**Package ‘genotypeR’**

May 22, 2018

**Title**  SNP Genotype Marker Design and Analysis

**Version**  0.0.1.8


**Depends**  R (>= 3.3.2)

**License**  GPL (>= 3)

**URL**  https://github.com/StevisonLab/genotypeR

**Encoding**  UTF-8

**LazyData**  true

**LazyLoad**  yes

**RoxygenNote**  6.0.1

**SystemRequirements**  The SequenomMarkers() marker design function requires 'vcftools' and 'Perl' on 'windows', and, in addition, 'awk' and 'bash' on '*nix'.

**Suggests**  testthat, knitr, rmarkdown, qtl

**Imports**  methods, reshape2, plyr, doBy, zoo, colorspace

**VignetteBuilder**  knitr

**NeedsCompilation**  no

**Author**  Stephen Alfred Sefick [aut, cre], Magdelena A Castronova [aut], Laurie S Stevison [aut]

**Maintainer**  Stephen Alfred Sefick <ssefick@gmail.com>

**Repository**  CRAN

**Date/Publication**  2018-05-22 09:37:37 UTC
**R topics documented:**

- binary_coding ................................................. 2
- CO .......................................................... 3
- convert2qtl_table ........................................... 4
- count_CO ...................................................... 5
- genotypeR-class .............................................. 6
- genotypes_data .............................................. 8
- GoldenGate2iCOM_design ..................................... 8
- grep_df_subset ............................................... 9
- Heterogametic_Genotype_Warnings .......................... 10
- illumina_Genotype_Table .................................... 11
- initialize_genotypeR_data .................................. 12
- make_marker_names ......................................... 13
- markers ...................................................... 14
- read_in_illumina_GoldenGate ................................ 14
- read_in_Master_SNPs_data ................................... 15
- read_in_sequenom_data ...................................... 16
- Ref_Alt_Table ................................................ 16
- SequenomMarkers ............................................ 17
- sort_sequenom_df ............................................ 18
- subsetChromosome .......................................... 19
- zero_one_two_coding ....................................... 19

**Index**

| binary_coding | Code genotypes as binary |

**Description**

binary_coding codes genotypes contained in a genotypeR object and places them into a genotypeR object's binary_genotype slot.

**Usage**

binary_coding(genotype_warnings2NA, genotype_table)

**Arguments**

- **genotype_warnings2NA**
  - this is a genotypeR object that has been through BC_Genotype_Warnings with either output="warnings2NA" or output="pass_through"

- **genotype_table**
  - this is a marker table produced with Ref_Alt_Table

**Value**

A data frame of binary coded genotypes as a slot in the genotypeR input genotype_warnings2NA
Examples

```r
data(genotypes_data)
data(markers)
# genotype table
marker_names <- make_marker_names(markers)
GT_table <- Ref_Alt_Table(marker_names)
# remove those markers that did not work
genotypes_data_filtered <- genotypes_data[,c(1, 2, grep("TRUE",
colnames(genotypes_data) %in% GT_table$marker_names))]

warnings_out2NA <- initialize_genotyper_data(seq_data = genotypes_data_filtered,
genotype_table = GT_table, output = "warnings2NA")
genotypes_object <- binary_coding(warnings_out2NA, GT_table)
```

---

CO Where crossovers occur per individual with 2 ways to deal with missing data

Description

CO is an internal function used in count_CO to count crossovers. CO is provided in case there is a use case for the user. We used this function for QA and it can be used for estimates of crossover assurance.

Usage

```r
CO(indata, naive = FALSE)
```

Arguments

- `indata`: this is a binary coded genotype data frame from a genotypeR object (see example below).
- `naive`: this takes 2 values: 1) FALSE (default) returns list with COs distributed by marker distance, and 2) TRUE returns a list with COs without regard to marker distance (i.e., at the final non-missing data point in a string of missing genotypes)

Value

- list of COs counted per individual
Examples

library(doBy)
data(genotypes_data)
data(markers)
## genotype table
marker_names <- make_marker_names(markers)
GT_table <- Ref_Alt_Table(marker_names)
## remove those markers that did not work
genotypes_data_filtered <- genotypes_data[,c(1, 2, grep("TRUE",
    colnames(genotypes_data)%in%GT_table$marker_names))]

warnings_outZNA <- initialize_genotypeR_data(seq_data = genotypes_data_filtered,
genotype_table = GT_table, output = "warningsZNA")
binary_coding_genotypes <- binary_coding(warnings_outZNA, genotype_table = GT_table)
chr2 <- subsetChromosome(binary_coding_genotypes, chromosome="chr2")
to_count_CO <- binary_genotypes(chr2)
counted_per_individuals <- lapply(splitBy(~SAMPLE_NAME+WELL, data=to_count_CO), CO)

convert2qtl_table write out table for import into rqtl

Description

convert2qtl_table will take a genotypeR object that contains binary coded genotypes, and export
this to a csv file suitable for use with Rqtl.

Usage

convert2qtl_table(genotypeR_object,
temp_cross_for_qtl = "temp_cross_for_qtl.csv", chromosome_vect = NULL)

Arguments

genotypeR_object
    this is a genotypeR object that has had genotypes coded as binary with bi-
    nary_coding

temp_cross_for_qtl
    name of the output file that will be output into the working directory

chromosome_vect
    this is a vector of marker chromosome the length of the markers

Value

table to disk for input into rqtl
count_CO

**Examples**

data(genotypes_data)
data(markers)
## genotype table
marker_names <- make_marker_names(markers)
GT_table <- Ref_Alt_Table(marker_names)
## remove those markers that did not work
genotypes_data_filtered <- genotypes_data[,c(1, 2, grep("TRUE",
  colnames(genotypes_data)%in%GT_table$marker_names))]

warnings_out2NA <- initialize_genotypeR.data(seq_data = genotypes_data_filtered,
genotype_table = GT_table, output = "warnings2NA")
binary_coding_genotypes <- binary_coding(warnings_out2NA, genotype_table = GT_table)
chr2 <- subsetChromosome(binary_coding_genotypes, chromosome="chr2")
count_CO <- count_CO(chr2)
convert2qtl_table(count_CO, paste(temp_cross_for_qtl=getwd(),
  "test_temp_cross.csv", sep="/"),
  chromosome_vect=rep("2", (length(colnames(binary_genotypes(count_CO))-2)))

---

**count_CO**

*Internal function to remove search and remove columns based on names*

**Description**

count_CO counts crossovers from binary coded genotypes in a genotypeR object. This function assigns crossovers to the counted_crossovers slot in a genotypeR object.

**Usage**

count_CO(data, naive = FALSE)

**Arguments**

data
  genotype data read in with read_in_sequenom_data

naive
  this takes 2 values: 1) FALSE (default) will count COs distributed by marker distance, and 2) TRUE returns will count COs without regard to marker distance (i.e., at the final non-missing data point in a string of missing genotypes)

**Value**

genotypeR object with counted crossovers
Examples

data(genotypes_data)
data(markers)
## genotype table
marker_names <- make_marker_names(markers)
GT_table <- Ref_Alt_Table(marker_names)
## remove those markers that did not work
genotypes_data_filtered <- genotypes_data[, c(1, 2, grep("TRUE",
colnames(genotypes_data) %in% GT_table$marker_names))]

warnings_outNA <- initialize_genotyper_data(seq_data = genotypes_data_filtered,
genotype_table = GT_table,
output = "warnings2NA")

binary_coding_genotypes <- binary_coding(warnings_out2NA, genotype_table = GT_table)

chr2 <- subsetChromosome(binary_coding_genotypes, chromosome="chr2")

count_CO <- count_CO(chr2)


---

genotypeR-class  

Class genotypeR.

---

Description

Class genotypeR Defines a class and data structure for working with genotyping data.

Generic genotypes.

Method impossible_genotype.

Method binary_genotypes.

Method counted_crossovers.

Method binary_genotypes<-

Method genotypes<-

Method counted_crossovers<-

Method show.

Usage

genotypes(object, ...)

## S4 method for signature 'genotypeR'
genotypes(object)

impossible_genotype(object, ...)

## S4 method for signature 'genotypeR'
impossible_genotype(object)
binary_genotypes(object, ...
## S4 method for signature 'genotypeR'
binary_genotypes(object)

counted_crossovers(object, ...
## S4 method for signature 'genotypeR'
counted_crossovers(object)

binary_genotypes(object) <- value
## S4 replacement method for signature 'genotypeR'
binary_genotypes(object) <- value

genotypes(object) <- value
## S4 replacement method for signature 'genotypeR'
genotypes(object) <- value

counted_crossovers(object) <- value
## S4 replacement method for signature 'genotypeR'
counted_crossovers(object) <- value

show(object, value)
## S4 method for signature 'genotypeR'
show(object, value)

Arguments

object is a genotypeR object
... is ...
value is a value

Slots

genotypes is a data frame of genotypes
impossible_genotype is a vector with Ref/Alt that is the impossible genotype in a back cross
design

binary_genotypes is a data frame of numeric coded genotypes
counted_crossovers is a data frame of counted crossovers
**genotypes_data**  
*Genotyping data from the sequenom platform from markers produced with genotypeR*

---

**Description**

Data from a recombination plasticity experiment in Drosophila pseudoobscura. This data is provided to demonstrate the use of the genotypeR package.

**Usage**

```r
data(genotypes_data)
```

**Format**

An object read in with `read_in_sequenom_data`; see `read_in_sequenom_data`.

**Examples**

```r
data(genotypes_data)
head(genotypes_data)
colnames(genotypes_data)
```

---

**GoldenGate2iCOM_design**  
*Output GoldenGate markers for assay development with illumina iCOM*

---

**Description**

GoldenGate2iCOM_design outputs GoldenGate markers for SNP genotyping assay development with illumina iCOM.

**Usage**

```r
goldengateRicom_design(sequenomMarkers, Target_Type = "SNP", Genome_Build_Version = "0", Chromosome = "0", Coordinate = "0", Source = "unknown", Source_Version = "0", Sequence_Orientation = "unknown", Plus_Minus = "Plus")
```
Arguments

SequenomMarkers
maker data frame from make SequenomMarkers
Target_Type
SNP/Indel
Genome_Build_Version
genome build version (number; default 0)
Chromosome
(default 0)
Coordinate
genomic coordinate (default 0)
Source
unknown/sequence information (default "unknown")
Source_Version
number (default 0)
Sequence_Orientation
"forward", "reverse", "unknown" (default "unknown")
Plus_Minus
strand orientation "Plus" or "Minus" (default "Plus")

Value
A data frame suited for the genotypeR package

Examples

library(genotypeR)
data(markers)
SequenomMarkers <- markers
## this is to reduce the marker lengths to 50 bp flanking SNP
SequenomMarkers$marker <- substr(markers$marker, 51, 155)
GG_SNPs <- GoldenGate2iCOM_design(SequenomMarkers)
write.csv(GG_SNPs, "test.csv", row.names=FALSE)

Description

grep_df_subset is an internal function that subsets a data frame based on supplied pattern. This function is provided in case it is found useful.

Usage

grep_df_subset(x, toMatch)

Arguments

x
data frame where columns are to be removed
toMatch
vector of characters to remove from x
**Heterogametic Genotype Warnings**

**Value**

A subset of a genotypeR object

**Examples**

```r
df <- data.frame(one=rep(1,5), two=rep(1,5), three=rep(1,5), four=rep(1,5))
toMatch <- paste(c("one", "two"), collapse="|")
# remove toMatch
grep_df_subset(df, toMatch)
```

**Description**

Heterogametic Genotype Warnings provides QA for back cross designs by determining those organisms that have an impossible genotypes based on their sex.

**Usage**

```r
Heterogametic_Genotype_Warnings(seq_data, sex_chromosome, sex_vector, heterogametic_sex)
```

**Arguments**

- **seq_data**: is genotyping data read in with `read_in_sequenom_data`
- **sex_chromosome**: character of the sex chromosome coded in sequenom markers produced with `make_marker_names`. For example, the sex chromosome in the data provided with genotypeR is chrXL and it has been coded as chrXL_start_end. The character provided would be "chrXL"
- **sex_vector**: a vector of the sex of each individual in seq_data coded the same as that in heterogametic sex. For example, a vector of "F" and "M".
- **heterogametic_sex**: character of heterogametic sex (e.g., "M")

**Value**

A data frame of warnings
Examples

data(genotypes_data)
seq_data <- genotypes_data
sex_vector <- do.call(rbind, strsplit(seq_data[, "SAMPLE_NAME"], split=""))[2]
Heterogametic_Genotype_Warnings(seq_data=seq_data, sex_chromosome="chrX", sex_vector=sex_vector, heterogametic_sex="M")

illumina_Genotype_Table

Make genotypeR Alt_Ref_Table

Description

illumina_Genotype_Table produces the Alt_Ref_Table needed by initialize_genotyper_data from illumina’s goldengate platform.

Usage

illumina_Genotype_Table(tab_delimited_file, flanking_region_length, chromosome)

Arguments

tab_delimited_file
is a tab delimited AB illumina GoldenGate file
flanking_region_length
is the length in bp of the flanking region of the SNP
chromosome
is a vector of chromosome names

Value

data frame useful used in genotypeR

Examples

## Not run:
##Files not included to provide working example
test_data <- read_in_illumina_GoldenGate(tab_delimited_file="path_to_goldengate_file", flanking_region_length=50, chromosome=rep("chr2", length.out=length(552960)))
illumina_table <- illumina_Genotype_Table(tab_delimited_file= "path_to_goldengate_file", flanking_region_length=50, chromosome=rep("chr2", length.out=length(552960)))

## End(Not run)
**initialize_genotypeR_data**

**initialize_genotypeR_data; must provide warning allele**

**Description**

This initializes the genotypeR data structure used throughout the package.

**Usage**

initialize_genotypeR_data(seq_data, genotype_table, warning_allele = "Ref", output = "pass_through")

**Arguments**

- **seq_data** is a data frame of genotyping data
- **genotype_table** is a data frame produced with Ref_Alt_Table
- **warning_allele** is the impossible allele for a BC design taking the value "Ref" or "Alt"
- **output** this can take 3 values: 1) "warnings" which returns a data frame of BC warnings, 2) "warnings2NA" which returns a genotyping data frame where the warnings have been converted to NAs, or "pass_through" which returns a data frame that is unchanged (default).

**Value**

A genotypeR object

**Examples**

data(genotypes_data)
data(markers)
## genotype table
marker_names <- make_marker_names(markers)
GT_table <- Ref_Alt_Table(marker_names)
## remove those markers that did not work
genotypes_data_filtered <- genotypes_data[,c(1, 2, grep("TRUE", colnames(genotypes_data)%in%GT_table$marker_names))]

warnings_out2NA <- initialize_genotypeR_data(seq_data = genotypes_data_filtered, genotype_table = GT_table, output = "warnings2NA")
**make_marker_names**

*Make genotypeR compliant marker names from the output of read_in_Master_SNPs_data function*

**Description**

`make_marker_names` makes genotypeR compliant names. This is used for input into SNP assay design software. The output is also used in Ref_Alt_Table.

**Usage**

```r
make_marker_names(x)
```

**Arguments**

- **x**
  
  Output of `read_in_Master_SNPs_data`

**Value**

A data frame of GrandMasterSNPs markers with correct marker names

**Examples**

```r
data(markers)
markers <- make_marker_names(markers)

## Not run:
## example
GrandMasterSNPs_markers <- read_in_Master_SNPs_data("GrandMasterSNPs_output")
marker_names_GrandMasterSNPs_markers <- make_marker_names(GrandMasterSNPs_markers)

If subset of markers needed
use the sequenom output to subset the overall marker set from GrandMasterSNPs output
seq_test_data <- read_in_sequenom_data("path_to_sequenom_data")
col_seq_data <- colnames(seq_test_data)
col_markers <- test_data_marker_names$marker_names
markerinstudy <- test_data_marker_names[col_markers%in%col_seq_data,]

## End(Not run)
```
Description

Data from a recombination plasticity experiment in Drosophila pseudoobscura. This data is provided to demonstrate the use of the genotypeR package.

Usage

```r
data(markers)
```

Format

An object read in with `read_in_Master_SNPs_data`; see `read_in_Master_SNPs_data`.

Examples

```r
data(markers)
head(markers)
colnames(markers)
```

---

`read_in_illumina_GoldenGate`

Read in Illumina GoldenGate AB tab delimited text file

Description

`read_in_illumina_GoldenGate` reads in a tab delimited output file from illumina GoldenGate SNP genotyping platform for use in genotypeR.

Usage

```r
read_in_illumina_GoldenGate(tab_delimited_file, chromosome, flanking_region_length)
```

Arguments

- `tab_delimited_file` is a tab delimited AB illumina GoldenGate file
- `chromosome` is a vector of chromosome names
- `flanking_region_length` is the length in bp of the flanking region of the SNP
read_in_Master_SNPs_data

Value

data frame useful used in genotypeR

Examples

## Not run:
test_data <- read_in_illumina_GoldenGate(tab_delimited_file= \  "path_to_golden_gate_file", flanking_region_length=50, \ chromosome=rep("chr2", length.out=length(552960)))

## End(Not run)

---

read_in_Master_SNPs_data

*Read in GrandMasterSNPs output*

Description

This reads in single nucleotide polymorphism markers generated by the GrandMasterSNPs Perl program.

Usage

```
read_in_Master_SNPs_data(x, ...)
```

Arguments

- `x` : This is a tab delimited text file from GrandMasterSNPs Perl program
- `...` : Other arguments passed to the function

Value

A data frame of GrandMasterSNPs markers

Examples

```r
##this should be used with the output of the PERL pipeline "GrandMasterSNPs"
marker_file <- system.file("extdata/filtered_markers.txt", package = "genotypeR")

GrandMasterSNPs_markers <- read_in_Master_SNPs_data(marker_file)
```
read_in_sequenom_data  

Read in Sequenom Data

Description

read_in_sequenom_data reads in a csv file produced from the Sequenom platform (i.e., sequenom excel output saved as a csv).

This function is a wrapper function around read.csv in order to read genotype data from the Sequenom Platform, and provide data compatible with the genotypeR package.

Usage

read_in_sequenom_data(x, sort_char = "chr|contig", ...)

Arguments

x

This is a csv formatted Genotypes tab of exported sequenom data that you would like to read in.

sort_char

is the character string output by the PERL pipeline in the marker design phase (i.e., chr 1000 1050 AAA[A/T]GTC; the chr is the sort_char. Defaults to chr or contig.

... Other arguments passed to the function

Value

A data frame suited for the genotypeR package

Examples

sequenom_file <- system.file("extdata/sequenom_test_data.csv", package = "genotypeR")
sequenom_data <- read_in_sequenom_data(sequenom_file)

ref_alt_table

Make reference/alternate allele table from make_marker_names output

Description

Ref_Alt_Table makes the ref/alt table used in for proper genotype coding and QA/QC.

Usage

Ref_Alt_Table(markers_in_study)
SequenomMarkers

Arguments

markers_in_study
make_marker_names output

Value

A data frame of Ref/Alt genotypes

Examples

data(markers)
markers_in_study <- make_marker_names(markers)
genotype_table <- Ref_Alt_Table(markers_in_study = markers_in_study)

SequenomMarkers  R wrapper script to run Sequenom Marker design pipeline

Description

SequenomMarkers runs the SNP genotyping marker design portion of the genotypeR pipeline.
This function designs Sequenom markers.

Usage

SequenomMarkers(vcf1 = NULL, vcf2 = NULL, outdir = NULL,
platform = "sq")

Arguments

vcf1 this is an uncompressed vcf file (Ref allele)
vcf2 this is an uncompressed vcf file (Alt allele)
outdir this is where the tab-delimited extended bed file will be written
platform is a character vector taking "sq" for sequenom (100 bp reference flanking region)
or "gg" for goldengate (50 bp reference flanking region).

Value

SequenomMarker design into "outdir"
sort_sequenom_df

Examples

```r
## Not run:
example_files <- system.file("SequenomMarkers_v2/two_sample/test_files", package = "genotypeR")

vcf1 <- paste(example_files, "Sample1.vcf", sep="/")
vcf2 <- paste(example_files, "Sample2.vcf", sep="/")

#look in outdir to look at the results in Master_SNPs.sorted.txt.
outdir <- paste(example_files, "test_dir", sep="/")

SequenomMarkers(vcf1, vcf2, outdir, platform="sq")

## End(Not run)
```

---

**sort_sequenom_df**  
Sequenom Data frame Sort

**Description**

This function sorts Sequenom Data at the read-in stage.

**Usage**

```r
sort_sequenom_df(Sequenom_Data2Sort, sort_char = "chr|contig")
```

**Arguments**

- `Sequenom_Data2Sort`: data frame to sort produced with the genotypeR package
- `sort_char`: is the character string output by the PERL pipeline in the marker design phase (i.e., chr 1000 1050 AAA[A/T]GTC; the chr is the sort_char. Defaults to chr or contig.

**Value**

A sorted data frame suited for the genotypeR package

**Examples**

```r
data(genotypes_data)
sort_sequenom_df(genotypes_data)
```
subsetChromosome

Subset genotypeR object by chromosome

Description

subsetChromosome subsets a genotypeR object based on the supplied chromosome name (must be the same as that in the data).

Usage

subsetChromosome(aa, chromosome)

Arguments

aa            genotypeR object before binary coding
chromosome    which chromosome to pull out (e.g., "chr2")

Value

A genotypeR object subset based on the pattern supplied with chromosome

Examples

data(genotypes_data)
data(markers)
## genotype table
marker_names <- make_marker_names(markers)
GT_table <- Ref_Alt_Table(marker_names)
## remove those markers that did not work
genotypes_data_filtered <- genotypes_data[,c(1, 2, grep("TRUE",
colnames(genotypes_data)$in%GT_table$marker_names))]

warnings_outZNA <- initialize_genotypeR_data(seq_data = genotypes_data_filtered,
genotype_table = GT_table, output = "warnings2NA")
chromosome_subset <- subsetChromosome(warnings_outZNA, "chr2")

zero_one_two_coding

Code genotypes as 0, 1, 2

Description

zero_one_two_coding code homozygous reference as 0, heterozygous as 1, and homozygous alternate as 2 using a genotypeR object created with initialize_genotypeR_data with the pass_through argument.
zero_one_two_coding

Usage

```
zero_one_two_coding(genotype_warnings_passthrough, genotype_table)
```

Arguments

- `genotype_warnings_passthrough` is a genotype object that has been processed by `BC_Genotype_Warnings` with `output="pass_through"`
- `genotype_table` is a data frame produced with `Ref_Alt_Table`

Value

A data frame of 0, 1, and 2 coded genotypes as a slot in the input

Examples

```r
data(genotypes_data)
data(markers)
## genotype table
marker_names <- make_marker_names(markers)
GT_table <- Ref_Alt_Table(marker_names)
## remove those markers that did not work
genotypes_data_filtered <- genotypes_data[,c(1, 2, grep("TRUE",
colnames(genotypes_data)%in%GT_table$marker_names))]

pass_through <- initialize_genotyper_data(seq_data = genotypes_data_filtered,
genotype_table = GT_table, output = "pass_through")
genotypes_object <- zero_one_two_coding(pass_through, GT_table)
```
Index

*Topic 0
  zero_one_two_coding, 19
*Topic 1
  zero_one_two_coding, 19
*Topic 2
  zero_one_two_coding, 19
*Topic Alt
  zero_one_two_coding, 19
*Topic Count
  count_C0, 5
*Topic Crossovers
  count_C0, 5
*Topic GoldenGate
  illumina_Genotype_Table, 11
  read_in_illumina_GoldenGate, 14
*Topic GrandMasterSNPs
  make_marker_names, 13
  read_in_Master_SNPs_data, 15
*Topic Heterozygous,
  zero_one_two_coding, 19
*Topic Homozygous
  zero_one_two_coding, 19
*Topic Ref,
  zero_one_two_coding, 19
*Topic SequenomMarker
  SequenomMarkers, 17
*Topic Sequenom
  GoldenGate2iCOM_design, 8
*Topic and
  zero_one_two_coding, 19
*Topic associated
  make_marker_names, 13
*Topic as
  binary_coding, 2
  zero_one_two_coding, 19
*Topic a
  count_C0, 5
*Topic binary
  binary_coding, 2

*Topic by
  subsetChromosome, 19
*Topic chromosome
  subsetChromosome, 19
*Topic code
  binary_coding, 2
  zero_one_two_coding, 19
*Topic columns
  grep_df_subset, 9
*Topic compliant
  make_marker_names, 13
*Topic count
  C0, 3
*Topic crossovers
  C0, 3
*Topic data
  genotypes_data, 8
  markers, 14
*Topic genotypeR,
  convert2qtl_table, 4
*Topic genotypeR
  C0, 3
  count_C0, 5
  initialize_genotypeR_data, 12
  make_marker_names, 13
  subsetChromosome, 19
*Topic genotypes
  binary_coding, 2
  zero_one_two_coding, 19
*Topic illumina
  illumina_Genotype_Table, 11
  read_in_illumina_GoldenGate, 14
*Topic make_marker_names
  Ref_Alt_Table, 16
*Topic markers
  make_marker_names, 13
*Topic marker
  make_marker_names, 13
*Topic matching
grep_df_subset, 9
*Topic names
  make_marker_names, 13
*Topic object
  count_C0, 5
  subsetChromosome, 19
*Topic of
  count_C0, 5
*Topic read
  GoldenGate2iCOM_design, 8
  read_in_Master_SNPs_data, 15
  read_in_sequenom_data, 16
*Topic remove
  grep_df_subset, 9
*Topic rqtl
  convert2qtl_table, 4
*Topic sequenom
  read_in_sequenom_data, 16
  sort_sequenom_df, 18
*Topic sets
  genotypes_data, 8
  markers, 14
*Topic sort
  sort_sequenom_df, 18
*Topic subset
  subsetChromosome, 19
*Topic the
  make_marker_names, 13
*Topic with
  make_marker_names, 13
binarystandardizing, 2
binary_genotypes, 6
binary_genotypes, genotypeR-method
  (genotypeR-class), 6
binary_genotypes<- (genotypeR-class), 6
binary_genotypes<-, genotypeR-method
  (genotypeR-class), 6
genotypes, 6
gettype_R(genotypeR-class), 6
genotypes, genotypeR-method
  (genotypeR-class), 6
impossible_genotype, 6
impossible_genotype, genotypeR-method
  (genotypeR-class), 6
initialize_genotyper_data, 12
make_marker_names, 13
markers, 14
read_in_illumina_GoldenGate, 14
read_in_Master_SNPs_data, 15
read_in_sequenom_data, 8, 16
Ref_Alt_Table, 16
SequenomMarkers, 17
show, 6
show, genotypeR-method
  (genotypeR-class), 6
sort_sequenom_df, 18
subsetChromosome, 19
zero_one_two_coding, 19