Package 'GALLO'

February 10, 2026

Title Genomic Annotation in Livestock for Positional Candidate LOci

Version 2.0

Description The accurate annotation of genes and Quantitative Trait Loci (QTLs) located within candidate markers and/or regions (haplotypes, windows, CNVs, etc) is a crucial step the most common genomic analyses performed in livestock, such as Genome-Wide Association Studies or transcriptomics. The Genomic Annotation in Livestock for positional candidate LOci (GALLO) is an R package designed to provide an intuitive and straightforward environment to annotate positional candidate genes and QTLs from high-throughput genetic studies in livestock. Moreover, GALLO allows the graphical visualization of gene and QTL annotation results, data comparison among different grouping factors (e.g., methods, breeds, tissues, statistical models, studies, etc.), and QTL enrichment in different livestock species including cattle, pigs, sheep, and chicken, among others.

URL <https://github.com/pablobio/GALLO>

Depends R (>= 4.0.0)

biocViews Software

Imports circlize, data.table, doParallel, dplyr, ggplot2, graphics, grDevices, foreach, lattice , parallel, RColorBrewer, rtracklayer, stats, stringr, unbalhaar, utils, DT, webshot, igraph, visNetwork, CompQuadForm, Matrix, reticulate

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests Hmisc, knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Pablo Fonseca [aut, cre], Aroa Suarez-Vega [aut], Gabriele Marras [aut], Angela Cánovas [aut]

Maintainer Pablo Fonseca <p.fonseca@csic.es>

Repository CRAN

Date/Publication 2026-02-09 23:50:20 UTC

Contents

`find_genes_qtls_around_markers`
Search genes and QTLs around candidate regions

Description

Takes a list of candidate markers and or regions (haplotypes, CNVs, windows, etc.) and search for genes or QTLs in a determined interval

Usage

```
find_genes_qtls_around_markers(
  db_file,
  marker_file,
  method = c("gene", "qtl"),
  marker = c("snp", "haplotype"),
  interval = 0,
  nThreads = NULL,
  verbose = TRUE
)
```

Arguments

db_file	The data frame obtained using the import_gff_gtf() function
marker_file	The file with the SNP or haplotype positions. Detail: For SNP files, the columns "CHR" and "BP" with the chromosome and base pair position, respectively, are mandatory. For the haplotype, the following columns are mandatory: "CHR", "BP1" and "BP2"
method	"gene" or "qtl"
marker	"snp" or "haplotype"
interval	The interval in base pair which can be included upstream and downstream from the markers or haplotype coordinates.
nThreads	Number of threads to be used
verbose	Logical value defining if messages should of not be printed during the analysis (default=TRUE)

Value

A dataframe with the genes or QTLs mapped within the specified intervals

Examples

```
data(QTLmarkers)
data(gffQTLs)
out.qtls<-find_genes_qtls_around_markers(db_file=gffQTLs, marker_file=QTLmarkers,
method = "qtl", marker = "snp",
interval = 500000, nThreads = 1)
```

Gene_PPA *Function to estimate the posterior probability of association (PPA) based on the integration of -OMICs and functional data via representation learning of candidate genes.*

Description

Function to estimate the posterior probability of association (PPA) based on the integration of -OMICs and functional data via representation learning of candidate genes.

Usage

```
Gene_PPA(
  gene.pval,
  Gene_ID,
  pval,
  embeddings,
  lambda = 0,
  alpha0 = 1,
```

```
  alpha1 = 0.2,
  pi0 = 0.6,
  pi1 = 0.4,
  max.it = 10,
  model,
  verbose = F
)
```

Arguments

gene.pval	A table containing the gene IDs and p-values.
Gene_ID	Name of the column containing the gene IDs (without NAs or duplicated IDs).
pval	Name of the column containing the gene-level p-values.
embeddings	The output of the Net_embedding function.
lambda	Ridge (L_2) regularization strength used only when model = "LR" for the logistic regression part that models the prior probability of being in the signal component.
alpha0	Mixing proportion (prior probability) that an observation belongs to the alternative/signal component $Z = 1$ in the two-component mixture; typically $\pi_1 = 1 - \pi_0$.
alpha1	Shape parameter of the alternative/signal p-value distribution, modeled as $p \sim \mathrm{Beta}(\alpha_1, 1)$ when $Z = 1$; values < 1 concentrate mass near 0 (enrichment of small p-values).
pi0	Mixing proportion (prior probability) that an observation belongs to the null/background component $Z = 0$ in the two-component mixture.
pi1	Mixing proportion (prior probability) that an observation belongs to the alternative/signal component $Z = 1$ in the two-component mixture; typically $\pi_1 = 1 - \pi_0$.
max.it	Maximum number of iterations for the EM algorithm.
model	One of the following options defining the model used during the EM algorithm: proj, M, LR, NB, and MVN. See Details for more information.
verbose	A logical value defining if the iteration information should be printed or not.

Details

The function Gene_PPA allows the user to select different options that define how the EM algorithm models the distribution of v (the auxiliary information matrix (feature/annotation matrix) attached to each p-value). The EM algorithms employed here are adapted from Wu et al. (2018):

- proj: Fits a two-component mixture model that combines p-values with an LDA-based one-dimensional projection of the feature matrix v. The projection is obtained by soft LDA using current posterior weights, and class-conditional densities of the projected score are estimated nonparametrically via weighted KDE.
- M: Fits a two-component mixture model that combines p-values with the feature matrix v by estimating a separate weighted KDE for each feature and each component. The joint density is approximated by the product of marginal densities (naive independence across features).

- `LR`: Fits a two-component mixture model where p-values follow a Beta-mixture and the prior probability of belonging to the signal component is modeled as a logistic function of v. Logistic regression parameters are estimated within the EM iterations, with optional ridge regularization controlled by `lambda`.
- `NB`: Fits a two-component mixture model that combines p-values with v assuming a Gaussian naive Bayes model for the features. Component-specific feature means are estimated, with feature-wise variances shared across components, and features are treated as conditionally independent given component membership.
- `MVN`: Fits a two-component mixture model that combines p-values with v using a multivariate normal model with shared covariance (parametric, correlation-aware).

Value

Returns the posterior probability of association for each gene.

References

Wu et al. (2018) Methods, 145, doi:10.1016/j.ymeth.2018.06.002.

`gene_pval` *Estimate a gene-level p-value using Weighted Z-score approach, Meta-analysis with LD correlation coefficients approach, and Davies algorithm*

Description

Estimate a gene-level p-value using Weighted Z-score approach, Meta-analysis with LD correlation coefficients approach, and Davies algorithm

Usage

```
gene_pval(data, db_file, marker_ld, interval, p)
```

Arguments

`data`	A data frame with the results of the association test performed for each marker
`db_file`	A data frame obtained from the import_gff_gtf containing the gtf information
`marker_ld`	A data frame containing the pairwise linkage disequilibrium between markers in a chromosome
`interval`	The interval (in base pairs) used to annotated markers downstream and upstream from the genes coordinates
`p`	The name of the column containing the P-values for each marker

Details

Requires a table with p-values from a association test, a gtf file file the gene coordinates in the same assembly used to map the variants used in the association study, and a data frame with pairwise linkage disequilibrium (LD) values between markers. This analysis must be performed for each chromosome individually. The data frame with the results of the association study must have three mandatory columns names as CHR, BP and SNP containing the chromosome, base pair position and marker name, respectively. The gtf file must be imported by the import_gff_gtf() function from GALLO or can be customized by the user, since it has the same columns names. The LD table must contain three mandatory columns, SNP_A, SNP_B and R. where, the first two columns must contain the marker names and the third column, the LD value between these markers. This data frame can be obtained using PLINK or any other software which computes pairwise LD between markers in the same chromosome. In the absence of LD values between any two SNPs in the data frame, a LD equal zero is assumed

Value

A data frame with the gene level p-values obtained using the Weighted Z-score approach (P_WZ_ld), Meta-analysis with LD correlation coefficients approach (P_meta_LD), and Liu algorithm (P_Liu, Liu et al. (2009))

References

Liu et al. (2009) Computational Statistics & Data Analysis, Volume 53, (doi:10.1016/j.csda.2008.11.025)

`import_gff_gtf` *Import .gtf and .gff files to be used during gene and QTL annotation, respectively*

Description

Takes a .gft or .gff file and import into a dataframe

Usage

```
import_gff_gtf(db_file, file_type)
```

Arguments

`db_file`	File with the gene mapping or QTL information. For gene mapping, a .gtf file from Ensembl database must be used. For the QTL search, a .gff file from Animal QTlLdb must be used. Both files must use the same reference annotation used in the original study
`file_type`	"gtf" or "gff"

Value

A dataframe with the gtf or gtf content

Examples

```
gffpath <- system.file("extdata", "example.gff", package="GALLO")

qtl.inp <- import_gff_gtf(db_file=gffpath,file_type="gff")
```

NetCen *Compute the centrality metrics for the nodes composing the network generated by the NetVis function*

Description

Compute the centrality metrics for the nodes composing the network generated by the NetVis function

Usage

```
NetCen(data, g1, g2)
```

Arguments

data	A data frame containing the relationship between the two groups to be represented in the network
g1	Name of the column containing the labels of the first group that will be used to create the network
g2	Name of the column containing the labels of the second group that will be used to create the network

Details

This function returns the following centrality metrics for each node that composed the network: Degree (The number of edges incident to the node), Betweenness (The fraction of shortest paths between pairs of nodes that pass through the node), Closeness (The inverse of the sum of the shortest path distances from the node to all other nodes), and Eigenvector Centrality (The centrality measure based on the eigenvector of the adjacency matrix).

Value

A data frame with the centrality metrics for each node in the network.

NetVis *Create a dynamic network representing the relationship between two groups of variables*

Description

Create a dynamic network representing the relationship between two groups of variables

Usage

```
NetVis(
  data,
  g1,
  g2,
  col1 = "aquamarine",
  col2 = "red",
  edge_col = "gray",
  remove_label = NULL,
  node_size = c(15, 40),
  font_size = 45,
  edge_width = 1
)
```

Arguments

data	A data frame containing the relationship between the two groups to be represented in the network
g1	Name of the column containing the labels of the first group that will be used to create the network
g2	Name of the column containing the labels of the second group that will be used to create the network
col1	Color of the nodes that will represent the first group represented in the network. The default value is aquamarine
col2	Color of the nodes that will represent the second group represented in the network The default value is red
edge_col	Color of the edges that will connect the nodes in the network. The default value is gray
remove_label	If is required to omit the labels for some of the groups, this argument receives the column name informed the g1 or g2 arguments. The default value is NULL
node_size	A vector with the node sizes to represent g1 and g2. The defaul values are 15 and 40, respectively
font_size	The size of the font of the labels of each node (The default value is 45)
edge_width	The width of the edges connecting the nodes in the network

Details

This function returns a dynamic network, using visNetwork, representing the connection between two groups. For example, the output from the find_genes_qtls_around_markers() function can be used here to represent the connections between markers and QTLs. Another option is to combine the data frames with both gene and QTL annotation around markers to reprsent the connections between genes and QTLs.

Value

A dynamic network representing the connection between two groups.

`Net_embedding` *Network Embedding using biased random walk and Word2Vec.*

Description

Network Embedding using biased random walk and Word2Vec.

Usage

```
Net_embedding(
  net_list,
  p = 0.25,
  q = 1,
  num = 10,
  l = 80,
  vector_size = 32,
  window = 10,
  min_count = 1,
  sg_model = 1,
  workers = 4,
  epochs = 20
)
```

Arguments

`net_list`	A list containing data frames that can be coerced into pandas data frames. Each data frame contains 3 columns, source, target and weight, representing the connections between nodes in the networks.
`p`	A parameter for the BiasedRandomWalk.Defines probability, 1/p, of returning to source node.
`q`	A parameter for the BiasedRandomWalk.Defines probability, 1/q, for moving to a node away from the source node.
`num`	A parameter for the BiasedRandomWalk. Defines the number of walks per node.
`l`	A parameter for the BiasedRandomWalk. Defines the length of each walk.

vector_size	The size of the Word2Vec vectors.
window	The window size for the Word2Vec model.
min_count	Minimum count for the Word2Vec model.
sg_model	Training algorithm for Word2Vec (0 for CBOW, 1 for skip-gram).
workers	Number of worker threads to train the Word2Vec model.
epochs	Number of epochs to train the Word2Vec model.

Details

This function performs a network embedding using the python libraries stellargraph and gensim through the BiasedRandomWalk and Word2Vec functions, respectively.

Value

A list of data frames with node embeddings. Each data frame contains nxm dimensions, where n is the number of unique nodes in the network and m is the number of reduced dimensions defined in the function.

Nmarkers_SimpleM *Estimate the number of effective markers in a chromosome based on an adapted version of the simpleM methodology*

Description

Estimate the number of effective markers in a chromosome based on an adapted version of the simpleM methodology

Usage

```
Nmarkers_SimpleM(ld.file, PCA_cutoff = 0.995)
```

Arguments

ld.file	A data frame with the pairwise linkage disequilibrium (LD) values for a chromosome. The column names SNP_A, SNP_B, and R are mandatory, where the SNP_A and SNP_B contained the markers names and the R column the LD values between the two markers.
PCA_cutoff	A cutoff for the total of the variance explained by the markers.

Details

This function estimate the effective number of markers in a chromosome using adapted version of the simpleM methodology described in Gao et al. (2008). The function use as input a data frame composed by three mandatory columns (SNP_A, SNP_B, and R). This data frame can be obtained using PLINK or any other software to compute LD between markers. Additionally, a threshold for percentage of the sum of the variances explained by the markers must be provided. The number of effective markers identified by this approach can be used in multiple testing corrections, such as Bonferroni.

Value

The effective number of markers identified by the SimpleM approach

References

Gao et al. (2008) Genet Epidemiol, Volume 32, Issue 4, Pages 361-369. (doi:10.1002/gepi.20310)

`Nseg_chr` *Estimate the number of independent segments in a chromosome based on the effective population size*

Description

Estimate the number of independent segments in a chromosome based on the effective population size

Usage

```
Nseg_chr(chr.table, chr_length, Ne)
```

Arguments

`chr.table`	A table containing the chromosomes and the chromosomal length (in centiMorgans).
`chr_length`	The name of the column where the length of the chromosomes are informed.
`Ne`	The effective population size.

Details

This function uses a adapted version of the formula proposed by Goddard et al. (2011) to estimate the independent number of segments in a chromosome based on the effective population size.

Value

A data frame with the effective number of segments in each chromosome.

References

Goddard et al. (2011) Journal of animal breeding and genetics, Volume 128, Issue 6, Pages 409-421. (doi:10.1111/j.14390388.2011.00964.x)

`overlapping_among_groups` *Overlapping between grouping factors*

Description

Takes a dataframe with a column of genes, QTLs (or other data) and a grouping column and create some matrices with the overlapping information

Usage

```
overlapping_among_groups(file, x, y)
```

Arguments

`file`	A dataframe with the data and grouping factor
`x`	The grouping factor to be compared
`y`	The data to be compared among the levels of the grouping factor

Value

A list with three matrices: 1) A matrix with the number of overlapping data; 2) A matrix with the percentage of overlapping; 3) A matrix with the combination of the two previous one

Examples

```
data(QTLmarkers)
data(gtfGenes)
genes.out <- find_genes_qtls_around_markers(db_file=gtfGenes,
marker_file=QTLmarkers,method="gene",
marker="snp",interval=100000, nThreads=1)
overlapping.out<-overlapping_among_groups(
file=genes.out,x="Reference",y="gene_id")
```

`PleioChiTest` *Compute a multi-trait test statistic for pleiotropic effects using summary statistics from association tests*

Description

Compute a multi-trait test statistic for pleiotropic effects using summary statistics from association tests

Usage

```
PleioChiTest(data)
```

Arguments

data	A data frame with the first column containing the SNP name and the remaining columns the signed t-values obtained for each marker in the association studies individually performed for each trait.

Details

This function tests a null hypothesis stating that each SNP does not affect any of the traits included in the input file. The method applied here is an implementation of the statistic proposed at Bolormaa et al. (2014) and is approximately distributed as a chi-squared with n degrees of freedom, where n is equal the number of traits included in the input file.

Value

A data frame with the multi-trait chi-squared statistics and the correspondent p-value obtained for each SNP.

References

Bolormaa et al. (2014) Plos Genetics, Volume 10, Issue 3, e1004198. (doi:10.1371/journal.pgen.1004198)

---

plot_overlapping *Plot overlapping between data and grouping factors*

Description

Takes the output from overlapping_among_groups function and creates a heatmap with the overlapping between groups

Usage

```
plot_overlapping(overlapping_matrix, nmatrix, ntext, group, labelcex = 1)
```

Arguments

overlapping_matrix	The object obtained in overlapping_amoung_groups function
nmatrix	An interger from 1 to 3 indicating which matrix will be used to plot the overlapping, where: 1) A matrix with the number of overllaping data; 2) A matrix with the percentage of overlapping; 3) A matrix with the combination of the two previous one
ntext	An interger from 1 to 3 indicating which matrix will be used as the text matrix for the heatmap, where: 1) A matrix with the number of overllaping data; 2) A matrix with the percentage of overlapping; 3) A matrix with the combination of the two previous one
group	A vector with the size of groups. This vector will be plotted as row and column names in the heatmap
labelcex	A numeric value indicating the size of the row and column labels

Value

A heatmap with the overlapping between groups

Examples

```
data(QTLmarkers)
data(gtfGenes)
genes.out <- find_genes_qtls_around_markers(
db_file=gtfGenes, marker_file=QTLmarkers,
method="gene", marker="snp",interval=100000,
nThreads=1)

overlapping.out<-overlapping_among_groups(
file=genes.out,x="Reference",y="gene_id")
plot_overlapping(overlapping.out,
nmatrix=2,ntext=2,
group=unique(genes.out$Reference))
```

---

plot_qtl_info *Plot QTLs information from the find_genes_qtls_around_markers output*

Description

Takes the output from find_genes_qtls_around_markers and create plots for the frequency of each QTL type and trait

Usage

```
plot_qtl_info(
  qtl_file,
  qtl_plot = c("qtl_type", "qtl_name"),
  n = "all",
  qtl_class = NULL,
  horiz = FALSE,
  ...
)
```

Arguments

qtl_file	The output from find_genes_qtls_around_markers function
qtl_plot	"qtl_type" or"qtl_name"
n	Number of QTLs to be plotted when the qtl_name option is selected
qtl_class	Class of QTLs to be plotted when the qtl_name option is selected
horiz	The legend of the pie plot for the qtl_type should be plotted vertically or horizontally. The default is FALSE. Therefore, the legend is plotted vertically.

...	Arguments to be passed to/from other methods. For the default method these can include further arguments (such as axes, asp and main) and graphical parameters (see par) which are passed to plot.window(), title() and axis.

Value

A plot with the requested information

Examples

```
data(QTLmarkers)
data(gffQTLs)

out.qtls<-find_genes_qtls_around_markers(db_file=gffQTLs,
marker_file=QTLmarkers, method = "qtl",
marker = "snp", interval = 500000,
nThreads = 1)

plot_qtl_info(out.qtls, qtl_plot = "qtl_type", cex=2)
```

QTLenrich_plot *Plot enrichment results for QTL enrichment analysis*

Description

Takes the output from qtl_enrich function and creates a bubble plot with enrichment results

Usage

```
QTLenrich_plot(qtl_enrich, x, pval)
```

Arguments

qtl_enrich	The output from qtl_enrich function
x	Id column to be used from the qtl_enrich output
pval	P-value to be used in the plot. The name informed to this argument must match the p-value column name in the enrichment table

Value

A plot with the QTL enrichment results

qtl_enrich *Performs a QTL enrichment analysis based on a hypergeometric test for each QTL class*

Description

Takes the output from find_genes_qtls_around_markers and run a QTL enrichment analysis

Usage

```
qtl_enrich(
  qtl_db,
  qtl_file,
  qtl_type = c("QTL_type", "Name"),
  enrich_type = c("genome", "chromosome"),
  chr.subset = NULL,
  nThreads = NULL,
  padj = c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"),
  verbose = TRUE
)
```

Arguments

qtl_db	The object obtained using the import_gff_gtf() function
qtl_file	The output from find_genes_qtls_around_markers function
qtl_type	A character indicating which type of enrichment will be performed. QTL_type indicates that the enrichment processes will be performed for the QTL classes, while Name indicates that the enrichment analysis will be performed for each trait individually
enrich_type	A character indicating if the enrichment analysis will be performed for all the chromosomes ("genome") or for a subset of chromosomes ("chromosome). If the "genome" option is selected, the results reported are the merge of all chromosomes
chr.subset	If enrich_type equal "chromosome", it is possible to define a subset of chromosomes to be analyzed. The default is equal NULL. Therefore, all the chromosomes will be analyzed
nThreads	The number of threads to be used.
padj	The algorithm for multiple testing correction to be adopted ("holm", "hochberg", "hommel", "bonferroni", "BH", "BY","fdr", "none")
verbose	Logical value defining if messages should of not be printed during the analysis (default=TRUE)

Details

The simple bias of investigation for some traits (such as milk production related traits in the QTL database for cattle) may result in a larger proportion of records in the database. Consequently, the simple investigation of the proportion of each QTL type might not be totally useful. In order to reduce the impact of this bias, a QTL enrichment analysis can be performed. The QTL enrichment analysis performed by GALLO package is based in a hypergeometric test using the number of annoatted QTLs within the candidate regions and the total number of the same QTL in the QTL database.

Value

A data frame with the p-value for the enrichment result

Examples

```
data(QTLmarkers)
data(gffQTLs)
out.qtls<-find_genes_qtls_around_markers(
db_file=gffQTLs,marker_file=QTLmarkers,
method = "qtl",marker = "snp",
interval = 500000, nThreads = 1)

out.enrich<-qtl_enrich(qtl_db=gffQTLs,
qtl_file=out.qtls, qtl_type = "Name",
enrich_type = "chromosome",chr.subset = NULL,
padj = "fdr",nThreads = 1)
```

---

relationship_plot *Plot relationship between data and grouping factors*

Description

Takes the output from find_genes_qtls_around_markers function and creates a chord plot with the relationship between groups

Usage

```
relationship_plot(
  qtl_file,
  x,
  y,
  grid.col = "gray60",
  degree = 90,
  canvas.xlim = c(-2, 2),
  canvas.ylim = c(-2, 2),
  cex,
  gap
)
```

Arguments

qtl_file	The output from find_genes_qtls_around_markers function
x	The first grouping factor, to be plotted in the left hand side of the chord plot
y	The second grouping factor, to be plotted in the left hand side of the chord plot
grid.col	A character with the grid color for the chord plot or a vector with different colors to be used in the grid colors. Note that when a color vector is provided, the length of this vector must be equal the number of sectors in the chord plot
degree	A numeric value corresponding to the starting degree from which the circle begins to draw. Note this degree is always reverse-clockwise
canvas.xlim	The coordinate for the canvas in the x-axis. By default is c(-1,1)
canvas.ylim	The coordinate for the canvas in the y-axis. By default is c(-1,1)
cex	The size of the labels to be printed in the plot
gap	A numeric value corresponding to the gap between the chord sectors

Value

A chords relating x and y

Examples

```
data(QTLmarkers)
data(gffQTLs)
out.qtls<-find_genes_qtls_around_markers(
db_file=gffQTLs, marker_file=QTLmarkers,
method = "qtl", marker = "snp",
interval = 500000, nThreads = 1)

out.enrich<-qtl_enrich(qtl_db=gffQTLs,
qtl_file=out.qtls, qtl_type = "Name",
enrich_type = "chromosome",
chr.subset = NULL, padj = "fdr",nThreads = 1)

out.enrich$ID<-paste(out.enrich$QTL," - ",
"CHR",out.enrich$CHR,sep="")

out.enrich.filtered<-out.enrich[which(out.enrich$adj.pval<0.05),]

out.qtls$ID<-paste(out.qtls$Name," - ",
"CHR",out.qtls$CHR,sep="")

out.enrich.filtered<-out.enrich.filtered[order(out.enrich.filtered$adj.pval),]

out.qtls.filtered<-out.qtls[which(out.qtls$ID%in%out.enrich.filtered$ID[1:10]),]

out.qtls.filtered[which(out.qtls.filtered$Reference==
"Feugang et al. (2010)"), "color_ref"]<-"purple"

out.qtls.filtered[which(out.qtls.filtered$Reference==
```

```
"Buzanskas et al. (2017)"),"color_ref"]<-"pink"

color.grid<-c(rep("black",length(unique(out.qtls.filtered$Abbrev))),
unique(out.qtls.filtered$color_ref))

names(color.grid)<-c(unique(out.qtls.filtered$Abbrev),
unique(out.qtls.filtered$Reference))

relationship_plot(qtl_file=out.qtls.filtered,
x="Abbrev", y="Reference",cex=1,gap=5,
degree = 90, canvas.xlim = c(-5, 5),
canvas.ylim = c(-3, 3), grid.col = color.grid)
```

Index