Package 'qtlcharts'

June 19, 2026

Version 0.22

Date 2026-06-19

Title Interactive Graphics for QTL Experiments

Author Karl W Broman [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4914-6671>),
Michael Bostock [ctb, cph] (d3.js library in htmlwidgets/lib,
https://d3js.org),
jQuery Foundation [cph] (jQuery library in htmlwidgets/lib,
https://jquery.com),
jQuery contributors [ctb] (jQuery library in htmlwidgets/lib; see
https://github.com/jquery/jquery/blob/master/AUTHORS.txt),
jQuery UI contributors [ctb] (jQuery UI library in htmlwidgets/lib; see
https://jqueryui.com/about/)

Maintainer Karl W Broman <broman@wisc.edu>

Description Web-based interactive charts (using D3.js) for the analysis of
experimental crosses to identify genetic loci (quantitative trait
loci, QTL) contributing to variation in quantitative traits.
Broman (2015) <doi:10.1534/genetics.114.172742>.

Depends R (>= 2.15)

Imports qtl (>= 1.30-4), htmlwidgets, graphics, stats, utils

Suggests htmltools, jsonlite, knitr, devtools, roxygen2, rmarkdown,
testthat

License GPL-3 | file LICENSE

URL https://kbroman.org/qtlcharts/,
https://github.com/kbroman/qtlcharts

BugReports https://github.com/kbroman/qtlcharts/issues

LazyData true

Encoding UTF-8

ByteCompile true

Config/roxygen2/version 8.0.0

NeedsCompilation no

Repository CRAN

Date/Publication 2026-06-19 12:20:18 UTC

Contents

qtlcharts-package . . . 2
cbindQTLeffects . . . 4
estQTLeffects . . . 4
geneExpr . . . 5
grav . . . 6
iboxplot . . . 7
idotplot . . . 8
iheatmap . . . 9
ipleiotropy . . . 10
iplot . . . 11
iplotCorr . . . 12
iplotCurves . . . 14
iplotMap . . . 15
iplotMScanone . . . 16
iplotPXG . . . 18
iplotRF . . . 19
iplotScanone . . . 20
iplotScantwo . . . 22
itimeplot . . . 23
itriplot . . . 25
qtlcharts-shiny . . . 26
qtlchartsversion . . . 27
scat2scat . . . 28
setScreenSize . . . 29

Index **30**

`qtlcharts-package` *R/qtlcharts: Interactive charts for QTL data*

Description

A QTL is a *quantitative trait locus*: a genetic locus that contributes to variation in a quantitative trait. The goal of R/qtlcharts is to provide interactive data visualizations for QTL analyses, and to make these visualizations available from R. It is a companion to the R/qtl package.

Details

Vignettes online at the R/qtlcharts website:

- User guide
- Developer guide
- Use with R Markdown [Rmd source]
- List of chart customization options
- Examples of use with Shiny
- Examples of use with Jupyter notebooks

Author(s)

Maintainer: Karl W Broman <broman@wisc.edu> (ORCID)

Authors:

- Karl W Broman <broman@wisc.edu> (ORCID)

Other contributors:

- Michael Bostock (d3.js library in htmlwidgets/lib, https://d3js.org) [contributor, copyright holder]
- jQuery Foundation (jQuery library in htmlwidgets/lib, https://jquery.com) [copyright holder]
- jQuery contributors (jQuery library in htmlwidgets/lib; see https://github.com/jquery/jquery/blob/master/AUTHORS.txt [contributor]
- jQuery UI contributors (jQuery UI library in htmlwidgets/lib; see https://jqueryui.com/about/) [contributor]

See Also

Useful links:

- https://kbroman.org/qtlcharts/
- https://github.com/kbroman/qtlcharts
- Report bugs at https://github.com/kbroman/qtlcharts/issues

cbindQTLeffects *Combine multiple runs of estQTLeffects*

Description

Combine multiple runs of estQTLeffects by applying cbind to each component

Usage

```
cbindQTLeffects(..., labels)
```

Arguments

...	Results of estQTLeffects()
labels	Vector of labels to use in the combination.

Value

list of matrices; each component corresponds to a position in the genome and is a matrix with phenotypes x effects

See Also

estQTLeffects()

Examples

```
library(qtl)
data(fake.f2)
fake.f2 <- calc.genoprob(fake.f2)
sex <- fake.f2$pheno$sex
eff.fem <- estQTLeffects(fake.f2[,sex==0], pheno.col=1)
eff.mal <- estQTLeffects(fake.f2[,sex==1], pheno.col=1)
eff <- cbindQTLeffects(eff.fem, eff.mal, labels=c("female", "male"))
```

estQTLeffects *Calculate QTL effects at each position across the genome*

Description

Calculates the effects of QTL at each position across the genome using Haley-Knott regression, much like qtl::effectscan(), but considering multiple phenotypes and not plotting the results

Usage

```
estQTLeffects(cross, pheno.col = 1, what = c("means", "effects"))
```

Arguments

cross	(Optional) Object of class "cross", see qtl::read.cross().
pheno.col	Phenotype columns in cross object.
what	Indicates whether to calculate phenotype averages for each genotype group or to turn these into additive and dominance effects.

Details

One should first run qtl::calc.genoprob(); if not, it is run with the default arguments.

The estimated effects will be poorly estimated in the case of selective genotyping, as Haley-Knott regression performs poorly in this case.

Value

list of matrices; each component corresponds to a position in the genome and is a matrix with phenotypes x effects

See Also

iplotMScanone(), qtl::effectscan() cbindQTLeffects()

Examples

```
data(grav)
library(qtl)
grav <- reduce2grid(calc.genoprob(grav, step=1))
out <- estQTLeffects(grav, phe=seq(1, nphe(grav), by=5))
```

---

geneExpr *Anonymized gene expression data*

Description

An anonymized set of gene expression values, for 100 genes all influenced by a common locus, plus a vector of genotypes for the 491 individuals.

Usage

```
data(geneExpr)
```

Format

A list containing a matrix expr with the gene expression data plus a vector genotype with the genotypes.

Author(s)

Karl W Broman, 2013-05-16

Examples

```
data(geneExpr)

# heat map of correlation matrix, linked to scatterplots
iplotCorr(geneExpr$expr, geneExpr$genotype, reorder=TRUE)
```

---

grav *Arabidopsis QTL data on gravitropism*

Description

Data from a QTL experiment on gravitropism in Arabidopsis, with data on 162 recombinant inbred lines (Ler x Cvi). The outcome is the root tip angle (in degrees) at two-minute increments over eight hours.

Usage

```
data(grav)
```

Format

An object of class "cross"; see qtl::read.cross().

Source

Mouse Phenome Database, https://phenome.jax.org/projects/Moore1b

References

Moore CR, Johnson LS, Kwak I-Y, Livny M, Broman KW, Spalding EP (2013) High-throughput computer vision introduces the time axis to a quantitative trait map of a plant growth response. Genetics 195:1077-1086 doi:10.1534/genetics.113.153346

Examples

```
data(grav)
times <- attr(grav, "time")
phe <- grav$pheno

iplotCurves(phe, times, phe[,c(61,121)], phe[,c(121,181)],
        chartOpts=list(curves_xlab="Time (hours)", curves_ylab="Root tip angle (degrees)",
                       scat1_xlab="Angle at 2 hrs", scat1_ylab="Angle at 4 hrs",
                       scat2_xlab="Angle at 4 hrs", scat2_ylab="Angle at 6 hrs"))
```

`iboxplot` *Modern boxplot linked to underlying histrograms*

Description

Creates an interactive graph for a large set of box plots (rendered as lines connecting the quantiles), linked to underlying histograms.

Usage

```
iboxplot(
  dat,
  qu = c(0.001, 0.01, 0.1, 0.25),
  orderByMedian = TRUE,
  breaks = 251,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

`dat`	Data matrix (individuals x variables)
`qu`	Quantiles to plot (All with $0 < qu < 0.5$)
`orderByMedian`	If TRUE, reorder individuals by their median
`breaks`	Number of bins in the histograms, or a vector of locations of the breakpoints between bins (as in `graphics::hist()`)
`chartOpts`	A list of options for configuring the chart (see the coffeescript code). Each element must be named using the corresponding option.
`digits`	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

`iplotCorr()`, `scat2scat()`

Examples

```
n.ind <- 500
n.gene <- 10000
expr <- matrix(rnorm(n.ind * n.gene, (1:n.ind)/n.ind*3), ncol=n.gene)
dimnames(expr) <- list(paste0("ind", 1:n.ind),
```

```
                        paste0("gene", 1:n.gene))

iboxplot(expr, chartOpts=list(xlab="Mice", ylab="Gene expression"))
```

`idotplot` *Interactive phenotype x genotype plot*

Description

Creates an interactive graph of phenotypes vs genotypes at a marker.

Usage

```
idotplot(
  x,
  y,
  indID = NULL,
  group = NULL,
  horizontal = FALSE,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

`x`	Vector of groups of individuals (e.g., a genotype)
`y`	Numeric vector (e.g., a phenotype)
`indID`	Optional vector of character strings, shown with tool tips
`group`	Optional vector of categories for coloring points
`horizontal`	If TRUE, switch role of x- and y-axes and have quantitative variable on x-axis.
`chartOpts`	A list of options for configuring the chart. Each element must be named using the corresponding option.
`digits`	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

`iplot()`, `iplotPXG()`

Examples

```
n <- 100
g <- sample(LETTERS[1:3], n, replace=TRUE)
y <- rnorm(n, match(g, LETTERS[1:3])*10, 5)

idotplot(g, y)
idotplot(g, y, horizontal=TRUE)
```

---

iheatmap *Interactive heat map*

Description

Creates an interactive heatmap, with each cell linked to plots of horizontal and vertical slices

Usage

```
iheatmap(z, x = NULL, y = NULL, chartOpts = NULL, digits = 5)
```

Arguments

z	Numeric matrix of dim `length(x)` x `length(y)`
x	Vector of numeric values for the x-axis
y	Vector of numeric values for the y-axis
chartOpts	A list of options for configuring the chart (see the coffeescript code). Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Details

By default, the z-axis limits are from `-max(abs(z))` to `max(abs(z))`, and negative cells are colored blue to white which positive cells are colored white to red.

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

`iplotCorr()`

Examples

```
n <- 101
x <- y <- seq(-2, 2, len=n)
z <- matrix(ncol=n, nrow=n)
for(i in seq(along=x))
    for(j in seq(along=y))
        z[i,j] <- x[i]*y[j]*exp(-x[i]^2 - y[j]^2)

iheatmap(z, x, y)
```

`ipleiotropy` *Tool to explore pleiotropy*

Description

Creates an interactive graph of a scatterplot of two phenotypes, plus optionally the LOD curves for the two traits along one chromosome, with a slider for selecting the locations of two QTL which are then indicated on the LOD curves and the corresponding genotypes used to color the points in the scatterplot.

Usage

```
ipleiotropy(
  cross,
  scanoneOutput = NULL,
  pheno.col = 1:2,
  lodcolumn = 1:2,
  chr = NULL,
  interval = NULL,
  fillgenoArgs = NULL,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

`cross`	(Optional) Object of class `"cross"`, see `qtl::read.cross()`.
`scanoneOutput`	(Optional) object of class `"scanone"`, as output from `qtl::scanone()`.
`pheno.col`	Vector indicating two phenotype column in cross object; either numeric or character strings (the latter being the phenotype column names).
`lodcolumn`	Vector of two numeric values indicating LOD score columns to plot.
`chr`	A single chromosome ID, as a character string.
`interval`	A numeric vector of length 2, defining an interval that indicates what portion of the chromosome should be included.
`fillgenoArgs`	List of named arguments to pass to `qtl::fill.geno()`, if needed.

chartOpts	A list of options for configuring the chart (see the coffeescript code). Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Details

qtl::fill.geno() is used to impute missing genotypes. In this case, arguments to qtl::fill.geno() are passed as a list, for example fillgenoArgs=list(method="argmax", error.prob=0.002, map.function="c-f").

Individual IDs (viewable when hovering over a point in the scatterplot of the two phenotypes) are taken from the input cross object, using the qtl::getid() function in R/qtl.

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

iplotScanone(), iplotMScanone(), iplotPXG()

Examples

```
library(qtl)
data(fake.bc)
fake.bc <- calc.genoprob(fake.bc[5,], step=1) # select chr 5
out <- scanone(fake.bc, method="hk", pheno.col=1:2)

ipleiotropy(fake.bc, out)


# omit the LOD curves
ipleiotropy(fake.bc)
```

---

iplot *Interactive scatterplot*

Description

Creates an interactive scatterplot.

Usage

```
iplot(x, y = NULL, group = NULL, indID = NULL, chartOpts = NULL, digits = 5)
```

Arguments

x	Numeric vector of x values
y	Numeric vector of y values (If NULL, we take the x values as y, and use 1:length(x) for x.)
group	Optional vector of categories for coloring the points
indID	Optional vector of character strings, shown with tool tips
chartOpts	A list of options for configuring the chart. Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

iplotCorr(), iplotCurves(), itriplot(), idotplot(), iplotPXG()

Examples

```
x <- rnorm(100)
grp <- sample(1:3, 100, replace=TRUE)
y <- x*grp + rnorm(100)

iplot(x, y, grp)
```

---

iplotCorr *Image of correlation matrix with linked scatterplot*

Description

Creates an interactive graph with an image of a correlation matrix linked to underlying scatterplots.

Usage

```
iplotCorr(
  mat,
  group = NULL,
  rows = NULL,
  cols = NULL,
  reorder = FALSE,
  corr = NULL,
  scatterplots = TRUE,
```

```
  chartOpts = NULL,
  digits = 5
)
```

Arguments

mat	Data matrix (individuals x variables)
group	Optional vector of groups of individuals (e.g., a genotype)
rows	Selected rows of the correlation matrix to include in the image. Ignored if corr is provided.
cols	Selected columns of the correlation matrix to include in the image. Ignored if corr is provided.
reorder	If TRUE, reorder the variables by clustering. Ignored if corr is provided as a subset of the overall correlation matrix
corr	Correlation matrix (optional).
scatterplots	If FALSE, don't have the heat map be linked to scatterplots.
chartOpts	A list of options for configuring the chart (see the coffeescript code). Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Details

corr may be provided as a subset of the overall correlation matrix for the columns of mat. In this case, the reorder, rows and cols arguments are ignored. The row and column names of corr must match the names of some subset of columns of mat.

Individual IDs are taken from rownames(mat); they must match names(group).

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

iheatmap(), scat2scat(), iplotCurves()

Examples

```
data(geneExpr)

iplotCorr(geneExpr$expr, geneExpr$genotype, reorder=TRUE,
          chartOpts=list(cortitle="Correlation matrix",
                         scattitle="Scatterplot"))

# use Spearman's correlation
corr <- cor(geneExpr$expr, method="spearman", use="pairwise.complete.obs")
```

```
# order by hierarchical clustering
o <- hclust(as.dist(1-corr))$order

iplotCorr(geneExpr$expr[,o], geneExpr$genotype, corr=corr[o,o],
          chartOpts=list(cortitle="Spearman correlation",
                         scattitle="Scatterplot"))
```

`iplotCurves` *Plot of a bunch of curves, linked to points in scatterplots*

Description

Creates an interactive graph with a panel having a number of curves (say, a phenotype measured over time) linked to one or two (or no) scatter plots (say, of the first vs middle and middle vs last times).

Usage

```
iplotCurves(
  curveMatrix,
  times = NULL,
  scatter1 = NULL,
  scatter2 = NULL,
  group = NULL,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

`curveMatrix`	Matrix (dim n_ind x n_times) with outcomes
`times`	Vector (length n_times) with time points for the columns of curveMatrix
`scatter1`	Matrix (dim n_ind x 2) with data for the first scatterplot
`scatter2`	Matrix (dim n_ind x 2) with data for the second scatterplot
`group`	Optional vector of groups of individuals (e.g., a genotype)
`chartOpts`	A list of options for configuring the chart (see the coffeescript code). Each element must be named using the corresponding option.
`digits`	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

iplotCorr(), iplot(), scat2scat()

Examples

```
# random growth curves, based on some data
times <- 1:16
n <- 100
start <- rnorm(n, 5.2, 0.8)
slope1to5 <- rnorm(n, 2.6, 0.5)
slope5to16 <- rnorm(n, 0.24 + 0.09*slope1to5, 0.195)
y <- matrix(ncol=16, nrow=n)
y[,1] <- start
for(j in 2:5)
    y[,j] <- y[,j-1] + slope1to5
for(j in 6:16)
    y[,j] <- y[,j-1] + slope5to16
y <- y + rnorm(prod(dim(y)), 0, 0.35)


iplotCurves(y, times, y[,c(1,5)], y[,c(5,16)],
            chartOpts=list(curves_xlab="Time", curves_ylab="Size",
                           scat1_xlab="Size at T=1", scat1_ylab="Size at T=5",
                           scat2_xlab="Size at T=5", scat2_ylab="Size at T=16"))
```

---

iplotMap *Interactive genetic map plot*

Description

Creates an interactive graph of a genetic marker map.

Usage

```
iplotMap(
  map,
  chr = NULL,
  shift = FALSE,
  horizontal = FALSE,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

map	Object of class "map", a list with each component being a vector of marker positions. You can also provide an object of class "cross", in which case the map is extracted with qtl::pull.map().

chr	(Optional) Vector indicating the chromosomes to plot.
shift	If TRUE, shift each chromsome so that the initial marker is at position 0.
horizontal	If TRUE, have chromosomes arranged horizontally
chartOpts	A list of options for configuring the chart. Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

`iplotScanone()`, `iplotPXG()`

Examples

```
library(qtl)
data(hyper)
map <- pull.map(hyper)

iplotMap(map, shift=TRUE)
```

---

iplotMScanone *Interactive LOD curve*

Description

Creates an interactive graph of a set of single-QTL genome scans, as calculated by `qtl::scanone()`. If `cross` or `effects` are provided, LOD curves will be linked to a panel with estimated QTL effects.

Usage

```
iplotMScanone(
  scanoneOutput,
  cross = NULL,
  lodcolumn = NULL,
  pheno.col = NULL,
  times = NULL,
  effects = NULL,
  chr = NULL,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

scanoneOutput	Object of class "scanone", as output from qtl::scanone().
cross	(Optional) Object of class "cross", see qtl::read.cross().
lodcolumn	Numeric value indicating LOD score column to plot.
pheno.col	(Optional) Phenotype column in cross object.
times	(Optional) Vector (length equal to the number of LOD score columns) with quantitative values to which the different LOD score columns correspond (times of measurements, or something like age or dose). These need to be ordered and equally-spaced. If omitted, the names of the columns in scanoneOutput are used and treated as qualitative.
effects	(Optional) Estimated QTL effects, as obtained with estQTLeffects().
chr	(Optional) Optional vector indicating the chromosomes for which LOD scores should be calculated. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
chartOpts	A list of options for configuring the chart (see the coffeescript code). Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Details

If cross is provided, Haley-Knott regression is used to estimate QTL effects at each pseudomarker.

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

iplotScanone()

Examples

```
data(grav)
library(qtl)
grav <- calc.genoprob(grav, step=1)
grav <- reduce2grid(grav)

# we're going to subset the phenotypes
phecol <- seq(1, nphe(grav), by=5)

# the times were saved as an attributed
times <- attr(grav, "time")[phecol]
```

```
# genome scan
out <- scanone(grav, phe=phecol, method="hk")


# plot with qualitative labels on y-axis
iplotMScanone(out)


# plot with quantitative y-axis
iplotMScanone(out, times=times)

# estimate QTL effect for each time point at each genomic position
eff <- estQTLeffects(grav, phe=seq(1, nphe(grav), by=5), what="effects")


# plot with QTL effects included (and with quantitative y-axis)
iplotMScanone(out, effects=eff, times=times,
              chartOpts=list(ylab="Time (hrs)", eff_ylab="QTL effect"))
```

`iplotPXG` *Interactive phenotype x genotype plot*

Description

Creates an interactive graph of phenotypes vs genotypes at a marker.

Usage

```
iplotPXG(
  cross,
  marker,
  pheno.col = 1,
  horizontal = FALSE,
  chartOpts = NULL,
  fillgenoArgs = NULL,
  digits = 5
)
```

Arguments

`cross`	Object of class `"cross"`, see `qtl::read.cross()`.
`marker`	Character string with marker name.
`pheno.col`	Phenotype column in cross object.
`horizontal`	If TRUE, switch role of x- and y-axes and put genotype on y-axis.
`chartOpts`	A list of options for configuring the chart. Each element must be named using the corresponding option.

fillgenoArgs	List of named arguments to pass to qtl::fill.geno(), if needed.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Details

The function qtl::fill.geno() is used to impute missing genotypes, with arguments passed as a list, for example fillgenoArgs=list(method="argmax", error.prob=0.002, map.function="c-f").

Individual IDs (viewable when hovering over a point) are taken from the input cross object, using the qtl::getid() function in R/qtl.

By default, points are colored blue and pink according to whether the marker genotype is observed or inferred, respectively.

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

idotplot(), iplot(), iplotScanone(), iplotMap()

Examples

```
library(qtl)
data(hyper)
marker <- sample(markernames(hyper), 1)

iplotPXG(hyper, marker)

# different colors
iplotPXG(hyper, marker, chartOpts=list(pointcolor=c("black", "gray")))
```

---

iplotRF *Interactive plot of recombination fractions*

Description

Creates an interactive graph of estimated recombination fractions and LOD scores for all pairs of markers.

Usage

```
iplotRF(cross, chr = NULL, chartOpts = NULL, digits = 5)
```

Arguments

cross	Object of class "cross", see qtl::read.cross().
chr	Optional vector indicating chromosomes to include. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
chartOpts	A list of options for configuring the chart. Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Details

The heatmap in the plot just depicts the LOD scores for testing rf=0.5.

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

qtl::est.rf(), qtl::plotRF()

Examples

```
library(qtl)
data(fake.f2)

fake.f2 <- est.rf(fake.f2)

iplotRF(fake.f2)
```

---

iplotScanone *Interactive LOD curve*

Description

Creates an interactive graph of a single-QTL genome scan, as calculated by qtl::scanone(). If cross is provided, the LOD curves are linked to a phenotype x genotype plot for a marker: Click on a marker on the LOD curve and see the corresponding phenotype x genotype plot.

Usage

```
iplotScanone(
  scanoneOutput,
  cross = NULL,
  lodcolumn = 1,
  pheno.col = 1,
  chr = NULL,
  pxgtype = c("ci", "raw"),
  fillgenoArgs = NULL,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

scanoneOutput	Object of class "scanone", as output from qtl::scanone().
cross	(Optional) Object of class "cross", see qtl::read.cross().
lodcolumn	Numeric value indicating LOD score column to plot.
pheno.col	(Optional) Phenotype column in cross object.
chr	(Optional) Vector indicating the chromosomes for which LOD scores should be calculated. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
pxgtype	If phenotype x genotype plot is to be shown, should it be with means $\pm$ 2 SE ("ci"), or raw phenotypes ("raw")?
fillgenoArgs	List of named arguments to pass to qtl::fill.geno(), if needed.
chartOpts	A list of options for configuring the chart (see the coffeescript code). Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Details

If cross is provided, qtl::fill.geno() is used to impute missing genotypes. In this case, arguments to qtl::fill.geno() are passed as a list, for example fillgenoArgs=list(method="argmax", error.prob=0.002, map.function="c-f").

With pxgtype="raw", individual IDs (viewable when hovering over a point in the phenotype-by-genotype plot) are taken from the input cross object, using the qtl::getid() function in R/qtl.

Value

An object of class htmlwidget that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

iplotMScanone(), iplotPXG(), iplotMap()

Examples

```
library(qtl)
data(hyper)
hyper <- calc.genoprob(hyper, step=1)
out <- scanone(hyper)

# iplotScanone with no effects
iplotScanone(out, chr=c(1, 4, 6, 7, 15))


# iplotScanone with CIs
iplotScanone(out, hyper, chr=c(1, 4, 6, 7, 15))


# iplotScanone with raw phe x gen
iplotScanone(out, hyper, chr=c(1, 4, 6, 7, 15),
             pxgtype='raw')
```

---

iplotScantwo *Interactive plot of 2d genome scan*

Description

Creates an interactive plot of the results of qtl::scantwo(), for a two-dimensional, two-QTL genome scan.

Usage

```
iplotScantwo(
  scantwoOutput,
  cross = NULL,
  lodcolumn = 1,
  pheno.col = 1,
  chr = NULL,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

scantwoOutput	Output of qtl::scantwo()
cross	(Optional) Object of class "cross", see qtl::read.cross().
lodcolumn	Numeric value indicating LOD score column to plot.

pheno.col	(Optional) Phenotype column in cross object.
chr	(Optional) Optional vector indicating the chromosomes for which LOD scores should be calculated. This should be a vector of character strings referring to chromosomes by name; numeric values are converted to strings. Refer to chromosomes with a preceding - to have all chromosomes but those considered. A logical (TRUE/FALSE) vector may also be used.
chartOpts	A list of options for configuring the chart. Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Details

The estimated QTL effects, and the genotypes in the phenotype x genotype plot, in the right-hand panels, are derived following a single imputation to fill in missing data, and so are a bit crude.

Note that the usual `height` and `width` options in `chartOpts` are ignored here. Instead, you may provide `pixelPerCell` (number of pixels per cell in the heat map), `chrGap` (gaps between chr in heat map), `wright` (width in pixels of right panels), and `hbot` (height in pixels of each of the lower panels)

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

`iplotScanone()`

Examples

```
library(qtl)
data(fake.f2)

fake.f2 <- calc.genoprob(fake.f2, step=5)
out <- scantwo(fake.f2, method="hk", verbose=FALSE)

iplotScantwo(out, fake.f2)
```

---

itimeplot *Interactive scatterplot with date/time on x-axis*

Description

Creates an interactive scatterplot with date/time on the x-axis.

Usage

```
itimeplot(
  datetime,
  y,
  group = NULL,
  indID = NULL,
  chartOpts = NULL,
  digits = 5
)
```

Arguments

datetime	Vector of date/time values
y	Numeric vector of y values
group	Optional vector of categories for coloring the points
indID	Optional vector of character strings, shown with tool tips
chartOpts	A list of options for configuring the chart. Each element must be named using the corresponding option.
digits	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

`iplotCorr()`, `iplotCurves()`, `itriplot()`, `idotplot()`, `iplotPXG()`

Examples

```
n_pts <- 100
x <-  seq(as.POSIXct("1969-05-01"), as.POSIXct("1969-05-04"), length=n_pts)
grp <- sample(1:3, n_pts, replace=TRUE)
y <- rnorm(n_pts, grp) + rnorm(n_pts)

itimeplot(x, y, grp)


# with shorter time span, times are shown instead of dates
x <-  seq(as.POSIXct("1969-05-01"), as.POSIXct("1969-05-02"), length=n_pts)

itimeplot(x, y, grp, chartOpts=list(xlab="Time"))

#'
```

`itriplot` *Interactive plot of trinomial probabilities*

Description

Creates an interactive graph of trinomial probabilities, represented as points within an equilateral triangle.

Usage

```
itriplot(p, indID = NULL, group = NULL, chartOpts = NULL, digits = 5)
```

Arguments

`p`	Matrix of trinomial probabilities (n x 3); each row should sum to 1.
`indID`	Optional vector of character strings, shown with tool tips
`group`	Optional vector of categories for coloring the points
`chartOpts`	A list of options for configuring the chart. Each element must be named using the corresponding option.
`digits`	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

`iplot()`, `iplotPXG()`, `idotplot()`

Examples

```
n <- 100
p <- matrix(runif(3*n), ncol=3)
p <- p / colSums(p)
g <- sample(1:3, n, replace=TRUE)

itriplot(p, group=g)
```

qtlcharts-shiny *Shiny bindings for R/qtlcharts widgets*

Description

Output and render functions for using R/qtlcharts widgets within Shiny applications and interactive Rmd documents. For examples, see https://github.com/kbroman/shiny_qtlcharts.

Usage

```
iboxplot_output(outputId, width = "100%", height = "900")

iboxplot_render(expr, env = parent.frame(), quoted = FALSE)

idotplot_output(outputId, width = "100%", height = "530")

idotplot_render(expr, env = parent.frame(), quoted = FALSE)

iheatmap_output(outputId, width = "100%", height = "1000")

iheatmap_render(expr, env = parent.frame(), quoted = FALSE)

ipleiotropy_output(outputId, width = "100%", height = "580")

ipleiotropy_render(expr, env = parent.frame(), quoted = FALSE)

iplot_output(outputId, width = "100%", height = "580")

iplot_render(expr, env = parent.frame(), quoted = FALSE)

iplotCorr_output(outputId, width = "100%", height = "1000")

iplotCorr_render(expr, env = parent.frame(), quoted = FALSE)

iplotCurves_output(outputId, width = "100%", height = "1000")

iplotCurves_render(expr, env = parent.frame(), quoted = FALSE)

iplotMScanone_output(outputId, width = "100%", height = "580")

iplotMScanone_render(expr, env = parent.frame(), quoted = FALSE)

iplotMap_output(outputId, width = "100%", height = "680")

iplotMap_render(expr, env = parent.frame(), quoted = FALSE)

iplotRF_output(outputId, width = "100%", height = "1000")
```

```
iplotRF_render(expr, env = parent.frame(), quoted = FALSE)

iplotScanone_output(outputId, width = "100%", height = "580")

iplotScanone_render(expr, env = parent.frame(), quoted = FALSE)

iplotScantwo_output(outputId, width = "100%", height = "1000")

iplotScantwo_render(expr, env = parent.frame(), quoted = FALSE)

itimeplot_output(outputId, width = "100%", height = "580")

itimeplot_render(expr, env = parent.frame(), quoted = FALSE)

itriplot_output(outputId, width = "100%", height = "530")

itriplot_render(expr, env = parent.frame(), quoted = FALSE)

scat2scat_output(outputId, width = "100%", height = "530")

scat2scat_render(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like "100\%", "400px", "auto") or a number, which will be coerced to a string and have "px" appended.
expr	An expression that generates the graph.
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

---

qtlchartsversion *print the installed version of R/qtlcharts*

Description

print the installed version of R/qtlcharts

Usage

```
qtlchartsversion()
```

Value

Character string with version number

Examples

```
qtlchartsversion()
```

`scat2scat` *Scatterplot driving another scatterplot*

Description

A pair of linked scatterplots, where each point the first scatterplot corresponds to a scatter of points in the second scatterplot. The first scatterplot corresponds to a pair of summary measures for a larger dataset.

Usage

```
scat2scat(scat1data, scat2data, group = NULL, chartOpts = NULL, digits = 5)
```

Arguments

`scat1data`	Matrix with two columns; rownames are used as identifiers. Can have an optional third column with categories for coloring points in the first scatterplot (to be used if `group` is not provided).
`scat2data`	List of matrices each with at least two columns, to be shown in the second scatterplot. The components of the list correspond to the rows in `scat1dat`. An option third column can contain categories. Row names identify individual points.
`group`	Categories for coloring points in the first scatterplot; length should be the number of rows in `scat1data`.
`chartOpts`	A list of options for configuring the chart. Each element must be named using the corresponding option.
`digits`	Round data to this number of significant digits before passing to the chart function. (Use NULL to not round.)

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

`iplotCorr()`

Examples

```
# simulate some data
p <- 500
n <- 300
SD <- runif(p, 1, 5)
r <- runif(p, -1, 1)
scat2 <- vector("list", p)
for(i in 1:p)
  scat2[[i]] <- matrix(rnorm(2*n), ncol=2) %*% chol(SD[i]^2*matrix(c(1, r[i], r[i], 1), ncol=2))
scat1 <- cbind(SD=SD, r=r)
# plot it
scat2scat(scat1, scat2)
```

---

setScreenSize *Set default maximum screen size*

Description

Set the default screen size as a global option.

Usage

```
setScreenSize(size = c("normal", "small", "large"), height, width)
```

Arguments

size	Character vector representing screen size (normal, small, large). Ignored if height and width are provided.
height	(Optional) Height in pixels
width	(Optional) Width in pixels

Details

Used to set a global option, `qtlchartsScreenSize`, that contains the maximum height and maximum width for a chart in the browser.

`"small"`, `"normal"`, and `"large"` correspond to 600x900, 700x1000, and 1200x1600, for height x width, respectively.

Value

None.

Examples

```
setScreenSize("large")
```

Index

∗ **datasets**
geneExpr, 5
grav, 6
∗ **hplot**
iboxplot, 7
idotplot, 8
iheatmap, 9
ipleiotropy, 10
iplot, 11
iplotCorr, 12
iplotCurves, 14
iplotMap, 15
iplotMScanone, 16
iplotPXG, 18
iplotRF, 19
iplotScanone, 20
iplotScantwo, 22
itimeplot, 23
itriplot, 25
scat2scat, 28
∗ **regression**
estQTLeffects, 4
∗ **utilities**
cbindQTLeffects, 4
setScreenSize, 29

cbindQTLeffects, 4
cbindQTLeffects(), *5*

estQTLeffects, 4
estQTLeffects(), *4*, *17*

geneExpr, 5
graphics::hist(), *7*
grav, 6

iboxplot, 7
iboxplot_output *(*qtlcharts-shiny*)*, 26
iboxplot_render *(*qtlcharts-shiny*)*, 26
idotplot, 8
idotplot(), *12*, *19*, *24*, *25*
idotplot_output *(*qtlcharts-shiny*)*, 26
idotplot_render *(*qtlcharts-shiny*)*, 26
iheatmap, 9
iheatmap(), *13*
iheatmap_output *(*qtlcharts-shiny*)*, 26
iheatmap_render *(*qtlcharts-shiny*)*, 26
ipleiotropy, 10
ipleiotropy_output *(*qtlcharts-shiny*)*, 26
ipleiotropy_render *(*qtlcharts-shiny*)*, 26
iplot, 11
iplot(), *8*, *15*, *19*, *25*
iplot_output *(*qtlcharts-shiny*)*, 26
iplot_render *(*qtlcharts-shiny*)*, 26
iplotCorr, 12
iplotCorr(), *7*, *9*, *12*, *15*, *24*, *28*
iplotCorr_output *(*qtlcharts-shiny*)*, 26
iplotCorr_render *(*qtlcharts-shiny*)*, 26
iplotCurves, 14
iplotCurves(), *12*, *13*, *24*
iplotCurves_output *(*qtlcharts-shiny*)*, 26
iplotCurves_render *(*qtlcharts-shiny*)*, 26
iplotMap, 15
iplotMap(), *19*, *22*
iplotMap_output *(*qtlcharts-shiny*)*, 26
iplotMap_render *(*qtlcharts-shiny*)*, 26
iplotMScanone, 16
iplotMScanone(), *5*, *11*, *22*
iplotMScanone_output *(*qtlcharts-shiny*)*, 26
iplotMScanone_render *(*qtlcharts-shiny*)*, 26
iplotPXG, 18
iplotPXG(), *8*, *11*, *12*, *16*, *22*, *24*, *25*
iplotRF, 19
iplotRF_output *(*qtlcharts-shiny*)*, 26
iplotRF_render *(*qtlcharts-shiny*)*, 26
iplotScanone, 20
iplotScanone(), *11*, *16*, *17*, *19*, *23*

iplotScanone_output *(qtlcharts-shiny)*, 26
iplotScanone_render *(qtlcharts-shiny)*, 26
iplotScantwo, 22
iplotScantwo_output *(qtlcharts-shiny)*, 26
iplotScantwo_render *(qtlcharts-shiny)*, 26
itimeplot, 23
itimeplot_output *(qtlcharts-shiny)*, 26
itimeplot_render *(qtlcharts-shiny)*, 26
itriplot, 25
itriplot(), *12*, *24*
itriplot_output *(qtlcharts-shiny)*, 26
itriplot_render *(qtlcharts-shiny)*, 26

qtl::calc.genoprob(), *5*
qtl::effectscan(), *4, 5*
qtl::est.rf(), *20*
qtl::fill.geno(), *10, 11*, *19*, *21*
qtl::getid(), *11*, *19*, *21*
qtl::plotRF(), *20*
qtl::pull.map(), *15*
qtl::read.cross(), *5, 6*, *10*, *17, 18*, *20–22*
qtl::scanone(), *10*, *16, 17*, *20, 21*
qtl::scantwo(), *22*
qtlcharts *(qtlcharts-package)*, 2
qtlcharts-package, 2
qtlcharts-shiny, 26
qtlchartsversion, 27

scat2scat, 28
scat2scat(), *7*, *13*, *15*
scat2scat_output *(qtlcharts-shiny)*, 26
scat2scat_render *(qtlcharts-shiny)*, 26
setScreenSize, 29