Package 'diaQTL'

January 13, 2021

Title QTL Analysis in Diallel Populations

Version 0.91

Author Jeffrey B. Endelman and Rodrigo R. Amadeu

Maintainer Jeffrey Endelman <endelman@wisc.edu>

Description QTL analysis of diploid and autotetraploid diallel populations. Phenotypes are regressed on genotype probabilities, and the regression coefficients are random effects.

Depends R (>= 3.5.0)

License GPL-3

LazyData true

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Encoding UTF-8

Imports BGLR, ggplot2, methods, coda, Matrix, scam, parallel, arrangements, tidyr, ggfittext, ggdendro, labeling

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

BayesCI 2
diallel_geno-class 3
diallel_geno_pheno-class 3
diallel_geno_pheno_G-class 4
diplo_freq 4
diplo_get 5
F1codes 6
fine_map 6
fitQTL 7
Gprep 8
haplo_cluster 9
haplo_freq 10
haplo_get 11
haplo_plot 11
IBDmat 12
LODthresh 13

phased_parents 14
read_data 14
read_polyancestry 16
S1codes 16
scan1 17
scan1_permute 18
scan1_summary 19
set_params 20

Index **22**

`BayesCI` *Bayesian Credible Interval for QTL position*

Description

Bayesian Credible Interval for QTL position

Usage

```
BayesCI(scan1_data, data, chrom, CI.prob = 0.9)
```

Arguments

`scan1_data`	data frame output from scan1
`data`	variable of class `diallel_geno_pheno`
`chrom`	chromosome
`CI.prob`	probability for the credible interval

Details

Parameter `CI.prob` sets the probability for the Bayesian credible interval (e.g., 0.90, 0.95) using the likelihood (10^LOD) distribution.

Value

subset of scan1_data with markers in the CI

Examples

```
## Not run:
  BayesCI(scan1_example,data,chrom="10",CI.prob=0.9)

## End(Not run)
```

diallel_geno-class *S4 class with genotype data*

Description

S4 class with genotype data

Slots

`ploidy` Either 2 or 4

`dominance` Integer 1-4 indicating 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance.

`X.GCA` Incidence matrix for GCA effects

`map` data frame with marker,chrom, position (cM and/or bp) and bin

`geno` list of length equal to the number of marker bins. Each element is a list of length `dominance`. The elements in the nested list are sparse matrices with dimensions (id x effects).

diallel_geno_pheno-class *S4 class with genotype and phenotype data*

Description

S4 class with genotype and phenotype data

Slots

`ploidy` Either 2 or 4

`dominance` Integer 1-4 indicating 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance.

`X.GCA` Incidence matrix for GCA effects

`map` data frame with marker,chrom, position (cM and/or bp) and bin

`geno` list of length equal to the number of marker bins. Each element is a list of length ploidy corresponding to additive, digenic, trigenic, and quadrigenic effects. The elements in the nested list are sparse matrices with dimensions (id x effects).

`pheno` data frame of phenotypes

`X` incidence matrix for fixed effects

`Z` incidence matrix for individuals

diallel_geno_pheno_G-class

S4 class with genotype and phenotype data and the additive relationship matrix

Description

S4 class with genotype and phenotype data and the additive relationship matrix

Slots

`ploidy` Either 2 or 4

`dominance` Integer 1-4 indicating 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance.

`X.GCA` Incidence matrix for GCA effects

`map` data frame with marker,chrom, position (cM and/or bp) and bin

`geno` list of length equal to the number of marker bins. Each element is a list of length ploidy corresponding to additive, digenic, trigenic, and quadrigenic effects. The elements in the nested list are sparse matrices with dimensions (id x effects).

`pheno` data frame of phenotypes

`X` incidence matrix for fixed effects

`Z` incidence matrix for individuals

`G1` additive relationship matrix computed by `IBDmat`

diplo_freq *Diplotype frequencies*

Description

Plot the frequency of individuals with diplotype dosage above a threshold

Usage

```
diplo_freq(data, diplotypes, dosage, position, chrom = NULL)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`diplotypes`	Names of diplotypes
`dosage`	Dosage threshold
`position`	Either "cM" or "bp" for plotting
`chrom`	Names of chromosomes (default is all)

Details

Useful for visualizing selection in selfed populations.

Value

List containing

result Data frame with the map and frequency

plot ggplot object

`diplo_get` *Dosage of parental diplotypes*

Description

Dosage of parental diplotypes

Usage

```
diplo_get(data, marker = NULL, id = NULL)
```

Arguments

`data`	Variable inheriting from class `diallel_geno`
`marker`	Name of marker
`id`	Name of individual

Details

Function can be used to get parental diplotype dosage estimates at a single marker for all individuals (in which case `id` should be `NULL`) or for a single individual for all markers (in which case `marker` should be `NULL`)

Value

Matrix of (id or markers) x parental diplotypes

Examples

```
## Not run:
  diplo_example = diplo_get(data = diallel_example,
                            marker = "solcap_snp_c2_25522")
  diplo_example = diplo_get(data = diallel_example,
                            id = "W15263-8R")

## End(Not run)
```

F1codes *Genotype codes for F1 populations*

Description

Character vector with the 100 possible tetraploid genotypes for a F1 population. Maternal haplotypes are denoted 1,2,3,4 and paternal haplotypes 5,6,7,8.

Usage

```
data(F1codes)
```

Format

character vector

fine_map *Visualize haplotype switches for fine mapping*

Description

Visualize haplotype switches for fine mapping

Usage

```
fine_map(data, haplotype, interval, trait = NULL)
```

Arguments

data	Variable inheriting from class diallel_geno
haplotype	Name of parental haplotype
interval	2-vector with marker names
trait	Name of trait to plot (optional)

Details

Function returns graphic for all individuals with a haplotype switch (defined as change in dosage from 0 to ≥ 1 or vice versa) for haplotype within interval. If trait is included, the trait values for each individual are displayed on the right side. The function requires map positions in bp to be included in data.

Value

ggplot2 variable

`fitQTL` *Fit a single QTL model*

Description

Fit a single QTL model

Usage

```
fitQTL(
  data,
  trait,
  marker,
  params,
  dominance = 1,
  CI.prob = 0.9,
  polygenic = TRUE,
  cofactor = NULL
)
```

Arguments

`data`	Variable of class `diallel_geno_pheno`
`trait`	Name of trait
`marker`	Name of marker to fit as QTL
`params`	List containing the number of burn-in (burnIn) and total iterations (nIter)
`dominance`	Dominance degree
`CI.prob`	Probability for Bayesian credible interval
`polygenic`	TRUE/FALSE whether to include polygenic background effect
`cofactor`	Name of marker to fit as cofactor (optional)

Details

LOD score is the difference between the log10-likelihood for the QTL model vs. no QTL model (higher is better). deltaDIC is the difference between the Deviance Information Criterion for the QTL model vs. no QTL model (lower values is better). r2 is the squared correlation between the fitted and observed values. Parameter `dominance` controls the genetic model: 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance. MCMC `params` can be estimated using `set_params`. Parameter `CI.prob` sets the probability (e.g., 0.90, 0.95) for the Bayesian credible interval for the estimated effects (to disable plotting of the CI, use `CI.prob=NULL`). The returned list `effects` contains the additive (and when included) digenic dominance effects. The proportion of variance for each effect is returned in `var`. The returned object `plots$dom` shows the digenic dominance effects above the diagonal, and below the diagonal is the sum of the additive and digenic dominance effects. The polygenic background effect has covariance equal to the additive relationship computed by `IBDmat`, leaving out the chromosome with the QTL. For faster execution with the polygenic model, use `Gprep` first.

Value

List containing

r2 sauared correlation betwen fitted and observed values

deltaDIC Deviance Information Criterion relative to null model

resid Residuals

var Matrix with proportion of variance for the effects

effects List of matrices containing the additive and higher order effects

plots List of ggplot objects for the effects

Examples

```
## Not run:
## additive effects
params1 <- set_params( diallel_example, trait = "tuber_shape" ,q=0.05,r=0.05)

fit1 <- fitQTL( data = diallel_example,
                trait = "tuber_shape",
                params = params1,
                marker = "solcap_snp_c2_25522",
                CI.prob = 0.9)

## additive + dominance effects
params2 <- set_params( diallel_example, trait = "tuber_shape", dominance=2,q=0.05,r=0.05)

fit2 <- fitQTL( data = diallel_example,
                trait = "tuber_shape",
                params = params2,
                marker = "solcap_snp_c2_25522",
                dominance = 2,
                CI.prob=0.9)


## End(Not run)
```

`Gprep` *Compute realized relationship matrix for fitQTL*

Description

Compute realized relationship matrix for fitQTL

Usage

```
Gprep(data, marker)
```

Arguments

`data`	variable inheriting from class `diallel_geno_pheno`
`marker`	marker that will be used for `fitQTL`

Details

Computes the additive realized relationship matrix with IBDmat so that fitQTL does not need to and can run more quickly. The chromosome containing marker is excluded from the relationship calculation.

Value

variable inheriting from class diallel_geno_pheno_G

Examples

```
## Not run:
  diallel_example <- Gprep(data = diallel_example, marker = "solcap_snp_c2_25522")

## End(Not run)
```

---

haplo_cluster *Cluster parental haplotypes*

Description

Cluster parental haplotypes

Usage

```
haplo_cluster(filename, marker, haplotypes = NULL)
```

Arguments

filename	Name of CSV input file
marker	Either target marker or marker interval (see Details).
haplotypes	Vector of haplotype names (default is all)

Details

The input file (diaQTL_parents.csv) should be generated by read_polyancestry. The argument marker can be either a single marker or vector of two markers. If a single marker, the function finds the smallest interval containing that marker such that the phased SNP haplotypes are all unique. If two markers are provided, that interval is used. Clustering utilizes hclust(method="average"). See also phased_parents for an additional visualization tool.

Value

List containing

haplo Data frame of haplotypes

dendro Dendrogram

haplo_freq *Haplotype frequencies*

Description

Plots the frequency of individuals with haplotype dosage above a threshold

Usage

```
haplo_freq(
  data,
  haplotypes,
  dosage,
  id = NULL,
  position = "cM",
  chrom = NULL,
  markers = NULL
)
```

Arguments

data	Variable inheriting from class diallel_geno
haplotypes	Names of haplotypes
dosage	Dosage threshold
id	Vector of id names (default is entire population)
position	Either "cM" (default) or "bp" for plotting
chrom	Names of chromosomes (default is all)
markers	Optional, markers to indicate with dashed line. Only available when plotting a single chromosome.

Details

Useful for visualizing selection in selfed populations. For multiple chromosomes, each haplotype is shown in its own panel using facet_wrap. For one chromosome, the haplotypes are shown on the same set of axes.

Value

List containing

result Data frame with the map and frequency

plot ggplot object

haplo_get *Dosage of parental haplotypes*

Description

Dosage of parental haplotypes

Usage

```
haplo_get(data, marker = NULL, id = NULL)
```

Arguments

data	Variable inheriting from class diallel_geno
marker	Name of marker
id	Name of individual

Details

Function can be used to get parental haplotype dosage estimates at a single marker for all individuals (in which case id should be NULL) or for a single individual for all markers (in which case marker should be NULL)

Value

Matrix of (id or markers) x parental haplotypes

Examples

```
## Not run:
  haplo_example = haplo_get(data = diallel_example,
                            marker = "solcap_snp_c2_25522")
  haplo_example = haplo_get(data = diallel_example,
                            id = "W15263-8R")

## End(Not run)
```

haplo_plot *Plot parental haplotype dosage*

Description

Plot parental haplotype dosages across the chromosome for one individual

Usage

```
haplo_plot(data, id, chrom, position = "cM", markers = NULL)
```

Arguments

data	Variable inheriting from class diallel_geno
id	Name of individual
chrom	Name of chromosome
position	Either "cM" (default) or "bp"
markers	Optional, markers to indicate with dashed line

Details

For "cM" plotting, only one marker per bin is displayed. For "bp" plotting, all markers are included.

Value

ggplot object

Examples

```
## Not run:
haplo_plot(data = diallel_example,
           id = "W15263-8R",
           chrom = 10)

haplo_plot(data = diallel_example,
           id = "W15263-8R",
           chrom = 10,
           marker = "solcap_snp_c2_25522")

## End(Not run)
```

IBDmat *Realized IBD relationship*

Description

Calculates realized relationship matrices (additive and dominance) from founder genotype probabilities

Usage

```
IBDmat(data, dominance = 1, chrom = NULL)
```

Arguments

data	Variable inheriting from class diallel_geno
dominance	One of 1,2,3,4
chrom	Optional, vector of chromosome names to include

Details

Parameter dominance refers to 1 = additive, 2 = digenic, 3 = trigenic, 4 = quadrigenic (Gallais 2003). Can specify to use only a subset of the chromosomes (by default, all chromosomes are used). Calculated based on the marker bins.

Value

Relationship matrix

References

Gallais, A. 2003. Quantitative Genetics and Breeding Methods in Autopolyploid Plants. Institut National de la Recherche Agronomique, Paris.

Examples

```
## Not run:
  IBD_example = IBDmat(data = diallel_example, dominance=1) #additive
  IBD_example = IBDmat(data = diallel_example, dominance=2) #digenic dominance

## End(Not run)
```

LODthresh *LOD thresholds for scan1*

Description

LOD thresholds for scan1

Usage

```
LODthresh(genome.size, num.parents, ploidy, dominance = 1)
```

Arguments

genome.size	Genome size in Morgans (not centiMorgans)
num.parents	Number of parents
ploidy	2 or 4
dominance	1 (additive) or 2 (digenic dominance)

Details

LOD thresholds to control the genome-wide false positive rate at 0.05 were determined via simulation for up to 20 parents and genome sizes up to 12 Morgans. A monotone increasing concave curve was fit to these results using R package scam and is used for prediction. (The LOD threshold does not depend on population size.)

Value

LOD threshold

phased_parents *Visualize phased SNPs of parents*

Description

Visualize phased SNPs of parents

Usage

```
phased_parents(filename, interval, markers, parents)
```

Arguments

filename	Name of CSV input file
interval	Vector of length 2 with the first and last marker names
markers	Vector of marker names to plot
parents	Vector of parent names to plot

Details

The input file can be generated by read_polyancestry. The solid circles in the figure represent the allele counted by dosage.

Value

ggplot2 object

read_data *Read data files*

Description

Reads genotype, pedigree, and phenotype data files

Usage

```
read_data(
  genofile,
  ploidy = 4,
  pedfile,
  phenofile = NULL,
  fixed = NULL,
  bin.markers = TRUE,
  dominance = 2,
  n.core = 1
)
```

Arguments

genofile	File with map and genotype probabilities
ploidy	Either 2 or 4
pedfile	File with pedigree data (id,parent1,parent2)
phenofile	File with phenotype data (optional)
fixed	If there are fixed effects, this is a character vector of "factor" or "numeric"
bin.markers	TRUE/FALSE whether to bin markers with the same cM position
dominance	Maximum value of dominance that will be used for analysis (1-4). See Details.
n.core	Number of cores for parallel execution

Details

Genotype and pedigree input files can be created from PolyOrigin output using read_polyancestry. The first 3 columns of the genotype file should be the genetic map (labeled marker, chrom, cM), and a fourth column for a reference genome position (labeled bp) can also be included. The map is followed by the members of the population. The genotype data for each marker x individual combination is a string with the format "state|state|state...=>prob|prob|prob...", where "state" refers to the genotype state and "prob" is the genotype probability in decimal format. Only states with nonzero probabilities need to be listed. The encoding for the states in tetraploids is described in the documentation for the F1codes and S1codes datasets that come with the package. For diploids, there are 4 F1 genotype codes, 1,2,3,4, which correspond to haplotype combinations 1-3,1-4,2-3,2-4, respectively; the S1 genotype codes 1,2,3 correspond to 1-1,1-2,2-2, respectively. For the phenotype file, first column is id, followed by traits, and then any fixed effects. Pass a character vector for the function argument "fixed" to specify whether each effect is a factor or numeric covariate. The number of traits is deduced based on the number of columns. Binary traits must be coded N/Y and are converted to 0/1 internally for analysis by probit regression. Missing data in the phenotype file should be coded as NA. The parameter dominance specifies the maximum value of dominance that can be used in subsequent analysis: 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance. For maximum flexibility, use dominance = 4, but more memory is required. This will allow you to use any value of dominance (from 1 to 4) in functions such as scan1 and fitQTL. Output files from the BGLR package are stored in a folder named 'tmp' in the current directory.

Value

Variable of class diallel_geno if phenofile is NULL, otherwise diallel_geno_pheno

Examples

```
## Not run:
  ## Get the location of raw csv files examples
  genocsv = system.file( "vignette_data", "potato_geno.csv", package = "diaQTL" )
  pedcsv = system.file( "vignette_data", "potato_ped.csv", package = "diaQTL" )
  phenocsv = system.file( "vignette_data", "potato_pheno.csv", package = "diaQTL" )

  ## Check their location in the system
  print(genocsv)
  print(pedcsv)
  print(phenocsv)

  ## Load them in R
  diallel_example <- read_data(genofile = genocsv,
```

```
                                ploidy = 4,
                                pedfile = pedcsv,
                                phenofile = phenocsv)

## End(Not run)
```

`read_polyancestry` *Create diaQTL input files from polyancestry file*

Description

Create diaQTL input files from polyancestry file

Usage

```
read_polyancestry(filename, mapfile = NULL, remove.outliers = TRUE)
```

Arguments

`filename`	Name of polyancestry file
`mapfile`	Optional name of CSV file containing the physical map (marker, chrom, bp)
`remove.outliers`	Should offspring flagged as outliers be removed (default is TRUE)

Details

Creates the pedigree (diaQTL_pedfile.csv) and genotype (diaQTL_genofile.csv) input files needed for `read_data` from the polyancestry output file generated by the PolyOrigin software. PolyOrigin outputs a genetic map in cM. To add a physical map in bp, use the option `mapfile`. The input file needed for `phased_parents` (diaQTL_parents.csv) is also created.

`S1codes` *Genotype codes for S1 populations*

Description

Character vector with the 35 possible tetraploid genotypes for a S1 population. Haplotypes are denoted 1,2,3,4.

Usage

```
data(S1codes)
```

Format

character vector

scan1 *Single QTL scan*

Description

Performs a linear regression for each position in the map.

Usage

```
scan1(
  data,
  trait,
  params,
  dominance = 1,
  chrom = NULL,
  cofactor = NULL,
  n.core = 1
)
```

Arguments

data	Variable of class diallel_geno_pheno
trait	Name of trait
params	List containing burnIn and nIter
dominance	Dominance degree (1-4)
chrom	Names of chromosomes to scan (default is all)
cofactor	Optional name of marker to include as cofactor in the scan
n.core	Number of cores for parallel execution

Details

LOD score is the difference between the log10-likelihood for the QTL model vs. no QTL model (higher is better). deltaDIC is the difference between the Deviance Information Criterion for the QTL model vs. no QTL model (lower values is better). r2 is the squared correlation between the fitted and observed values. Parameter dominance controls the genetic model: 1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance. MCMC params can be estimated using set_params.

Value

Data frame containing the map, LOD, r2 and deltaDIC results.

Examples

```
## Not run:
  par1 <- set_params(data = diallel_example,
                     trait = "tuber_shape")

  scan1_example <- scan1(data = diallel_example,
                chrom = 10,
```

```
                trait = "tuber_shape",
                params = par1)

## End(Not run)
```

---

scan1_permute *Permutation test for scan1*

Description

Permutation test for scan1

Usage

```
scan1_permute(
  data,
  trait,
  params,
  n.permute = 1000,
  chrom = NULL,
  dominance = 1,
  cofactor = NULL,
  n.core = 1
)
```

Arguments

data	Variable of class diallel_geno_pheno
trait	Name of trait
params	List containing burnIn and nIter
n.permute	Number of permutations
chrom	Names of chromosomes to scan (default is all)
dominance	Dominance degree (1-4)
cofactor	Optional name of marker to include as cofactor in the scan
n.core	Number of cores for parallel execution

Value

Data frame with maximum LOD and minimum deltaDIC for each iteration

Examples

```
## Not run:
  par1 <- set_params(data = diallel_example,
                     trait = "tuber_shape")

  ans1_permut <- scan1_permute(data = diallel_example,
                               chrom = 10,
                               trait = "tuber_shape",
```

```
                                params = par1,
                                n.permute = 100)

## End(Not run)
```

---

scan1_summary *Summary of scan1 result*

Description

Summary of scan1 result

Usage

```
scan1_summary(scan1_data, thresh = NULL, chrom = NULL, position = "cM")
```

Arguments

scan1_data	output from scan1
thresh	optional, LOD threshold for plotting
chrom	optional, subset of chromosomes to plot
position	Either "cM" or "bp"

Value

List containing

peaks Data frame of the markers with the highest LOD score per chromosome

plot ggplot object

Examples

```
## Not run:
  scan1_summary( scan1_example )
  scan1_summary( scan1_example, chrom = "10" )
  scan1_summary( scan1_example, chrom = c( "10", "12" ) )

## End(Not run)
```

`set_params` *Determine burn-in and total number of iterations*

Description

Determine burn-in and total number of iterations

Usage

```
set_params(
  data,
  trait,
  dominance = 1,
  marker = NULL,
  q = 0.5,
  r = 0.1,
  nIter = 2000
)
```

Arguments

`data`	variable of class `diallel_geno_pheno`
`trait`	name of trait
`dominance`	dominance degree
`marker`	name of marker (optional)
`q`	quantile to estimate
`r`	tolerance for quantile
`nIter`	number of iterations

Details

Determines the burn-in and total number of iterations using the Raftery and Lewis diagnostic from R package `coda`, based on a 95% probability that the estimate for quantile `q` of the additive effects is within the interval `(q-r,q+r)`. The 90th percentile for burn-in and total iterations across the additive effects is returned. Parameter `dominance` specifies which genetic model (1 = additive, 2 = digenic dominance, 3 = trigenic dominance, 4 = quadrigenic dominance) to use when determining the number of iterations, but this must be parameter must still be specified when calling functions such as `scan1` or `fitQTL`. Suggested values for `scan1` are q=0.5 and r=0.1. For `fitQTL`, the values depend on the desired Bayesican credible interval. For a 90% CI, suggested values are q=0.05 and r=0.025. If `marker=NULL` (default), the first marker of every chromosome is analyzed to generate parameters suitable for `scan1`. Parameter `nIter` sets the number of iterations used to apply the Raftery and Lewis diagnostic; the default value is 2000, and if a larger number is needed, an error will be generated with this information.

Value

List containing

burnIn Number of burn-in iterations

nIter Total number of iterations

Examples

```
## Not run:
  # Parameters for scan1
  par1 <- set_params(data = diallel_example,
                     trait = "tuber_shape",q=0.5,r=0.1)

  # Parameters for fitQTL
  par2 <- set_params(data = diallel_example,
                     trait = "tuber_shape",q=0.05,r=0.05,marker="solcap_c2_25522")

## End(Not run)
```

Index

∗ **datasets**
F1codes, 6
S1codes, 16

BayesCI, 2

diallel_geno, *4–6*, *10–12*, *15*
diallel_geno *(*diallel_geno-class*)*, 3
diallel_geno-class, 3
diallel_geno_pheno, *2*, *7, 8*, *15*, *17, 18*, *20*
diallel_geno_pheno
*(*diallel_geno_pheno-class*)*, 3
diallel_geno_pheno-class, 3
diallel_geno_pheno_G, *9*
diallel_geno_pheno_G
*(*diallel_geno_pheno_G-class*)*, 4
diallel_geno_pheno_G-class, 4
diplo_freq, 4
diplo_get, 5

F1codes, 6
fine_map, 6
fitQTL, 7, *8, 9*, *15*, *20*

Gprep, *7*, 8

haplo_cluster, 9
haplo_freq, 10
haplo_get, 11
haplo_plot, 11

IBDmat, *4*, *7*, *9*, 12

LODthresh, 13

phased_parents, *9*, 14, *16*

read_data, 14, *16*
read_polyancestry, *9*, *14, 15*, 16

S1codes, 16
scan1, *15*, 17, *20*
scan1_permute, 18
scan1_summary, 19
set_params, *7*, *17*, 20