
segments <- getadjustedsegments(copyNumbersSegmented, 2)[,1:3]
newtemplate <- forcesegmentsontemplate(segments, template)
first50M <- forcesegmentsontemplate(data.frame(chr = 1:22,
  start = rep(1, 22), end = rep(50000000, 22)), template)
singleplot(first50M)
```

---

getadjustedsegments	<i>Create a data frame with segment information corresponding to a model</i>
---------------------	--

---

**Description**

getadjustedsegments applies model parameters to segment data and returns a data frame with information of the individual segments, scaled according to the model.

**Usage**

```
getadjustedsegments(template, QDNAseqobjectsample = FALSE,
  cellularity = 1, ploidy = 2, sgc = c(),
  standard, log = FALSE)
```

**Arguments**

template	Object. Either a data frame as created by <a href="#">objectsampletotemplate</a> , or a QDNAseq-object
QDNAseqobjectsample	Integer. Specifies which sample to analyze from the QDNAseq-object. Required when using a QDNAseq-object as template. Default = FALSE
cellularity	Numeric. Used for rescaling bin and segment values. Default = 1
ploidy	Integer. Assume the median of segments has this absolute copy number. Default = 2

sgc	Integer or character vector. Specify which chromosomes occur with only a single copy in the germline
standard	Numeric. Force the given ploidy to represent this raw value. When omitted, the standard will be calculated from the data. When using parameters obtained from squaremodel, specify standard = 1
log	Logical. When TRUE, log2-values are calculated straight from raw data, unadjusted! Convenience function to resemble DNACopy output as used for ABSOLUTE and others. Default = FALSE

### Details

The output contains two columns for segment mean. The first is the adjusted segment value, the second (Segment\_Mean2) is the mean of the adjusted copy number values. I do not know how the QDNAseq or DNACopy calculates the segment mean, but there is always a very small difference between the two. The P\_log10 is the 10-base log of the two-sided probability that the real segment mean is the integer closest to the segment mean. While this gives an indication of the chance that a segment is subclonal, it should be interpreted with care. Because segments usually comprise many bins, these values can easily be very low. A small bias in the normalization can cause "significant", but not necessarily relevant results.

### Value

Returns a data frame with segment information

### Note

If your data contains sex chromosomes, then make sure to specify `sgc = c("X", "Y")` when analyzing data from a male individual.

### Author(s)

Jos B. Poell

### See Also

[analyzegenomiclocations](#), [postanalysisloop](#)

### Examples

```
## using segmented data from a QDNAseq-object
data("copyNumbersSegmented")
singlemodel(copyNumbersSegmented, QDNAseqobjectsample = 2)
getadjustedsegments(copyNumbersSegmented, QDNAseqobjectsample = 2,
  cellularity = 0.39)
```

---

linkvariants	<i>Append columns with total genomic copies and mutant copies to a file with variant/mutation data</i>
--------------	--

---

### Description

linkvariants combines copy number data, estimated tumor cell percentage, and variant allele frequency (e.g. mutation data) to calculate how many variant (mutant) copies the tumor genome harbors. It requires a data frame or tab-delimited file with variant data and a data frame or tab-delimited file with adjusted segment data as obtained with `getadjustedsegments`. Also make sure to provide the correct (estimated) cellularity. Output is a file with "\_ACE" added to the original file name before the extension. It can either be a copy of the input with `Copynumbers` and `Variant_copies` appended as extra columns at the end, or a file with the columns `Chromosome`, `Position`, `Frequency`, `Copynumbers` and `Mutant_copies`. To perform batch analysis, use [postanalysisloop](#). linkvariants can provide upper and lower bounds of a confidence interval if read depth is given.

### Usage

```
linkvariants(variantdf, segmentdf, cellularity = 1, hetSNPs = FALSE,
             chrindex=1, posindex=2, freqindex, altreadsindex,
             totalreadsindex, refreadsindex, confidencelevel=FALSE,
             append=TRUE, outputdir, sgc = c())
```

### Arguments

variantdf	Data frame or character string. File path to tab-delimited text (either .tsv, .csv, .txt or .xls) containing variant data or the corresponding data frame. File must contain a header and columns for chromosome, position, and frequency of the mutation. Optionally a column with read depth information. If frequency is missing, <code>altreads + totalreads</code> or <code>altreads + refreads</code> is required.
segmentdf	Data frame or character string containing file path of tab-delimited text with segment data. Expects data in the format provided by <code>getadjustedsegments</code> with argument <code>log=FALSE</code> .
cellularity	Numeric. Used to infer variant copies from frequency and total copies. Default = 1
hetSNPs	Logical. If TRUE, half of the germline copies are assumed to be variant. Default = FALSE
chrindex	Integer. Column index in input file specifying the chromosome associated with the genomic location. Default = 1
posindex	Integer. Column index in input file specifying the position on the chromosome associated with the genomic location. Default = 2
freqindex	Integer. Column index in input file specifying the frequency (as a percentage) of the variant
altreadsindex	Integer. Column index in input file specifying the number of variant-supporting reads

totalreadsindex	Integer. Column index in input file specifying the read depth at the genomic location of the variant
refreadsindex	Integer. Column index in input file specifying the number of reference-supporting reads
confidencelevel	Numeric or logical. If read depth information is available, calculate the upper and lower bounds of this confidence level for the frequency and the number of variant copies of each variant. Will be skipped if FALSE. Default = FALSE
append	Logical. When TRUE, appends the output columns to the original mutation input file, but it still saves the result in a new file. When FALSE, the output file will only contain the columns "Chromosome", "Position", "Frequency", "Copy-numbers", and "Mutant_copies". Default = TRUE
outputdir	Character string. Convenience function to save output into a custom directory
sgc	Integer or character vector. Specify which chromosomes occur with only a single copy in the germline

### Details

The default formula that calculates mutant copies works if the variant is not present in normal tissue. If you are interested in heterozygous germline variant, you can set the argument `hetSNPs` to TRUE. The confidence intervals are calculated using the `binom.test` function.

### Value

Prints output to a tab-delimited file, or returns a data frame with columns added for copies and mutant copies.

### Note

Make sure the variant data matches with the genome build used for alignment / binning of sequence reads for copy number analysis. If the resulting `Variant_copies` are very low, the variant allele frequencies were probably provided as fraction, not percentage. Just multiply by 100. If your data contains sex chromosomes, then make sure to specify `sgc = c("X", "Y")` when analyzing data from a male individual.

### Author(s)

Jos B. Poell

### See Also

[getadjustedsegments](#), [analyzegenomiclocations](#), [postanalysisloop](#)

### Examples

```
## using manually simulated mutation data
## see vignette for more practical uses
data("copyNumbersSegmented")
```

```

segmentdf <- getadjustedsegments(copyNumbersSegmented,
  QDNaseqobjectsample = 2, cellularity = 0.38)
Gene <- c("CASP8", "CDKN2A", "TP53")
Chromosome <- c(2, 9, 17)
Position <- c(202149589, 21971186, 7574003)
Frequency <- c(47.46, 36.28, 43.48)
AltReads <- c(345, 198, 284)
variantdf <- data.frame(Gene, Chromosome, Position, Frequency, AltReads)
linkvariants(segmentdf, variantdf, cellularity = 0.38,
  chrindex = 2, posindex = 3, freqindex = 4)
linkvariants(segmentdf, variantdf, cellularity = 0.38,
  chrindex = 2, posindex = 3, freqindex = 4,
  altreadsindex = 5, confidencelevel = 0.9)

```

---

loopsquaremodel

*Create squaremodel summaries for all samples in a QDNaseq-object*


---

## Description

loopsquaremodel performs a [squaremodel](#) fitting on all samples in a QDNaseq-object. It prints the summaries of the squaremodels to the specified output directory. It also returns a list with squaremodels which can be saved to a variable.

## Usage

```

loopsquaremodel(object, ptop = 5, pbottom = 1, prows = 100,
  method = 'RMSE', penalty = 0, penploidy = 0,
  outputdir, imagetype = 'pdf', trncname = FALSE,
  returnmodels = FALSE, printplots = TRUE,
  printobjectsummary = TRUE)

```

## Arguments

object	QDNaseq-object
ptop	Numeric. Sets the highest ploidy at which to start testing fits. Default = 5
pbottom	Numeric. Sets the lowest ploidy to be tested. Default = 1
prows	Integer. Sets the resolution of the ploidy-axis. Determines how many decrements are used to get from ptop to pbottom (see below). Therefore, the actual number of rows is actually prows + 1. Default = 100
method	Character string specifying which error method to use. For more documentation, consult the vignette. Can be "RMSE", "SMRE", or "MAE". Default = "RMSE"
penalty	Numeric. Penalizes fits at lower cellularities. Suggested values between 0 and 1. Default = 0 (no penalty)
penploidy	Numeric. Penalizes fits that diverge from 2N with the formula $(1 + \text{abs}(\text{ploidy} - 2))^{\text{penploidy}}$ . Default = 0
outputdir	Character string. Print the plots to this directory



imagetype	Character string. Plots are printed to file using this graphics device. "pdf" will result in pdf-files containing 8 pages with individual plots, while the other devices print 2x4 mosaics per sample. Default = "pdf"
trncname	Logical. If set to TRUE, trncname truncates the sample name from the first instance of "_" in the name. You can also specify the regular expression here, e.g. trncname = "-.*" truncates the name from the first dash. Default = FALSE
returnmodels	Logical. Return the squaremodel results as a list. If set to TRUE, the entire squaremodel is returned for each sample. You can also specify which aspects of the model you wish to capture, for instance returnmodels = c("penalty", "minimadf"). The sample name will always be included as the first listed item. Default = FALSE
printplots	Logical. Print the plots to file. Note that the object summary is not affected by this argument. Default = TRUE
printobjectsummary	Logical. Print the object summary to file. Default = TRUE

### Details

This function is basically the squaremodel equivalent of [ploidyplotloop](#). One key difference is the output. loopsquaremodel makes a single page summary of each sample in the object. It can also return a list with the squaremodels of each sample as a variable within the R environment. The squaremodels are supplemented with the sample names. The output printed to file is optional (though by default enabled).

### Value

Optionally returns a list with squaremodels (or the selected items of interest from the models) for each sample in a QDNAseq-object. If printplots is TRUE, the plots will be printed to file. If printobjectsummary is TRUE, an object summary will be printed, containing the matrixplot and the copy number plot of the best fit of each sample.

### Note

In case of large numbers of samples, you may have to set printobjectsummary to FALSE to prevent file size or memory issues. For similar reasons, the default of returnmodels is FALSE.

### Author(s)

Jos B. Poell

### See Also

[squaremodelssummary](#), [squaremodel](#), [ploidyplotloop](#)

### Examples

```
## Not run:
data("copyNumbersSegmented")
loopsquaremodel(copyNumbersSegmented, penalty = 0.5, penploidy = 0.5)
```

```
## End(Not run)
lsm <- loopsquaremodel(copyNumbersSegmented, printplots = FALSE,
  printobjectsummary = FALSE, penalty = 0.5, penploidy = 0.5,
  returnmodels = TRUE)
ls(lsm[[1]])
lsm[[1]]$samplename
lsm[[1]]$matrixplot
```

---

objectsampletotemplate

*Converts data of a sample in a QDNAseq-object to a template for ACE functions*

---

### Description

objectsampletotemplate parses QDNAseq-objects into the data frame structure used by various ACE functions. These functions call objectsampletotemplate itself when necessary, but it might be convenient to make a template if you expect some repeated use of the functions or if you want to make manual adjustments to the template.

### Usage

```
objectsampletotemplate(copyNumbersSegmented, index = 1)
```

### Arguments

copyNumbersSegmented  
QDNAseqCopyNumbers object with segmented data

index  
Integer. Specifies the index of the sample

### Value

Returns a template data frame with the columns bin, chromosome, copynumbers, and segments (when available)

### Note

If segment values are missing, the function will still return a data frame, but only containing the copynumbers column, not the segments column

### Author(s)

Jos B. Poell

### See Also

[singlemodel](#), [squaremodel](#), [singleplot](#)

**Examples**

```
data("copyNumbersSegmented")
template <- objectsampletotemplate(copyNumbersSegmented, index = 1)
```

---

postanalysisloop	<i>Batch analysis of samples in a QDNAseq-object for which models have been chosen</i>
------------------	--

---

**Description**

When models have been chosen for all (or just multiple) samples in a QDNAseq-object, this function can be used to perform a batch analysis on those samples. This encompasses printing segment data, printing copy number plots, and linking mutation data.

**Usage**

```
postanalysisloop(copyNumbersSegmented, modelsfile, variantdata,
  prefix="", postfix="", trncname=FALSE, inputdir=FALSE,
  hetSNPs=FALSE, chrindex=1, posindex=2, freqindex,
  altreadsindex, totalreadsindex, refreadsindex,
  confidencelevel=FALSE, append=TRUE,
  dontmatchnames=FALSE, printsegmentfiles=TRUE,
  printnewplots=TRUE, imagetype='pdf',
  onlyautosomes=TRUE, outputdir=".", log=FALSE,
  segext='tsv', genderci)
```

**Arguments**

copyNumbersSegmented	QDNAseq-object with segmented data or file path of an rds-file containing a QDNAseq-object
modelsfile	Character string or data frame. When a character, it specifies the file path of a tab-delimited text containing model variables of samples. Expects columns with a header. It contains at least two columns: the first specifying the sample names and the second specifying the cellularity. The third column is the ploidy of the samples. When omitted, it is assumed to be 2. The fourth column is the standard of the samples. When omitted, it is calculated from the data in the object. The fitpicker.tsv file created by <a href="#">runACE</a> can be used as modelsfile after cellularity of the likely fit is specified in the second column.
variantdata	Character string. Specifies directory containing variant data of samples. Optional. When argument inputdir is used, the function will first see if this argument is specified, if not it will check if the directory inputdir/variantdata exists, if not it will look for the variant files in inputdir itself. When inputdir is not used and this argument is omitted, the function will not link variant data. Mutation files need to have the same file extensions, which can be either .csv, .tsv, .txt, or .xls.

prefix	Character string. Used when a uniform character string precedes the sample name in the file name. E.g. "mutations_sample1.csv" has prefix "mutations_". Default = ""
postfix	Character string. As prefix, but then after the sample name. E.g. "sample1_somatics.csv" has postfix "_somatics". Default = ""
trncname	Logical. When TRUE, truncates the sample names of the QDNAseq-object starting from the first "_", or specify a character string with your regular expression of choice (trncname uses the gsub function). NOTE: use only when this will provide matches with the mutation files and the sample names in the modelsfile. Default = FALSE
inputdir	Character string. Specifies the directory which contains the files to be analyzed. Convenience function. Reduces the amount of arguments required when all data is available in the same directory: the QDNAseq-object, a file named "models.tsv" with the model parameters, and the mutation data (either in the inputdir itself or in a subdirectory "mutationdata"). Specifying the first arguments (copyNumbersSegmented, modelsfile, mutationdata) will take priority. When missing it will look in inputdir. When multiple rds-files are present in the inputdir, it will try the first one. Note: the path specified has no consequences for the location of the output. Default = FALSE
hetSNPs	Logical. If TRUE, half of the germline copies are assumed to be variant. Default = FALSE
chrindex	Integer. Column index in input file specifying the chromosome associated with the genomic location. Default = 1
posindex	Integer. Column index in input file specifying the position on the chromosome associated with the genomic location. Default = 2
freqindex	Integer. Column index in input file specifying the frequency (as a percentage) of the variant
altreadsindex	Integer. Column index in input file specifying the number of variant-supporting reads
totalreadsindex	Integer. Column index in input file specifying the read depth at the genomic location of the variant
refreadsindex	Integer. Column index in input file specifying the number of reference-supporting reads
confidencelevel	Numeric or logical. If read depth information is available, calculate the upper and lower bounds of this confidence level for the frequency and the number of variant copies of each variant. Will be skipped if FALSE. Default = FALSE
append	Logical. When TRUE, appends the output columns to the original mutation input file, but it still saves the result in a new file. When FALSE, the output file will only contain the columns "Chromosome", "Position", "Frequency", "Copy-numbers", and "Mutant_copies" (and including the upper and lower bounds of the frequency and variant copies confidence interval, when applicable). Default = TRUE

dontmatchnames	Logical. When TRUE, the model variables are called by the index of the sample in the QDNAseq-object. This will only work if the order of samples in the object exactly matches the order of samples in the modelsfile. Use with caution! This is somewhat of an emergency option if for some reason the name matching is not working. I recommend trying to get the name matching to work. Default = FALSE
printsegmentfiles	Logical. When TRUE, prints a tab-delimited text file for each sample into a "segmentfiles" folder. Default = TRUE
printnewplots	Logical. When TRUE, prints plots into a "newplots" folder in the specified image type. Default = TRUE
imagetype	Character string specifying the image type graphics device. Default = "pdf"
onlyautosomes	Logical or integer. Specifies whether only or which autosomes are plotted. For more documentation, see <code>singleplot</code>
outputdir	Character string. Save output into this custom directory. Default = "./"
log	Logical or integer. Use log conversion for creating segments output. Default = FALSE
segext	Character string specifying the extension for the segments output. Default = "tsv"
genderci	Integer. Column index in modelsfile or data frame specifying the gender of the corresponding sample. See note

### Details

If your input is tailored for this function, you could run it without any arguments! Most arguments help with matching sample names in the QDNAseq-object, the modelsfile, and the names and columns of the files containing variant data. You can `trim` the name of the file with variant data using the `prefix` (everything before the name) and `postfix` (everything after the name, but before the file extension) arguments to match your sample names. `trncname` might help trimming the name in the QDNAseq-object, but be sure it still matches the sample names in the modelsfile (and mutation data file names when applicable).

### Value

Prints the specified output to an indicated directory. Returns a list of copy number plots.

### Note

The use of `inputdir` and `outputdir` should be fairly robust. However, using irregular file paths might cause problems. If you suspect problems with file paths, try setting the working directory to the intended `inputdir`.

If you intend to plot or analyze variant data on sex chromosomes, make sure you specify the gender of each individual using the `genderci` option. The function will look for the gender in the indicated column number of the modelsfile (or data frame). Suggested indication within this column is "M" for male and "F" for female. When missing, the function defaults to "F".

**Author(s)**

Jos B. Poell

**See Also**[getadjustedsegments](#), [linkvariants](#), [runACE](#)**Examples**

```
## see the vignette for examples
## Not run:
data("copyNumbersSegmented")
postanalysisloop(copyNumbersSegmented, "models.tsv", "variantdata",
  outputdir = "loop_output")

## End(Not run)
```

runACE

*Absolute Copy number Estimation***Description**

ACE scales copy number data to fit with integer copy numbers, providing an estimate for tumor cell percentage in the process. runACE uses segmented data from the QDNAseq package. A folder with either bam-files (aligned sequencing data) or rds-files of QDNAseq-objects can be used as input. Model fitting and production of all output files (except "parameters.tsv") is executed by ploidyplootloop, which handles one QDNAseq-object at a time.

**Usage**

```
runACE(inputdir = "./", outputdir, filetype = 'rds', genome = 'hg19',
  binsizes, ploidy = 2, imagetype = 'pdf', method = 'RMSE', penalty = 0,
  cap = 12, bottom = 0, trncname = FALSE, printsummaries = TRUE,
  savereadcounts = FALSE, autopick = FALSE)
```

```
ploidyplootloop(copyNumbersSegmented, currentdir, ploidy = 2,
  imagetype = 'pdf', method = 'RMSE', penalty = 0, cap = 12,
  bottom = 0, trncname = FALSE, printsummaries = TRUE,
  autopick = FALSE)
```

**Arguments**

`inputdir` Character string specifying the directory containing the files you want analyzed. Note: will analyze ALL rds-files or bam-files in the given directory. Default = `"./"`

`copyNumbersSegmented` QDNAseq-object with segmented data

outdir, currentdir	Character string specifying the directory to which ACE should write the output. When missing, ACE will try to write to inputdir. For ploidyplotloop, currentdir is required.
filetype	Character string specifying the file type of your input, either "bam" or "rds". Default = "rds"
genome	Character string specifying genome and version. Availability depends on QD-Naseq. Default = "hg19"
binsizes	Numeric vector, specifying which binsizes (in kbp) to analyze. Possible values are 1, 5, 10, 15, 30, 50, 100, 500, and 1000. When omitted, defaults to c(100,500,1000)
ploidies	Numeric vector, specifying which ploidies (N) to analyze. Use positive natural numbers. Default = 2
imagetype	Character string specifying the image type graphics device, default = "pdf"
method	Character string specifying what error method to use. See also section "Error methods". Default = "RMSE"
penalty	Numeric value. Penalizes fits at lower cellularities. Suggested values between 0 and 1. Default = 0 (no penalty)
cap, bottom	Integer. Influences your output copy number graphs. The upper and lower limits of the y-axis are set at these values. Bins and segments that exceed/subceed the cap/bottom are represented by a special mark. Default = 12 and 0 respectively
trncname	Logical. Convenience functionality. If all your samples have a certain extension to their name, you can use this to truncate this extension and be left with the actual sample name. When TRUE, the regular expression is "_.*". That means it will chop off everything from the sample name starting with the first underscore. Instead of a logical, you can specify a character string to match your regular expression of choice. You can test whether it will work with the gsub function, since this is what ACE uses to truncate names. Default = FALSE
printsummaries	Logical. If you do not want the big summary files, you can set this argument to FALSE. If you still want the summary files containing only error plots, you can set this to 2. Default = TRUE
savereadcounts	Logical. When set to TRUE, readCounts-objects will be saved. Default = FALSE
autopick	Logical. When set to TRUE, ACE will fill in the cellularity of the best fit in the column likely_fit of the fitpicker file(s). Default = FALSE

## Details

Since this is the core functionality of ACE, extensive documentation is available in the vignette.

## Value

runACE and ploidyplotloop do not return any values, they print all their output to the indicated location. The output comprises

- the file "parameters.tsv" which simply reports the used parameters

- rds-files (only in case you had bam-files as input)
- for each ploidy a "fitpicker.tsv" file which can be used for selecting the most likely fits
- a summary file of likely fits and error plot of each sample (if printsummaries is set to TRUE)
- a summary file of all error plots (if printsummaries is set to TRUE or 2)
- a directory with copy number plots of the likely fits of all samples
- a directory for each sample, containing the error plot, a summary file with all fits of the sample, and individual copy number plots of all fits in a subdirectory

### Note

You can use the example data for testing: see Examples

### Author(s)

Jos B. Poell

### See Also

[singlemodel](#), [squaremodel](#), [singleplot](#)

### Examples

```
## Not run:
runACE("./bam/", outputdir = "./results", penalty = 0.5,
       binsizes = c(100, 1000), imagetype = 'png')
data("copyNumbersSegmented")
ploidyplotloop(copyNumbersSegmented, ".", ploidies = c(2,3))

## End(Not run)
```

---

segmentstotemplate	<i>Create a template data frame from input that only provides segment information</i>
--------------------	---

---

### Description

segmentstotemplate "explodes" segment data and creates a data frame with information for all bins. This enables the use of data limited to segments in ACE-related functions.

### Usage

```
segmentstotemplate(segmentdf, chrcki = 1, startci = 2, endci = 3,
                   binsci = 4, meanci = 5, seci, sdci, log = FALSE)
```



**Arguments**

segmentdf	Data frame or file path. Data frame or file path to tab-delimited file with segment data
chrcki	Integer. Specifies the index of the column containing chromosome information. Default = 1
startci	Integer. Specifies the index of the column containing start positions. Default = 2
endci	Integer. Specifies the index of the column containing end positions. Default = 3
binsci	Integer. Specifies the index of the column containing number of bins or number of probes. Default = 4
meanci	Integer. Specifies the index of the column containing the segment value. Default = 5
secki	Integer. Optional. Specifies the index of the column containing standard errors of the segment mean. See details below
sdcik	Integer. Optional. Specifies the index of the column containing standard deviations of the segment mean. See details below
log	Logical or integer. If the data is given in log-values, the data is converted to linear scale using the given log base, or inverting the natural logarithm when set to TRUE. Default = FALSE

**Details**

Even though there is no copy number information for each bin, this needs to be provided for several ACE-related functions. By default, `segmentstotemplate` will just "copy" the segment data. If the segment data comes with either standard deviations or standard errors, it is possible to "simulate" copy number data. To do so, `segmentstotemplate` will use the R stats `rnorm` function.

**Value**

Returns a template data frame.

**Note**

- Keep in mind that copy number data is used in some functions, notably [ACEcall](#). If information on the variance of copy number values within segments is missing, calls are only based on adjusted segment values, because the p-values are all 0.
- Also note that resulting templates should not be used for any of the functions that perform re-segmentation (e.g. [twosamplecompare](#)), or the results should be interpreted with due caution.
- The functions [singlemodel](#) and [squaremodel](#) only use segment values and can be used to analyze the resulting template data frames. [singleplot](#) can also be used without problem.
- This function works fine with high resolution data, but the resulting template can be a bit much for other functions. For instance, the segmented data files provided by TCGA are the result of SNP arrays that have a resolution of roughly 1.5 kbp. The function [compresstemplate](#) can help out to make input more manageable.

**Author(s)**

Jos B. Poell

**See Also**[compresstemplate](#)

singlemodel

*Calculate potential fits for a single sample***Description**

singlemodel performs the basic fitting algorithm of ACE on a single sample. Input can be either a template or a QDNAseq-object with the index of the sample specified. Returns a list with input parameters (ploidy, standard, and penalty) and model characteristics (calculated minima, the relative error corresponding with the minima, and the errors calculated at every cellularity). It also returns the plot associated with the error list. The minima represent cellularities, as can be seen in the plot.

**Usage**

```
singlemodel(template, QDNAseqobjectsample = FALSE, ploidy = 2,
            standard, method = 'RMSE', exclude = c("X", "Y"),
            sgc = c(), penalty = 0, highlightminima = TRUE)
```

**Arguments**

template	Object. Either a data frame as created by <a href="#">objectsampletotemplate</a> , or a QDNAseq-object
QDNAseqobjectsample	Integer. Specifies which sample to analyze from the QDNAseqobject. Required when using a QDNAseq-object as template. Default = FALSE
ploidy	Integer. Calculate fits assuming the median of segments has this absolute copy number. Default = 2
standard	Numeric. Force the given ploidy to represent this raw value. When omitted, the standard will be calculated from the data
method	String character specifying which error method to use. For more documentation, consult the vignette. Can be "RMSE", "SMRE", or "MAE". Default = "RMSE"
exclude	Integer or character vector. Specifies which chromosomes to exclude for model fitting. Default = c("X", "Y")
sgc	Integer or character vector. Specifies which chromosomes occur with only a single copy in the germline
penalty	Numeric value. Penalizes fits at lower cellularities. Suggested values between 0 and 1. Default = 0 (no penalty)
highlightminima	Logical. Minima are highlighted in the errorplot by a red color. Default = TRUE

**Details**

All ACE fitting algorithms work by calculating "expected values" of integer copies given a certain cellularity. It calculates these expected values for 1-12 copies at cellularities 0.05-1 (in increments of 0.01). First of all, this means that fits at cellularities below 0.05 are not calculated. These low-cellularity fits will not give very meaningful results, and only obscure more plausible fits. Second, it means that 0 copies and >12 copies are not "fitted". This prevents fits predicting many and/or large segments with 0 or >12 copies, which is biologically unlikely. More explanation is given in the vignette.

**Value**

Returns a list, containing

ploidy	Absolute copy number that corresponds with the median segment value
standard	Ploidy corresponds to this raw data value. Unless specified as argument, it corresponds to the median segment value
method	Applied error method
penalty	Applied penalty factor
minima	Vector with cellularities at which the error reached a minimum
rerror	Vector with relative errors corresponding to the minima
errorlist	List of errors of all cellularities tested
errorplot	ggplot2-graph of the relative errors calculated at each cellularity

**Note**

singlemodel() only needs a data frame with columns named chr and segments. Every row should contain an individual genomic feature, i.e. a bin or a probe. If you have data with each row representing a segment, and the size of the segment given in a column (e.g. NumBins or NumProbes), you can create the data frame as follows:

```
template <- data.frame(chr = rep(Chromosome, NumProbes), segments = rep(SegmentMean, NumProbes))
```

Alternatively you can look into [segmentstotemplate](#).

If your data contains sex chromosomes and you wish to include these for model fitting, then make sure to specify `exclude = c()`, and `sgc = c("X", "Y")` when analyzing data from a male individual.

**Author(s)**

Jos B. Poell

**See Also**

[objectsampletotemplate](#), [squaremodel](#), [singleplot](#)

## Examples

```
## toy data assuming each chromosome comprises 100 bins
s <- jitter(c(1, 1, 0.8, 1.2, rep(1, 5), 1.4, rep(1, 13)), amount = 0)
n <- c(100, 100, 40, 60, rep(100, 5), 100, rep(100, 13))
df <- data.frame(chr = rep(1:22, each = 100), segments = rep(s, n))
singlemodel(df)
singlemodel(df, ploidy = 3)
singlemodel(df, method = 'MAE', penalty = 0.5)
singlemodel(df, exclude = 1:3)

## using segmented data from a QDNaseq-object
data("copyNumbersSegmented")
singlemodel(copyNumbersSegmented, QDNaseqobjectsampl = 2)
```

---

singleplot

*Plot an absolute copy number profile for a single sample*

---

## Description

singleplot is the core plotting function of ACE. Input can be either a template or a QDNaseq-object with the index of the sample specified. Several of the arguments are parameters obtained from model fitting. Returns a ggplot2 graph with absolute copies on the y-axis and genomic position on the x-axis.

## Usage

```
singleplot(template, QDNaseqobjectsampl = FALSE,
           cellularity = 1, error, ploidy = 2, standard, title,
           trncname = FALSE, cap = 12, bottom = 0, chrsubset,
           onlyautosomes = TRUE, sgc = c())
```

## Arguments

template	Object. Either a data frame as created by <a href="#">objectsampltotemplate</a> , or a QDNaseq-object
QDNaseqobjectsampl	Integer. Specifies which sample to analyze from the QDNaseq-object. Required when using a QDNaseq-object as template. Default = FALSE
cellularity	Numeric. Used for rescaling bin and segment values. Printed on graph. Default = 1
error	Numeric. When given, it is printed on the graph below cellularity.
ploidy	Integer. Assume the median of segments has this absolute copy number. Default = 2
standard	Numeric. Force the given ploidy to represent this raw value. When omitted, the standard will be calculated from the data. When using parameters obtained from <code>squaremodel</code> , specify <code>standard = 1</code>

title	Character string. Overwrites the automatically generated title
trncname	Logical. In case of a QDNAseq object, the name of the sample is retrieved from the object and used as title. If set to TRUE, trncname truncates the sample name from the first instance of "_" in the name. You can also specify the regular expression here, e.g. trncname = "-.*" truncates the name from the first dash. Default = FALSE
cap	Integer. Influences your output copy number graphs. The upper limit of the y-axis is set at this number. When set to "max", it sets the cap to the maximum absolute copynumber value, rounded up. Bins and segments that exceed the cap are represented by a special mark. Recommended use between 8 and 16. Default = 12
bottom	Integer. Similar to cap, but for the lower limit of the y-axis. When set to "min", it sets the bottom to the minimum absolute copynumber value, rounded down. Bins and segments that subceed the bottom are represented by a special mark. Default = 0
chrsubset	Integer vector. Specify the chromosomes you want to plot. It will always take the full range of chromosomes in your subset, so specifying chrsubset = c(4, 8) will give the same plot as chrsubset = 4:8. When using a subset, singleplot will not plot the cellularity and error on the plot. Therefore, you can use this to make a copy number plot without this information by specifying chrsubset = 1:22
onlyautosomes	Logical or integer. You can fill in an integer to specify how many autosomes your species has. When TRUE, singleplot defaults to 22 (human) autosomes. When FALSE, singleplot will also plot whichever other chromosomes are specified in the template, e.g. "X", "Y", "MT". You can combine this argument with chrsubset, for instance chrsubset[1:23] to only include chromosome X (provided this is the 23rd chromosome)
sgc	Integer or character vector. Specify which chromosomes occur with only a single copy in the germline

**Value**

Returns a graph generated through the ggplot2 package.

**Note**

singleplot expects chromosome names, as specified in the chr column of the template, to be either just the integer chromosome number, or "chr" followed by the chromosome number. This is strictly required when onlyautosomes = TRUE.

When plotting sex chromosomes, make sure to specify sgc = c() when plotting the copy number profile of a male individual.

**Author(s)**

Jos B. Poell

**See Also**

[objectsampletotemplate](#), [squaremodel](#), [singlemodel](#)

## Examples

```
## simulated data assuming each chromosome comprises 100 bins
s <- jitter(c(1, 1, 0.8, 1.2, rep(1, 5), 1.4, rep(1, 13)), amount = 0)
n <- c(100, 100, 40, 60, rep(100, 5), 100, rep(100, 13))
bin <- 1:2200
chr <- rep(1:22, each = 100)
start <- rep(0:99*1000000+1, 22)
end <- rep(1:100*1000000, 22)
copynumbers <- jitter(rep(s,n), amount = 0.05)
segments <- rep(s, n)
template <- data.frame(bin = bin, chr = chr, start = start, end = end,
  copynumbers = copynumbers, segments = segments)
model <- singlemodel(template)
bestfit <- model$minima[model$error==min(model$error)]
singleplot(template, cellularity = tail(bestfit, 1), title = "sim")

## using segmented data from a QDNAseq-object
data("copyNumbersSegmented")
singlemodel(copyNumbersSegmented, QDNAseqobjectsample = 1)
singleplot(copyNumbersSegmented, QDNAseqobjectsample = 1,
  cellularity = 0.79)
## QDNAseq 'blacklists' sex chromosomes, but singleplot can plot them
singleplot(copyNumbersSegmented, QDNAseqobjectsample = 1,
  cellularity = 0.79, chrsubset = 12:24, onlyautosomes = FALSE)
```

---

squaremodel

*Calculate potential fits for a single sample using ploidy as a variable*

---

## Description

squaremodel performs a "two-dimensional" fitting algorithm on a single sample. It calculates the error of the fit at each cellularity over a range of "ploidies". Input can be either a template or a QDNAseq-object with the index of the sample specified. Returns a list with input parameters (method, penalty, and penploidy) and model characteristics (an error matrix, a logical matrix specifying minima, a data frame with all information, a data frame with only minima, and a graphical representation of the error matrix).

## Usage

```
squaremodel(template, QDNAseqobjectsample = FALSE, rows=100,
  ptop=5, pbottom=1, method = 'RMSE', exclude = c("X", "Y"),
  sgc = c(), penalty = 0, penploidy = 0, cellularities = seq(5,100),
  highlightminima = TRUE, standard)
```

## Arguments

template      Object. Either a data frame as created by `objectsamplenetemplate`, or a QDNAseq-object

QDNaseqobjectsample	Integer. Specifies which sample to analyze from the QDNaseqobject. Required when using a QDNaseq-object as template. Default = FALSE
prows	Integer. Sets the resolution of the ploidy-axis. Determines how many decrements are used to get from ptop to pbottom (see below). Therefore, the actual number of rows is actually prows + 1. Default = 100
ptop	Numeric. Sets the highest ploidy at which to start testing fits. Default = 5
pbottom	Numeric. Sets the lowest ploidy to be tested. Default = 1
method	Character string specifying which error method to use. For more documentation, consult the vignette. Can be "RMSE", "SMRE", or "MAE". Default = "RMSE"
exclude	Integer or character vector. Specifies which chromosomes to exclude for model fitting. Default = c("X", "Y")
sgc	Integer or character vector. Specifies which chromosomes occur with only a single copy in the germline
penalty	Numeric. Penalizes fits at lower cellularities. Suggested values between 0 and 1. Default = 0 (no penalty)
penploidy	Numeric. Penalizes fits that diverge from 2N with the formula $(1 + \text{abs}(\text{ploidy} - 2))^{\text{penploidy}}$ . Default = 0
cellularities	Numeric vector. Specifies the cellularities (in percentage) to be tested
highlightminima	Logical. Minima are highlighted in the matrixplot by a black dot. Default = TRUE
standard	Numeric. Force the ploidy to represent this raw value. When omitted, the standard will be calculated from the data

## Details

Unlike other functionality of ACE, squaremodel does not use the "standard", but it fits all tested ploidies to "standard = 1". It is therefore necessary that segment values are normalized to 1 (which they are by default coming from QDNaseq). The penalty parameter is the same as in singlemodel. Additionally, it is possible to penalize fits at ploidies diverging from 2N using the penploidy parameter. For other details on the fitting algorithm, see [singlemodel](#). Range of ploidies is set by parameters ptop and pbottom, and resolution is determined by prows. Resolution on the X-axis can be adapted by changing the cellularities option. To create good contrast in the matrixplot, the color scale derives from the inverse of the error, and the opacity of the dots marking the minima is calculated as  $\text{min}(\text{error})/\text{error}$ .

## Value

Returns a list, containing

method	Applied error method
penalty	Applied penalty factor for low cellularities
penploidy	Applied penalty factor for diverging ploidies
errormatrix	Numeric matrix with errors of all combinations of ploidy and cellularity

<code>minimatrix</code>	Logical matrix indicating whether the combination of ploidy and cellularity represents a minimum
<code>errordf</code>	Data frame with columns ploidy, cellularity, error, and minimum
<code>minimadf</code>	Same as <code>errordf</code> , but only containing minima and sorted by error
<code>matrixplot</code>	<code>ggplot2</code> -graph of the relative errors calculated at each combination of ploidy and cellularity

**Note**

`squaremodel()` only needs a data frame with columns named `chr` and `segments`. Every row should contain an individual genomic feature, i.e. a bin or a probe. If you have data with each row representing a segment, and the size of the segment given in a column (e.g. `NumBins` or `NumProbes`), you can create the data frame as follows (giving the correct variable names of course):

```
template <- data.frame(chr = rep(Chromosome, NumProbes), segments = rep(SegmentMean, NumProbes))
```

Alternatively you can look into [segmentstotemplate](#).

If your data contains sex chromosomes and you wish to include these for model fitting, then make sure to specify `exclude = c()`, and `sgc = c("X", "Y")` when analyzing data from a male individual.

**Author(s)**

Jos B. Poell

**See Also**

[objectsampletotemplate](#), [squaremodel](#), [singleplot](#)

**Examples**

```
## toy data assuming each chromosome comprises 100 bins
s <- jitter(c(1, 1, 0.8, 1.2, rep(1, 5), 1.4, rep(1, 13)), amount = 0)
n <- c(100, 100, 40, 60, rep(100, 5), 100, rep(100, 13))
df <- data.frame(chr = rep(1:22, each = 100), segments = rep(s, n))
squaremodel(df)$matrixplot
sm <- squaremodel(df, method = 'MAE', penalty = 0.5, penploidy = 0.5)
sm$matrixplot
mdf <- sm$minimadf
head(mdf[order(mdf$error, -mdf$cellularity),])

## using segmented data from a QDNaseq-object
data("copyNumbersSegmented")
sqm <- squaremodel(copyNumbersSegmented, QDNaseqobjectsample = 2,
  penalty = 0.5, penploidy = 0.5,
  ptop = 4.3, pbottom = 1.8, prows = 250)
sqm$matrixplot
mdf <- sqm$minimadf
head(mdf[order(mdf$error, -mdf$cellularity),])
```



---

squaremodelsummary      *Create a graphical summary of the result of squaremodel fitting*

---

### Description

squaremodelsummary creates a graphical summary of a squaremodel by plotting the matrixplot and absolute copy number profiles corresponding with the 7 best fits. The list of plots can be saved to a variable and/or a file.

### Usage

```
squaremodelsummary(template, QDNAseqobjectsample = FALSE,
  squaremodel, samplename, printplots = TRUE, outputdir,
  imagetype = 'pdf', trncname = FALSE)
```

### Arguments

template	Object. Either a data frame as created by <a href="#">objectsampletotemplate</a> , or a QDNAseq-object
QDNAseqobjectsample	Integer. Specifies which sample to analyze from the QDNAseqobject. Required when using a QDNAseq-object as template. Default = FALSE
squaremodel	List of objects returned by <a href="#">squaremodel</a>
samplename	Character string. Use this sample name in the title of the matrixplot. If the sample comes from a QDNAseq-object and this argument is omitted, the sample name is taken from the QDNAseq-object
printplots	Logical. Print the plots to file. Default = TRUE
outputdir	Character string. Print the plots to this directory
imagetype	Character string. Plots are printed to file using this graphics device. "pdf" will result in 8 pages with individual plots, while the other devices print a 2x4 mosaic. Default = "pdf"
trncname	Logical. If set to TRUE, trncname truncates the sample name from the first instance of "_" in the name. You can also specify the regular expression here, e.g. trncname = "-.*" truncates the name from the first dash. Default = FALSE

### Value

Returns a list with eight plots (ggplot2-objects): the matrixplot and the copy number plots corresponding to the seven best fits. If printplots is TRUE, the plots will be printed to file.

### Author(s)

Jos B. Poell

**See Also**[squaremodel](#)**Examples**

```
## simulated data assuming each chromosome comprises 100 bins
s <- jitter(c(1, 1, 0.8, 1.2, rep(1, 5), 1.4, rep(1, 13)), amount = 0)
n <- c(100, 100, 40, 60, rep(100, 5), 100, rep(100, 13))
bin <- 1:2200
chr <- rep(1:22, each = 100)
start <- rep(0:99*1000000+1, 22)
end <- rep(1:100*1000000, 22)
copynumbers <- jitter(rep(s,n), amount = 0.05)
segments <- rep(s, n)
template <- data.frame(bin = bin, chr = chr, start = start, end = end,
  copynumbers = copynumbers, segments = segments)
sm <- squaremodel(template, method = 'MAE', penalty = 0.5,
  penploidy = 0.5)
sms <- squaremodelsummary(template, squaremodel = sm,
  samplename = "sim", printplots = FALSE)
sms[[1]]
sms[[2]]
## using segmented data from a QDNAseq-object
data("copyNumbersSegmented")
sqm <- squaremodel(copyNumbersSegmented, QDNAseqobjectsample = 2,
  penalty = 0.5, penploidy = 0.5,
  ptop = 4.3, pbottom = 1.8, prow = 250)
sqms <- squaremodelsummary(copyNumbersSegmented, 2,
  squaremodel = sqm, printplots = FALSE)
sqms[[1]]
sqms[[2]] + ggplot2::ggtitle("Top fit for sample2")
```

---

templatefromequalsegments

*Create a template data frame with artificial segments of equal length*

---

**Description**

Divide chromosomes in artificial segments with the specified number of bins. You can provide a template without segment values (for instance, an unsegmented QDNAseq-object) as input for this function. You can also use this function to only resegment a subset of chromosomes.

**Usage**

```
templatefromequalsegments(template, QDNAseqobjectsample = FALSE,
  equalsegments = 20, funtype = 'mean', chrsubset,
  onlyautosomes = TRUE)
```

**Arguments**

template	Object. Either a data frame as created by <a href="#">objectsampletotemplate</a> , or a QDNAseq-object
QDNAseqobjectsample	Integer. Specifies which sample to analyze from the QDNAseqobject. Required when using a QDNAseq-object as template. Default = FALSE
equalsegments	Integer. <code>templatefromequalsegments</code> "resegments" simply by creating segments containing as many bins as specified in this argument. Default = 20
funtype	Character string. Specifies the function used to calculate new segment values. Alternative is "median". Default = "mean"
chrsubset	Integer vector. Specify the chromosomes you want to resegment
onlyautosomes	Logical or integer. You can fill in an integer to specify how many autosomes your species has. When TRUE, <code>templatefromequalsegments</code> defaults to 22 (human) autosomes. When FALSE, <code>templatefromequalsegments</code> will also resegment whichever other chromosomes are specified in the template, e.g. "X", "Y", "MT"

**Details**

`templatefromequalsegments` resegments the input template chromosome by chromosome. It makes segments with the number of bins specified in the argument `equalsegments`. Any leftover bins are divided equally among all newly created segments. Only bins with values (!is.na) are considered. If there are fewer bins on a chromosome than two times the specified value, then all bins of this chromosome are put in a single segment.

**Value**

Returns a template data frame

**Note**

The `chrsubset` argument works slightly differently compared to the plotting functions. It does not necessarily segment only contiguous chromosomes. For instance, specifying `chrsubset = c(8,12)` will only resegment chromosomes 8 and 12. For the other chromosomes, segment values will not be changed.

**Author(s)**

Jos B. Poell

**See Also**

[twosamplecompare](#), [correlationmatrixadjusted](#)

**Examples**

```
data("copyNumbersSegmented")
tfes <- templatefromequalsegments(copyNumbersSegmented,
  QDNAseqobjectsample = 2)
```

---

twosamplecompare	<i>Overlay copy number data of two samples and compare segment values</i>
------------------	---

---

### Description

twosamplecompare "resegments" two samples to have the same breakpoints. Both samples' means of the resulting segments are tested for equality using the two-sided Welch two sample t-test. twosamplecompare returns a data frame with the comparisons per segment, it returns the correlation of segments, and a copy number plot with an overlay of (scaled) segment values of both samples and the associated  $-\log_{10}$ -transformed q-values.

### Usage

```
twosamplecompare(template1, index1 = FALSE, ploidy1 = 2,
  cellularity1 = 1, standard1, name1, template2, index2 = FALSE,
  ploidy2 = 2, cellularity2 = 1, standard2, name2,
  equalsegments = FALSE, althmethod = FALSE, cap = 12, qcap = 12,
  bottom = 0, plot = TRUE, trncname = FALSE, legend = TRUE,
  chrsubset, onlyautosomes = TRUE, sgc = c(),
  showcorrelation = TRUE)
```

### Arguments

template1	Object. Either a data frame as created by <a href="#">objectsampletotemplate</a> , or a QDNAseq-object
index1	Integer. If template1 is a QDNAseqobject, this specifies the index of the first sample. Default = FALSE
ploidy1	Integer. Assume the median of segments of the first sample has this absolute copy number. Default = 2
cellularity1	Numeric. Used for rescaling bin and segment values of the first sample. Default = 1
standard1	Numeric. Forces ploidy1 to represent this raw value. When omitted, the standard will be calculated from the data. When using parameters obtained from squaremodel, specify standard1 = 1
name1	Character string. Name of the first sample. Printed on graph
template2	Object. Either a data frame as created by <a href="#">objectsampletotemplate</a> , or a QDNAseq-object. When omitted, template1 will be used
index2	Integer. Specifies the index of the second sample in template2 or, when template2 is omitted, in template1. Default = FALSE
ploidy2	Integer. Assume the median of segments of the second sample has this absolute copy number. Default = 2
cellularity2	Numeric. Used for rescaling bin and segment values of the second sample. Default = 1

standard2	Numeric. Forces ploidy2 to represent this raw value. When omitted, the standard will be calculated from the data. When using parameters obtained from squaremodel, specify standard2 = 1
name2	Character string. Name of the second sample. Printed on graph
equalsegments	Logical or integer. If TRUE, twosamplecompare "resegments" both samples simply by creating segments containing roughly 20 bins, or as many bins as specified in this argument. When FALSE, both samples are resegmented by combining the break points and applying them to both samples. Default = FALSE
altmethod	Logical or character string. Instead of scaling the sample segments to absolute copies, scale them to standard units. There are two options: "SD" and "MAD". In the first case, the mean of segments is set to 0 and for each segment the distance (in standard deviations or "SD units" from the segment mean to the mean of segments is calculated in standard deviations. In case of "MAD", instead the median of segments, segment median, and median absolute deviation is used. Adjust the y-axis with the cap and bottom arguments for better visualization. Default = FALSE
cap	Integer. Influences your output copy number graphs. The upper limit of the y-axis is set at this number. When set to "max", it sets the cap to the maximum absolute copynumber value, rounded up. Bins and segments that exceed the cap are represented by a special mark. Recommended use between 8 and 16. Default = 12
qcap	Integer. Sets the upper limit of the secondary y-axis. Default = 12
bottom	Integer. Similar to cap, but for the lower limit of the y-axis. When set to "min", it sets the bottom to the minimum absolute copynumber value, rounded down. Bins and segments that subceed the bottom are represented by a special mark. Default = 0
plot	Logical. Produce a two-sample copy number plot. Default = TRUE
trncname	Logical. In case of a QDNAseq object, the name of the sample is retrieved from the object and used as title. If set to TRUE, trncname truncates the sample name from the first instance of "_" in the name. You can also specify the regular expression here, e.g. trncname = "-.*" truncates the name from the first dash. Default = FALSE
legend	Logical. Add the legend to the two-sample plot. Default = TRUE
chrsubset	Integer vector. Specify the chromosomes you want to plot. It will always take the full range of chromosomes in your subset, so specifying chrsubset = c(4, 8) will give the same plot as chrsubset = 4:8. When using a subset, twosamplecompare will not plot the cellularity and error on the plot.
onlyautosomes	Logical or integer. You can fill in an integer to specify how many autosomes your species has. When TRUE, twosamplecompare defaults to 22 (human) autosomes. When FALSE, twosamplecompare will also plot whichever other chromosomes are specified in the template, e.g. "X", "Y", "MT"
sgc	Integer or character vector. Specify which chromosomes occur with only a single copy in the germline. Note that this is assumed for both samples.
showcorrelation	Logical. Add the correlation to the two-sample plot. Default = TRUE

## Details

This function can be used for different types of comparisons. It can be used to compare a tumor sample with a healthy (preferably matched) control. In this case, it may not be necessary to fill in the cellularity, because it will not make a difference for the statistical tests. In this ability the function helps to determine which (if any) segments are significantly different from normal. The other major use is to compare two tumors from potentially the same origin, but that were separated in space or time. You can then assess if changes have occurred, or even whether the two samples are from different clonal origin. In this case it is important to achieve maximum similarity in segments. Now the argument `altmethod` may come in handy, because it does not require model fitting and optimization. The q-values that are obtained with this function should be interpreted with caution. The two-sample statistical tests will easily reach significance when the sample sizes, in this case bins per segment, are large. By creating equal segment sizes with the argument `equalsegments`, these biases disappear.

## Value

- `twosampled` - data frame with the newly created segments and the information and comparison of both samples
- `correlation` - Pearson correlation of the segment values of all bins between both samples
- `subsetcorrelation` - same as `correlation`, but only applying to subset of chromosomes specified by the argument `chrsubset`
- `compareplot` - `ggplot2`-graph of both samples with segment values in red (first sample) and blue (second sample). Green bars indicate q-values of the segments, scaled on the secondary axis

## Note

The data frame, plot, and `subsetcorrelation` all use the same selection of chromosomes. The correlation in the plot corresponds to the displayed chromosomes. Note that the returned value `correlation` uses all segments in the data, also from the sex chromosomes when available. However, if there is no useful data for an entire chromosome, it will not constitute a segment and thus be excluded from the data frame, even though the chromosome may be included in the plot.

If you want to get rid of the green significance bars in the plot, you can set `qcap = 0`. If you insist on getting rid of the entire secondary axis, save the plot to an object, then run: `plotobject + scale_y_continuous(name = "copies", sec.axis = sec_axis(~., labels = NULL), expand = c(0,0))`

## Author(s)

Jos B. Poell

## See Also

[templatefromequalsegments](#)

## Examples

```
## simulated data assuming each chromosome comprises 100 bins
s1 <- jitter(c(1, 1, 0.8, 1.2, rep(1, 5), 1.4, rep(1, 13)), amount = 0)
```

```
s2 <- jitter(c(1, 1, 1.25, rep(1, 5), 1.5, rep(1, 13)), amount = 0)
n1 <- c(100, 100, 40, 60, rep(100, 5), 100, rep(100, 13))
n2 <- c(rep(100, 22))
bin <- 1:2200
chr <- rep(1:22, each = 100)
start <- rep(0:99*1000000+1, 22)
end <- rep(1:100*1000000, 22)
copynumbers1 <- jitter(rep(s1,n1), amount = 0.05)
copynumbers2 <- jitter(rep(s2,n2), amount = 0.05)
segments1 <- rep(s1, n1)
segments2 <- rep(s2, n2)
template1 <- data.frame(bin = bin, chr = chr, start = start, end = end,
  copynumbers = copynumbers1, segments = segments1)
template2 <- data.frame(bin = bin, chr = chr, start = start, end = end,
  copynumbers = copynumbers2, segments = segments2)
twosamplecompare(template1 = template1, template2 = template2,
  cellularity1 = 0.4, cellularity2 = 0.5)
twosamplecompare(template1 = template1, template2 = template2,
  cellularity1 = 0.4, cellularity2 = 0.5, equalsegments = 20)
## using segmented data from a QDNAseq-object
data("copyNumbersSegmented")
## for derivations of the parameters for this fit, see squaremodel
twosamplecompare(copyNumbersSegmented, index1 = 1, cellularity1 = 0.4,
  standard1 = 1, index2 = 2, cellularity2 = 0.41, ploidy2 = 2.08,
  standard2 = 1)
```

# Index

## \* datasets

- copyNumbersSegmented, 9
- ACE (ACE-package), 2
- ACE-package, 2
- ACEcall, 3, 3, 25
- analyzegenomiclocations, 3, 6, 13, 15
- compresstemplate, 7, 25, 26
- copyNumbersSegmented, 9
- correlationmatrix, 9
- correlationmatrixadjusted, 35
- correlationmatrixadjusted  
(correlationmatrix), 9
- corrplot, 10
- forcesegmentsontemplate, 3, 11
- getadjustedsegments, 3, 7, 11, 12, 12, 15, 22
- ggplot2, 2
- linkvariants, 3, 7, 14, 22
- loopsquaremodel, 3, 16
- objectsampletotemplate, 4, 8, 11, 12, 18,  
26–30, 32, 33, 35, 36
- ploidyplootloop, 17
- ploidyplootloop (runACE), 22
- postanalysisloop, 3, 13–15, 19
- QDNAseqCopyNumbers, 2
- runACE, 2, 19, 22, 22
- segmentstotemplate, 8, 24, 27, 32
- singlemodel, 2, 18, 24, 25, 26, 29, 31
- singleplot, 2, 3, 5, 18, 24, 25, 27, 28, 32
- squaremodel, 3, 16–18, 24, 25, 27, 29, 30,  
32–34
- squaremodelsummary, 3, 17, 33
- templatefromequalsegments, 9, 10, 34, 38
- twosamplecompare, 3, 9, 10, 12, 25, 35, 36