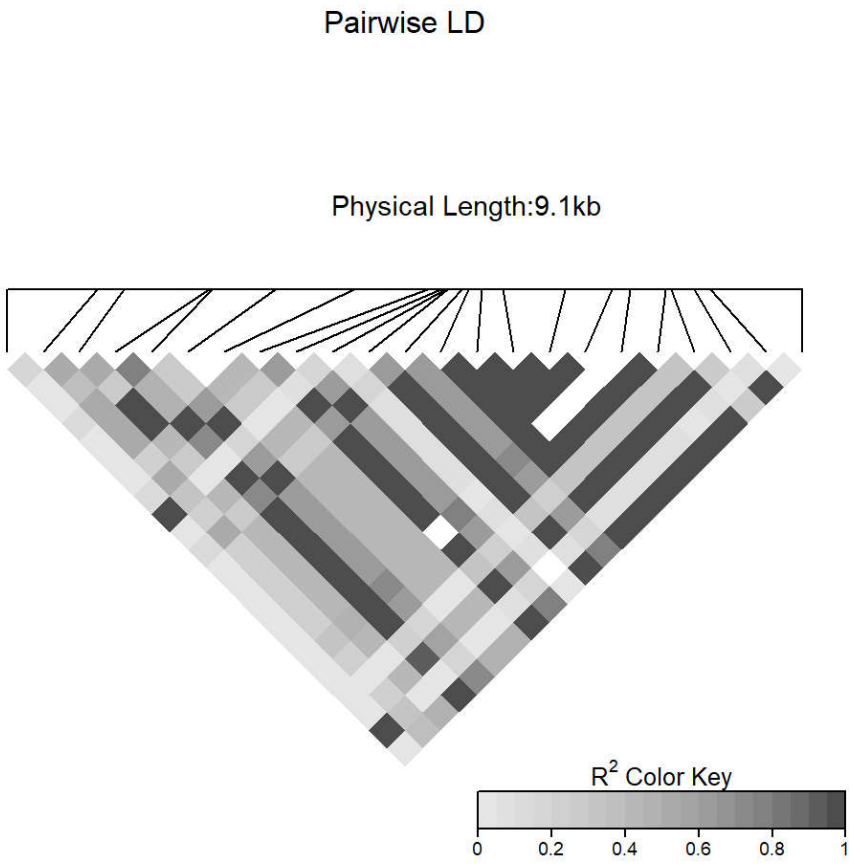
Bug fix documentation

Matthew and Gloria

2018-07-10

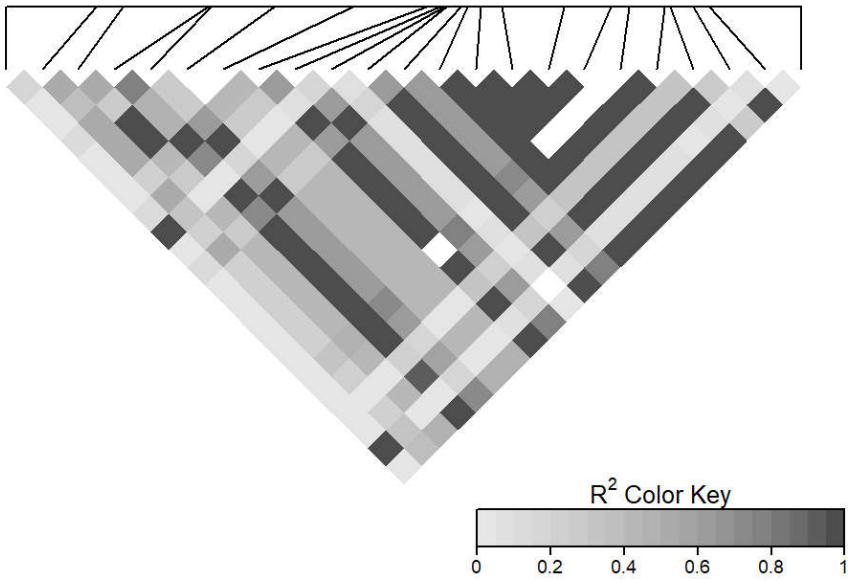
This document will discuss the bugs experienced by users of the LDheatmap package, the code to replicate the bugs, and the implemented solutions. The aim is to provide a concise walk through of the process and to highlight any need for further changes.

The following is the sample code provided in the Vignette, found here: https://cran.r-project.org/web/packages/LDheatmap/vignettes/addTracks.pdf (https://cran.r-project.org/web/packages/LDheatmap/vignettes/addTracks.pdf)



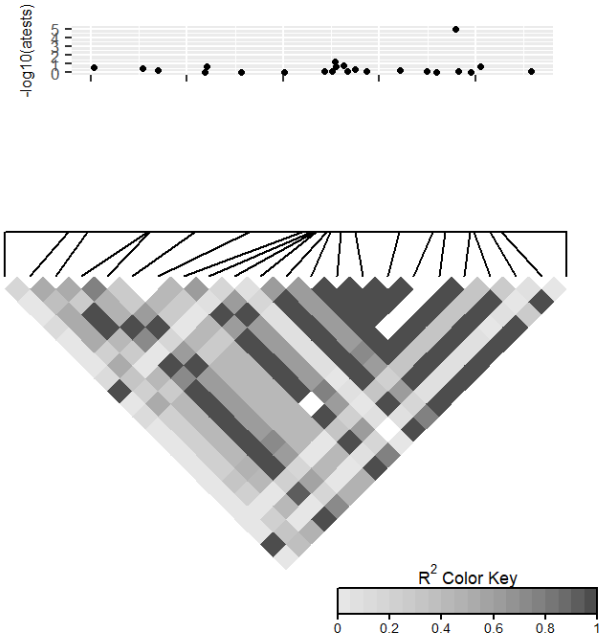
Pairwise LD

Physical Length:9.1kb



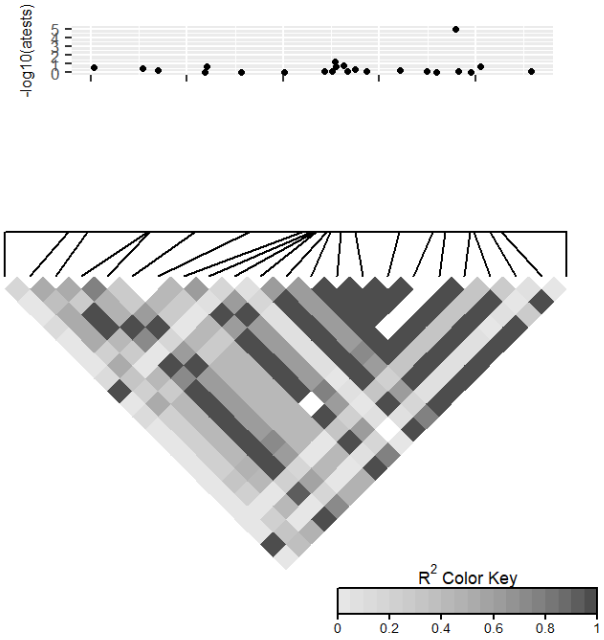
Pairwise LD

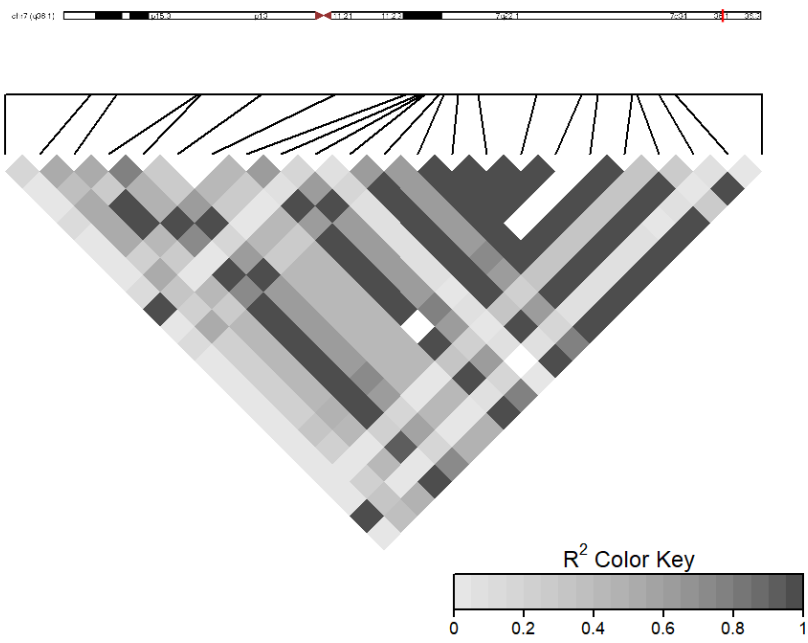
Physical Length:9.1kb



Pairwise LD

Physical Length:9.1kb





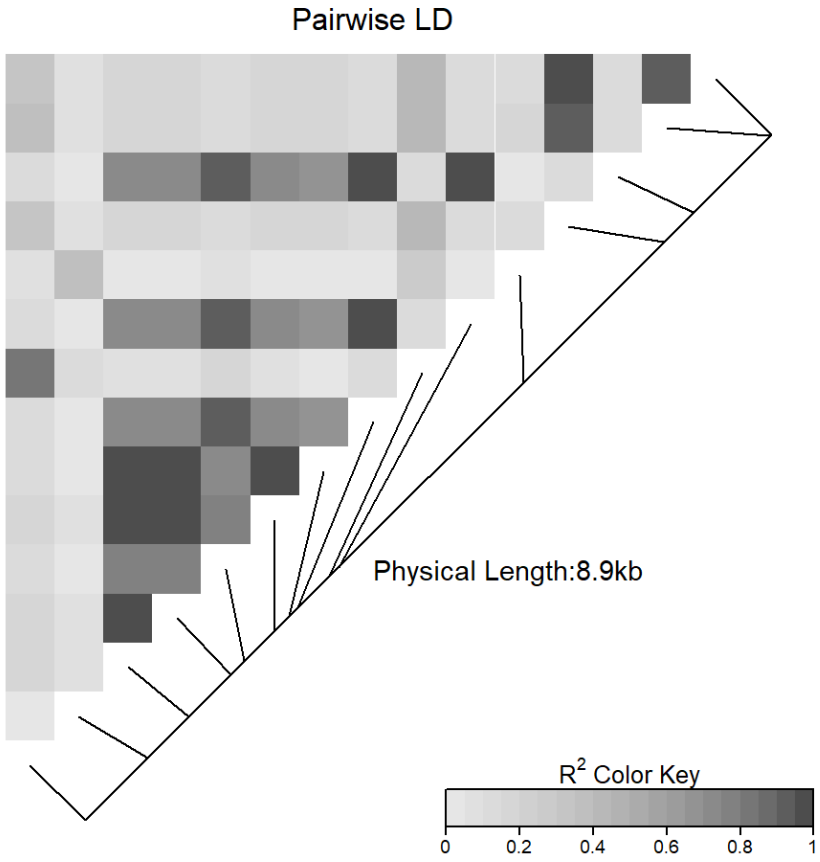
As LDheatmap has been used in many different papers and works over the years, users have experienced corner case behaviour beyond what is laid out in the original vignettes. The general core of the package is functional and handles most necessary functions adequately. We were tasked with evaluating whether the package needed an overhaul or if bug fixing would be sufficient.

Exploration of the bugs found that one component of the package often triggered corner case behaviour. The flipping of the heatmap, done by rotating a viewport, caused a collection of bugs. Upon review, the bugs appear to have been oversights or small missteps in the code. What follows is a discussion of each bug, the code that generates the undesired output (if applicable), and the suggested fix along with an example of how it solves the problem.

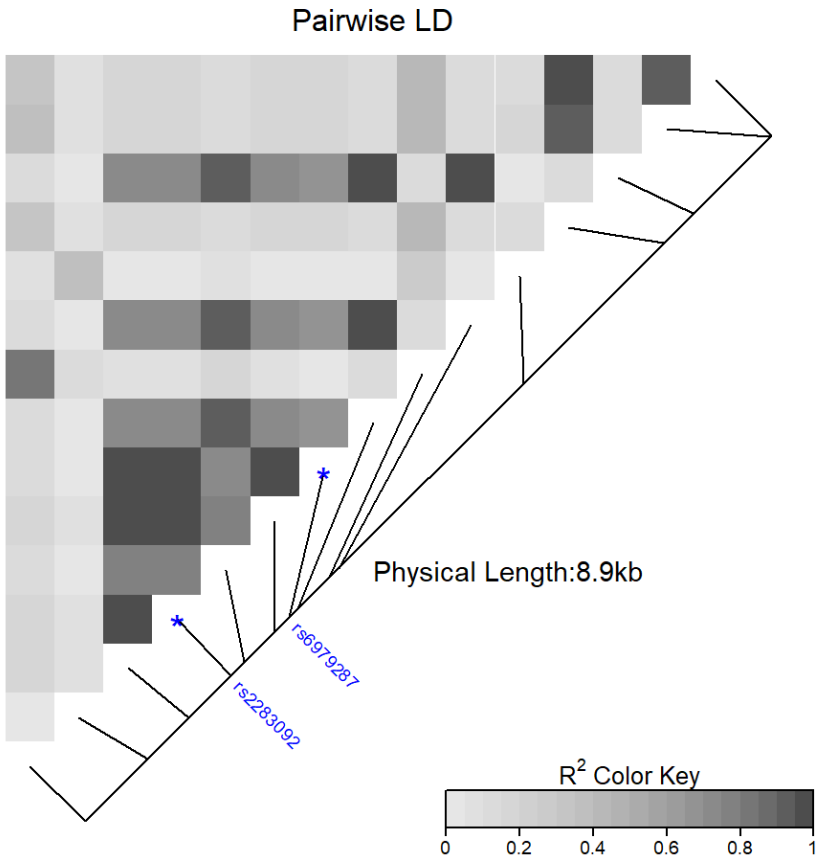
Bug Report 1: When the heatmap is flipped, symbols used to mark given SNPs in the genome are removed. This behaviour does not occur in the non-flipped version.

Example of the bug:

```
# Bug report: "Symbols" are gone when LDheatmap is generated with flip = TRUE
# Fix: LDheatmapMap.add() had a code section only if(flip) that set symbols <- NULL.
    # Resolved by using same assignment as was used in the non-flipped case
#require('LDheatmap')

 # taken from examples LDheatmap\demo\LDheatmap.R, works correctly
data(CEUData)
MyHeatmap <- LDheatmap(CEUSNP, genetic.distances = CEUDist,
                          color = grey.colors(20))
```



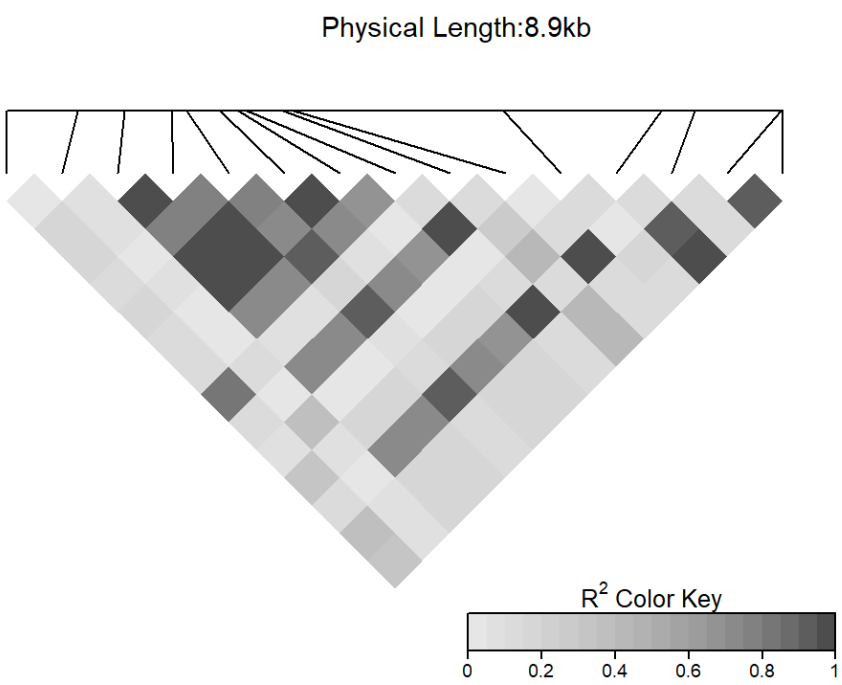

```
LDheatmap(MyHeatmap, SNP.name = c("rs2283092", "rs6979287"))
```



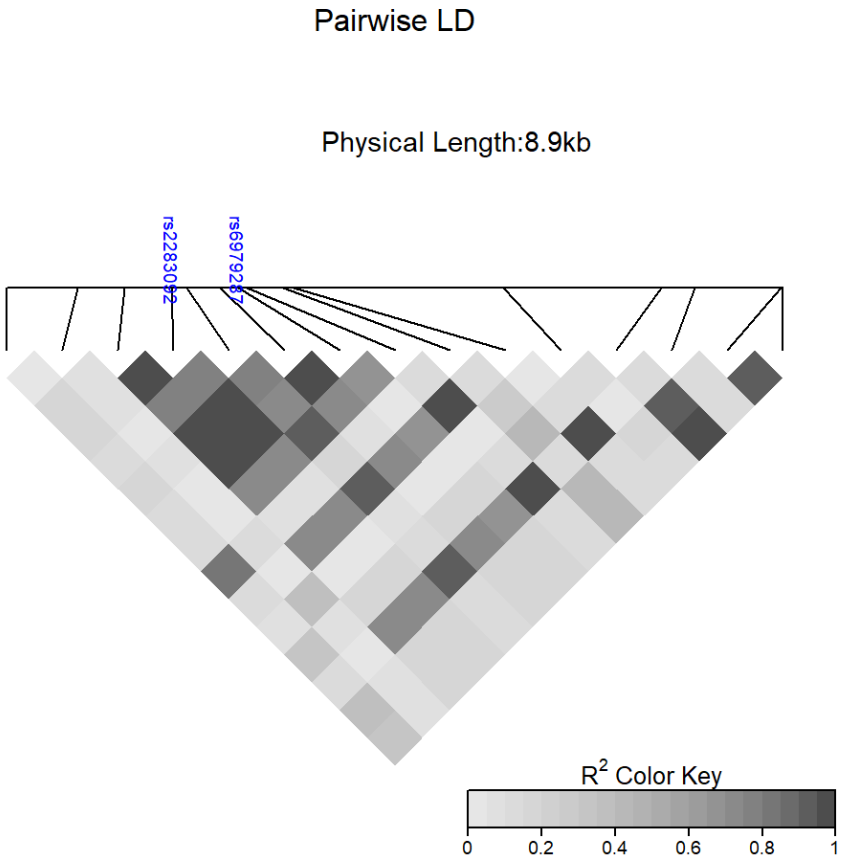

```
childNames(grid.get("geneMap"))
```

```
## [1] "diagonal" "segments" "title"    "symbols"  "SNPnames"
```

```
 #BUG: With flip=TRUE it does not work anymore
  MyHeatmapF <- LDheatmap(CEUSNP, genetic.distances = CEUDist,
                          color = grey.colors(20), flip=TRUE)
```



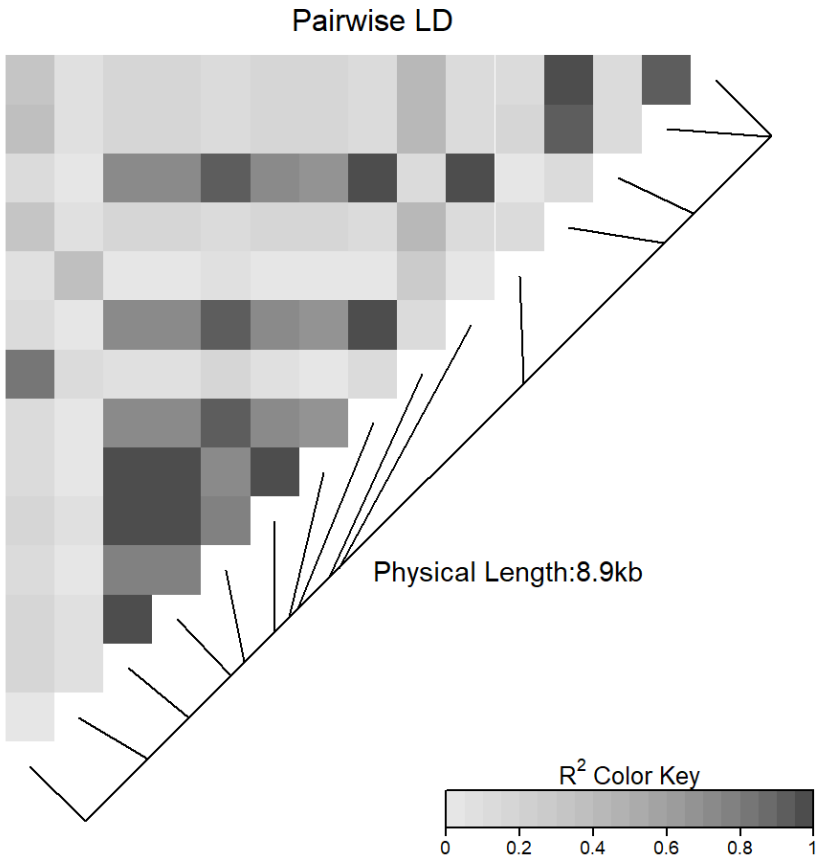

```
LDheatmap(MyHeatmapF, SNP.name = c("rs2283092", "rs6979287"))
```



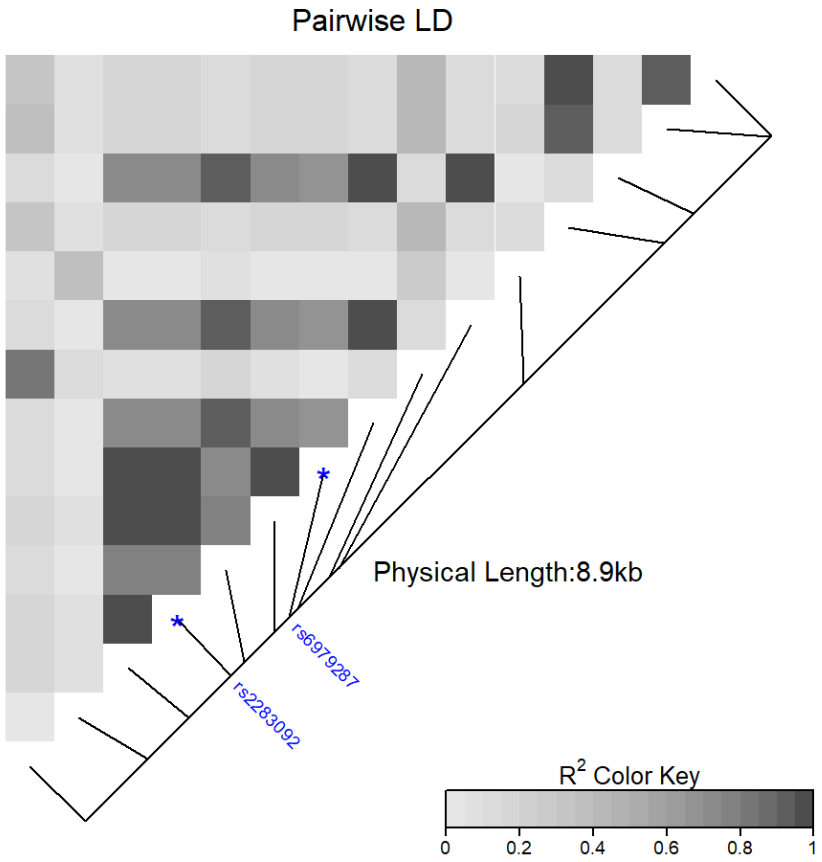

```
  childNames(grid.get("geneMap"))
```

```
## [1] "diagonal" "segments" "title"    "SNPnames"
```

```
  MyHeatmapTest <- LDheatmap(CEUSNP, genetic.distances = CEUDist,
                          color = grey.colors(20), flip=FALSE)
```




```
LDheatmap(MyHeatmapTest, SNP.name = c("rs2283092", "rs6979287"))
```




```
  childNames(grid.get("geneMap"))
```

```
## [1] "diagonal" "segments" "title"    "symbols"  "SNPnames"
```

The result of the code above shows the flipped graph, when specific SNP names are given, fails to output the symbols requested. This is further noted as the childNames() call to the flipped graph's genemap does not display 'symbols' as one of the children. Other uses of childNames() on non-flipped graphs return 'symbols' as a child of the genemap.

The bug-producing code can be found in the LDheatmapMap.add() function from this package. Below is the segment that causes the undesired behaviour.

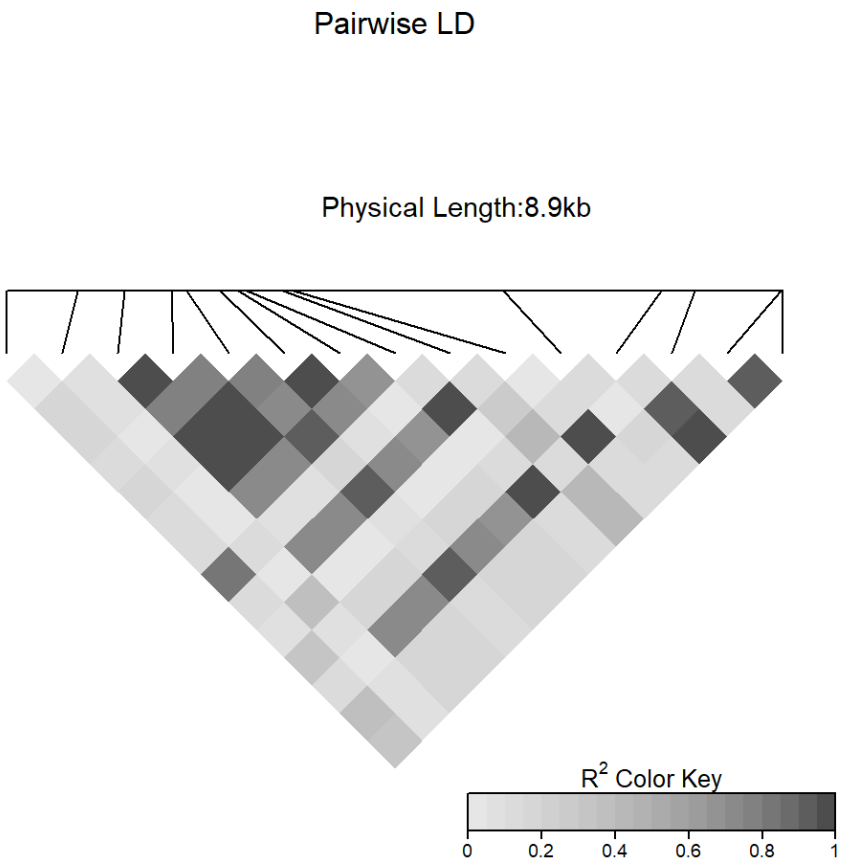
```
## Labelling some SNPs
placeholder <- function(nsnps, add.map, genetic.distances,
                     geneMapLocation=0.15,
                     geneMapLabelX=NULL, geneMapLabelY=NULL,
                     distances="physical", vp=NULL,
                     SNP.name=NULL, ind=0, flip=FALSE){
  # # # # # #
  if (!is.null(SNP.name) && (any(ind!=0))){
    symbols <- pointsGrob(snp[ind], snp[ind], pch="*",
              gp=gpar(cex=1.25, bg="blue", col="blue"), name="symbols", vp=vp)
    SNPnames <- textGrob(paste(" ", SNP.name), just="left", rot=-45,
            regionx[ind], regiony[ind], gp=gpar(cex=0.6, col="blue"), name="SNPname
s", vp=vp)
    if (flip) {
      lenght_SNP_name <- max(nchar(SNP.name))
      long_SNP_name <- paste(rep(8,lenght_SNP_name), collapse="")
      name_gap <- convertWidth(grobWidth(textGrob(long_SNP_name)), "npc",valueOnly=TRU
E)/sqrt(2)
      diagonal<-linesGrob(seq.x, seq.y, gp=gpar(lty=1), name="diagonal", vp=vp)
      #diagonal<-linesGrob(seq.x+name_gap, seq.y-name_gap, gp=gpar(lty=1), name="diago
nal", vp=vp)
      segments <- segmentsGrob(snp, snp, regionx, regiony, name="segments", vp=vp)
      #segments <- segmentsGrob(snp+name_gap, snp-name_gap, regionx+name_gap, regiony-
name_gap, name="segments", vp=vp)
      symbols <- NULL
      SNPnames <- textGrob(SNP.name, just="left", rot=-45,
            regionx[ind]-name_gap, regiony[ind]+name_gap, gp=gpar(cex=0.6, col="blu
e"), name="SNPnames", vp=vp)
            # snp[ind], snp[ind], gp=gpar(cex=0.6, col="blue"), name="SNPnames", vp=v
p)
      title <- editGrob(title, y=unit(geneMapLabelY+name_gap, "npc"))
    }
    geneMap <- gTree(children=gList(diagonal, segments, title, symbols, SNPnames),name
="geneMap")
  }} # if(add.map) end
```

In the if(flip) condition, symbols are set to NULL. This results in the symbols not being displayed or recorded on the geneMap. Below is the corrected code for the if(flip) condition.

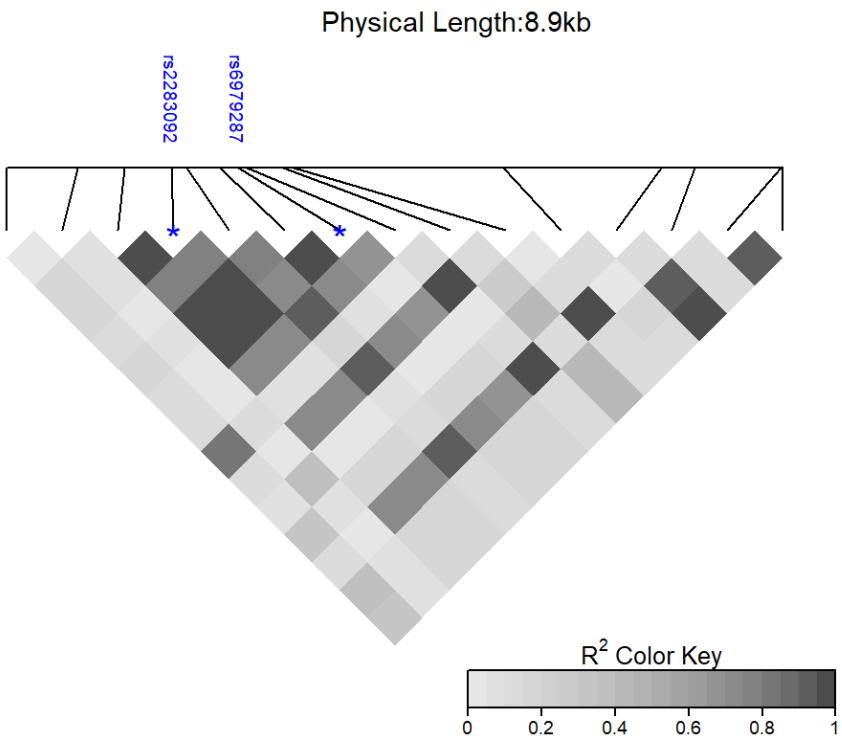
```
placeholder2 <- function(nsnps, add.map, genetic.distances,
                     geneMapLocation=0.15,
                     geneMapLabelX=NULL, geneMapLabelY=NULL,
                     distances="physical", vp=NULL,
                     SNP.name=NULL, ind=0, flip=FALSE)
  {
                if (flip) {
                  length_SNP_name <- max(nchar(SNP.name))
                  long_SNP_name <- paste(rep(8,length_SNP_name), collapse="")
                  name_gap <- convertWidth(grobWidth(textGrob(long_SNP_name)), "npc",v
alueOnly=TRUE)/sqrt(2)
                  diagonal<-linesGrob(seq.x, seq.y, gp=gpar(lty=1), name="diagonal", v
p=vp)
                  #diagonal<-linesGrob(seq.x+name_gap, seq.y-name_gap, gp=gpar(lty=
1), name="diagonal", vp=vp)
                  segments <- segmentsGrob(snp, snp, regionx, regiony, name="segment
s", vp=vp)
                  #segments <- segmentsGrob(snp+name_gap, snp-name_gap, regionx+name_g
ap, regiony-name_gap, name="segments", vp=vp)

                  ##############################################
                  # Bug: symbols was set to NULL here for some reason
                  symbols <- pointsGrob(snp[ind], snp[ind], pch="*",
                                        gp=gpar(cex=1.25, bg="blue", col="blue"), name
="symbols", vp=vp)
                  ##############################################
                  SNPnames <- textGrob(SNP.name, just="left", rot=-45,
                                       regionx[ind]-name_gap, regiony[ind]+name_gap, g
p=gpar(cex=0.6, col="blue"), name="SNPnames", vp=vp)
                  # snp[ind], snp[ind], gp=gpar(cex=0.6, col="blue"), name="SNPname
s", vp=vp)
                  title <- editGrob(title, y=unit(geneMapLabelY+name_gap, "npc"))
                }
                  geneMap <- gTree(children=gList(diagonal, segments, title, symbols,
SNPnames),name="geneMap")
  }
```

The corrected function is used in the revised LDheatmap function, currently named LDtest(). Demonstration of this improvement follows. Still to do: Correct the text placement on the flipped image. Functional at the moment but could be better.

```
# New code
if(!exists("LDTest", mode="function")) source("LDHeatmapTestFunctionMR.R")
if(!exists("LDheatmapMapNew.add", mode="function")) source("LDheatmapHelpers.R")
data(CEUData)
 # Flipped
 MyHeatmap <- LDTest(CEUSNP, genetic.distances = CEUDist,
                          color = grey.colors(20), flip=TRUE)
```



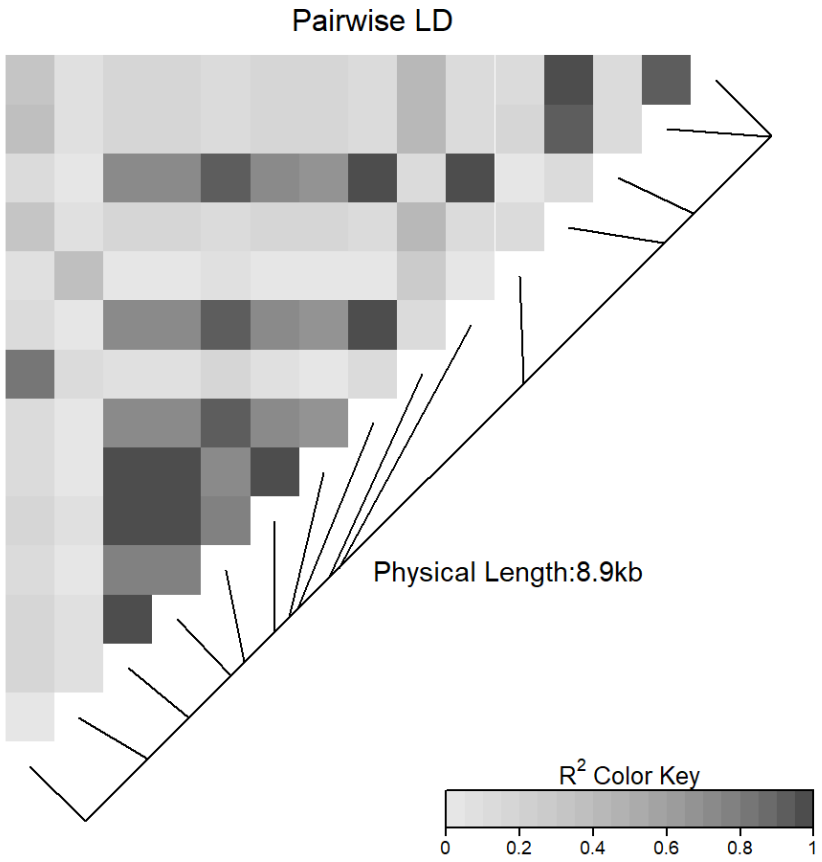

```
MyHeatmapSymbols <- LDTest(MyHeatmap, SNP.name = c("rs2283092", "rs6979287"))
```



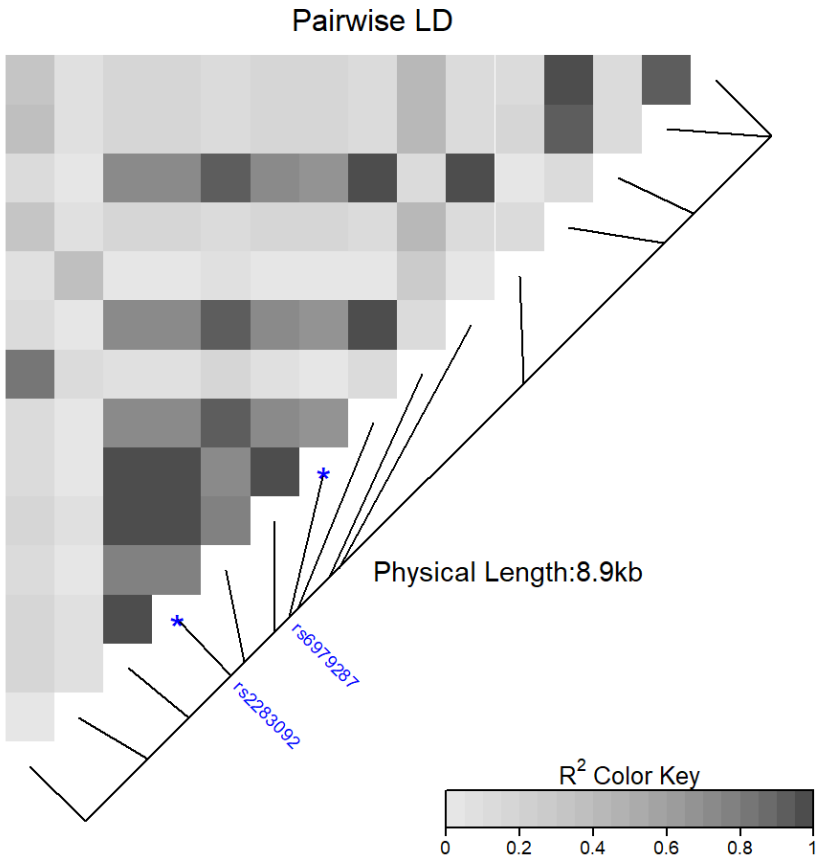

```
#childNames(grid.get("geneMap")) # Symbols available
childNames(MyHeatmapSymbols$LDheatmapGrob$children$geneMap)
```

```
## [1] "diagonal" "segments" "title"    "symbols"  "SNPnames"
```

```
# Not flipped
MyHeatmapTest <- LDTest(CEUSNP, genetic.distances = CEUDist,
                        color = grey.colors(20), flip=FALSE)
```




```
MyHeatmapTestSymbols <- LDTest(MyHeatmapTest, SNP.name = c("rs2283092", "rs6979287"))
```



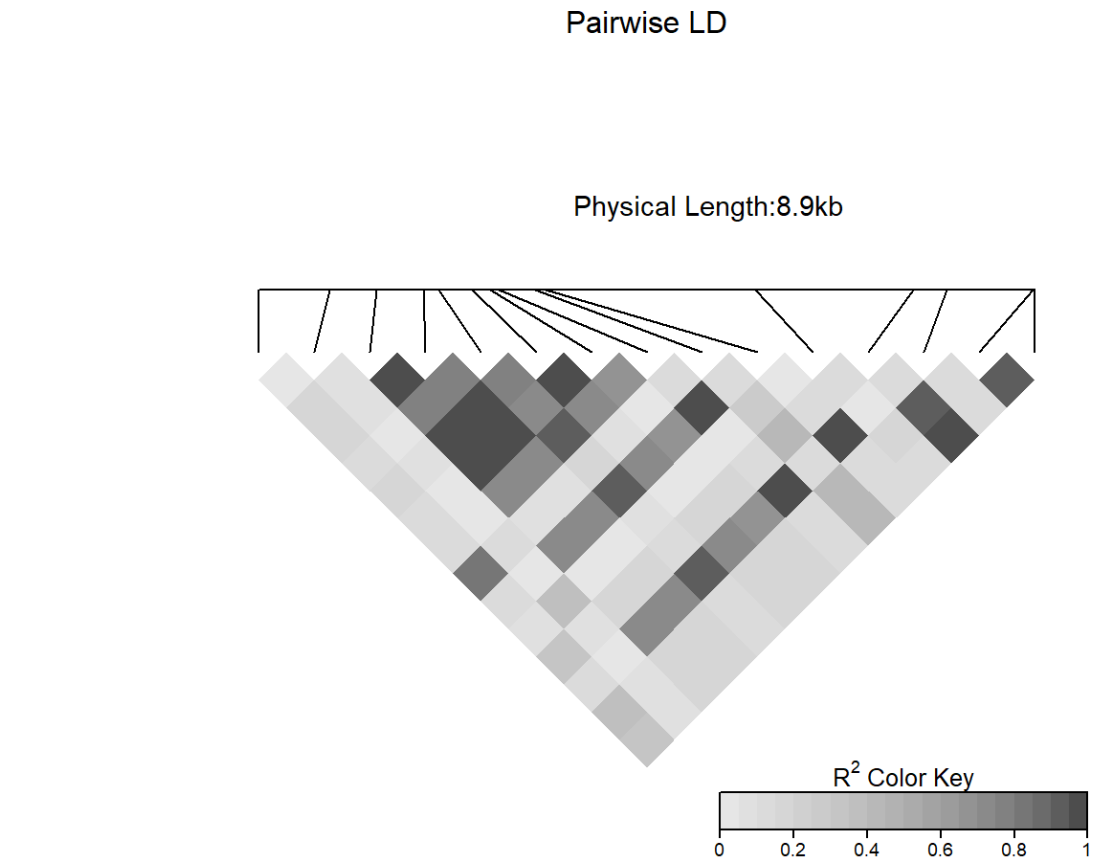

```
childNames(MyHeatmapTestSymbols$LDheatmapGrob$children$geneMap) # Symbols available
```

```
## [1] "diagonal" "segments" "title"    "symbols"  "SNPnames"
```

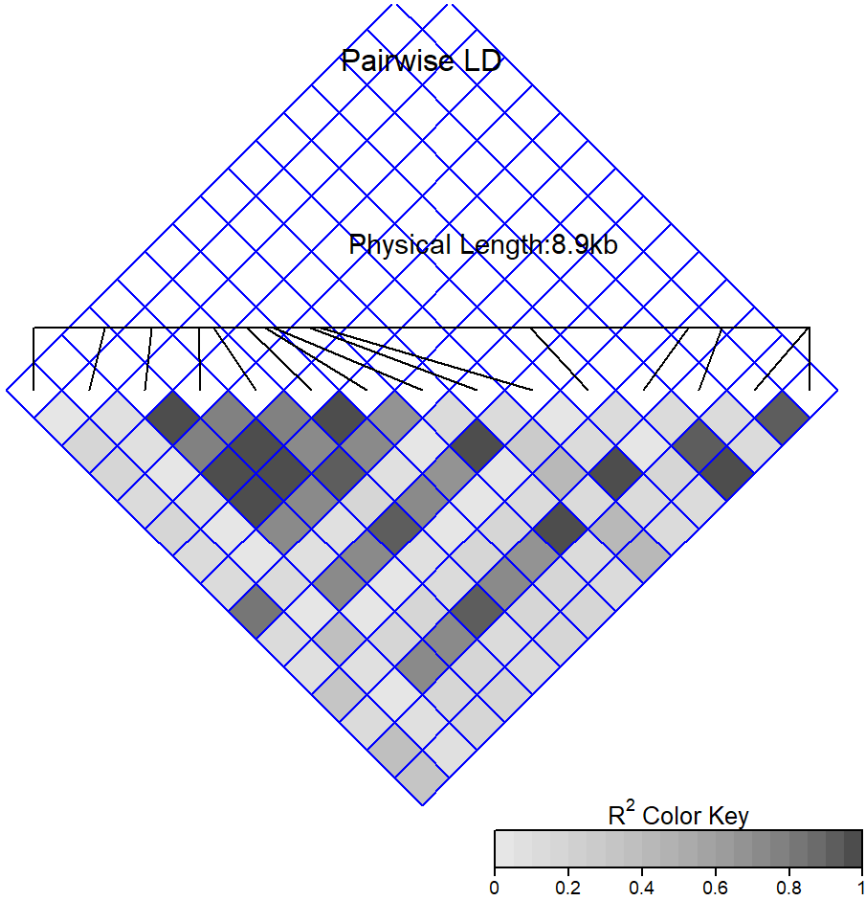
Bug Report 2: User reported issues with the underlying grid being entirely visible if they wanted to outline the cells in the heatmap. Request was for the cells only within the heatmap to be highlighted when using colours other than white.

Example of issue:

```
if(!exists("LDTest", mode="function")) source("LDHeatmapTestFunctionMR.R")
if(!exists("LDheatmapMapNew.add", mode="function")) source("LDheatmapHelpers.R")
data(CEUData)
LDTest(CEUSNP, genetic.distances = CEUDist,
                          color = grey.colors(20), flip=TRUE)
```




```
grid.edit(gPath("ldheatmap", "heatMap", "heatmap"), gp = gpar(col = "blue", lwd = 1))
```



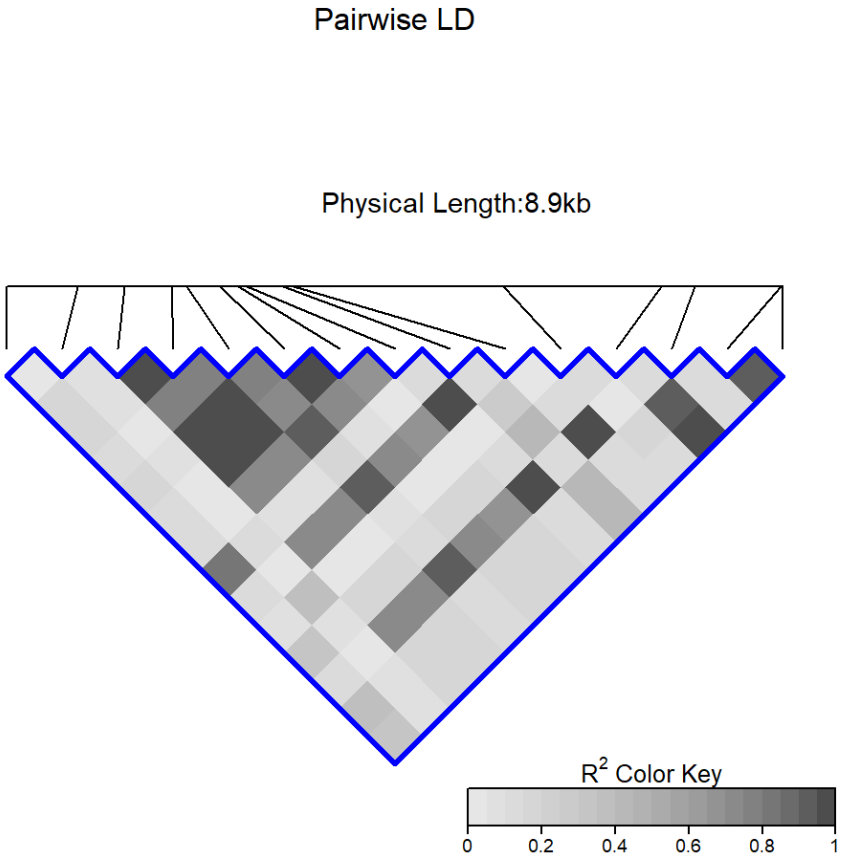
The problem demonstrated here was being caused by the grid.edit call editing all the cells in the output due to the matrix structure. Many resolution approaches could be taken to improve this situation though the path of least resistance seemed to be modifying the LDheatmap.highlight() function as it already outline groups of cells. Modifications to this function allowed it to outline the individual cells within a heatmap. Below is the modification and an exmaple of its use. Code for LDTest.highlight() has been saved in LDHeatmap.highlight.R

```
if(!exists("LDTest", mode="function")) source("LDHeatmapTestFunctionMR.R")
if(!exists("LDheatmapMapNew.add", mode="function")) source("LDheatmapHelpers.R")
LDTest.highlight <- function(LDheatmap, i, j, fill="NA", col="black", lwd=1, lty=1, fl
ipOutline=F, crissCross = F){

  requireNamespace("grid")
  # Highlights the perimeter of selected cells in the heatmap as a block
  backbone <- function(i,j,nSNP){
    x <- c(i-1,i-1,j-1)/nSNP
    y <- c(i,j,j)/nSNP
    cbind(x,y)
  }
  backboneFlip <- function(i,j,nSNP){
    x <- c(i,j,j)/nSNP
    y <- c(i-1,i-1,j-1)/nSNP
    cbind(x,y)
  }

  rectVert <- function(i, j , nSNP){
    rectangles <- data.frame()
    for( k in i:(j-1)){
      x <- c(k, k, k + 1, k + 1) / nSNP
      y <- c(k, j, k, j) / nSNP
      coords <- cbind(x, y)
      rectangles <- rbind(rectangles, coords)
    }
    return(rectangles)
  }

  rectHorizontal <- function(i, j ,nSNP){
    rectangles <- data.frame()
    for(m in i:(j-1)){
      x <- c(i-1, m , i-1, m) / nSNP
      y <- c(m+1, m+1, m+2, m+2) / nSNP
      coords <- cbind(x, y)
      rectangles <- rbind(rectangles, coords)
    }
    return(rectangles)
  }

  zigzag <- function(i,j,nSNP){
    c1 <- j-i
    nvert <- (2*c1)-1
    x <-c(j-1,rep((j-2):(j-c1),each=2))
    y <- c(rep((j-1):(j-(c1-1)),each=2),j-c1)
    cbind(x,y)/nSNP
  }
  zigzagFlip <- function(i,j,nSNP){
```

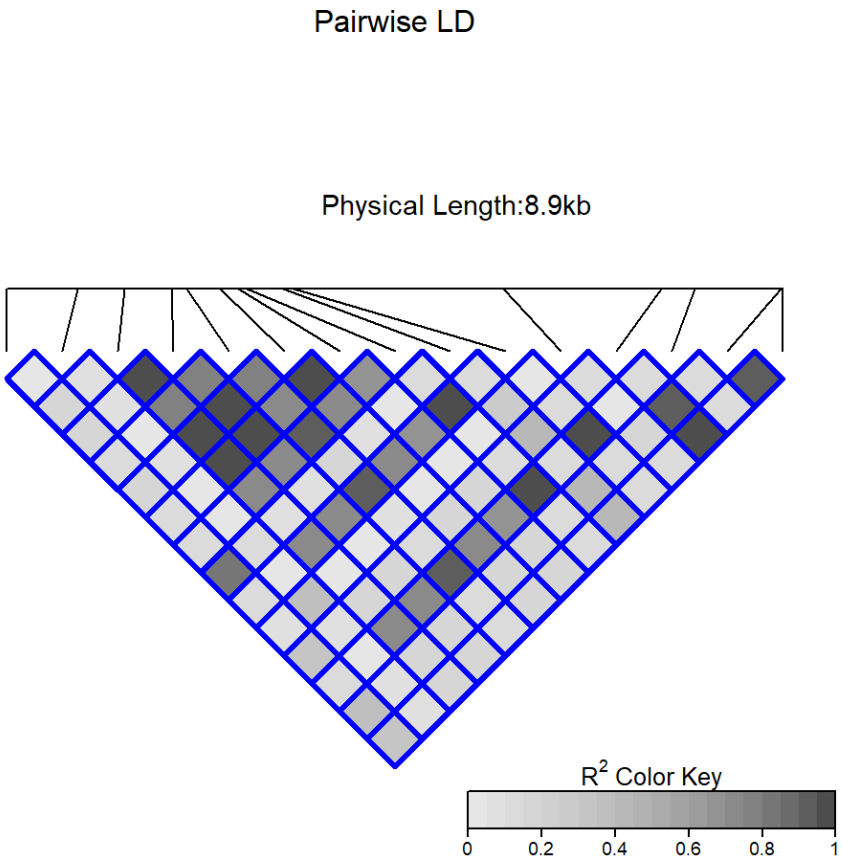
```
    c1 <- j-i
    nvert <- (2*c1)-1
    y <-c(j-1,rep((j-2):(j-c1),each=2))
    x <- c(rep((j-1):(j-(c1-1)),each=2),j-c1)
    cbind(x,y)/nSNP
  }

  nSNP <- dim(LDheatmap$LDmatrix)[1]
  if(length(i)>1 | length (j) > 1) stop("i and j must be scalar indices")
  if((i<1 | i>nSNP) |(j<1 | j>nSNP) )
    stop(paste("index out of bounds, i and j must be in (1,",nSNP,")",sep=""))
  if(i==j) stop("i cannot be equal to j")
  if(i>j){
    h<-i
    i <- j
    j <- h
  }
  pgon <- data.frame(rbind(backbone(i,j,nSNP), zigzag(i,j,nSNP)))
  if(!is.null(LDheatmap$flipVP)) pgon <- data.frame(rbind(backboneFlip(i,j,nSNP), zigz
agFlip(i,j,nSNP)))
  ## Square or almost square interior Blocks
  names(pgon) <- c("x","y")
  # For the grid highlight case
  vertRectangles <- rectVert(i, j, nSNP = dim(LDheatmap$LDmatrix)[1])
  horizonRectangles <- rectHorizontal(i, j, nSNP = dim(LDheatmap$LDmatrix)[1])
  names(vertRectangles) <- c("x", "y")
  names(horizonRectangles) <- c("x", "y")
  #
  heatmap.vp <- LDheatmap$heatmapVP$name
  #If heatmap.vp is on the grid display list, i.e., it is included in the
  #returned value of current.vpTree(), a[1]=1 else a[1]=NA:
  a <- grep(paste("[", heatmap.vp, "]", sep=""), as.character(current.vpTree()), fixed
=TRUE)
  if(!is.na(a[1]))   seekViewport(heatmap.vp)
  else               pushViewport(LDheatmap$heatmapVP)
  if (!is.null(LDheatmap$flipVP)) pushViewport(LDheatmap$flipVP)
  # Added section #
  if(flipOutline == T){
    tempy <- pgon$y
    tempx <- pgon$x
    pgon$y <- tempx
    pgon$x <- tempy
  }
  highlight <- polygonGrob(x=pgon$x, y=pgon$y,
                           gp=gpar(col=col, fill=fill, lwd=lwd, lty=lty), name="highli
ght")

 #
  if(crissCross == TRUE){
```

```
    for(i in 1:(dim(vertRectangles)[1]/4)){
      width <- vertRectangles$x[(i-1)*4 + 3] - vertRectangles$x[(i-1)*4 + 1]
      height <- vertRectangles$y[(i-1)*4 + 1] - vertRectangles$y[(i-1)*4 + 2]
      if(is.null(LDheatmap$flipVP)){
        oneRect <- rectGrob(x = vertRectangles$x[(i-1)*4+1] - width/2, y = vertRectang
les$y[(i-1)*4+1] - height/2,
                            width = width,
                            height= height,
                            gp=gpar(col=col, fill=fill, lwd=lwd, lty=lty), name="rec
t")
      }
      else{
        # Flip is swapping of x-y coordinates, therefore reverse assignment of width a
nd height
        width <- vertRectangles$y[(i-1)*4 + 1] - vertRectangles$y[(i-1)*4 + 2]
        height <- vertRectangles$x[(i-1)*4 + 3] - vertRectangles$x[(i-1)*4 + 1]
        oneRect <- rectGrob(x = vertRectangles$y[(i-1)*4+1] - width/2, y = vertRectang
les$x[(i-1)*4+1] - height/2,
                            width = width,
                            height= height,
                            gp=gpar(col=col, fill=fill, lwd=lwd, lty=lty), name="rec
t")
      }

      grid.draw(oneRect)
    }
    for(j in 1:(dim(horizonRectangles)[1]/4)){
      width <- horizonRectangles$x[(j-1)*4 + 2] - horizonRectangles$x[(j-1)*4 + 1]
      height<- horizonRectangles$y[(j-1)*4 + 3] - horizonRectangles$y[(j-1)*4 + 1]
      if(is.null(LDheatmap$flipVP)){
        oneRect <- rectGrob(x = horizonRectangles$x[(j-1)*4 + 2] - width/2, y = horizo
nRectangles$y[(j-1)*4 + 1] - height/2,
                            width = width,
                            height = height,
                            gp=gpar(col=col, fill=fill, lwd=lwd, lty=lty), name="rec
t")
      }
      else{
        # Flip is swapping of x-y coordinates, therefore reverse assignment of width a
nd height
        width <- horizonRectangles$y[(j-1)*4 + 3] - horizonRectangles$y[(j-1)*4 + 1]
        height<- horizonRectangles$x[(j-1)*4 + 2] - horizonRectangles$x[(j-1)*4 + 1]
        oneRect <- rectGrob(x = horizonRectangles$y[(j-1)*4 + 1] - width/2, y = horizo
nRectangles$x[(j-1)*4 + 2] - height/2,
                            width = width,
                            height = height,
                            gp=gpar(col=col, fill=fill, lwd=lwd, lty=lty), name="rec
t")
```

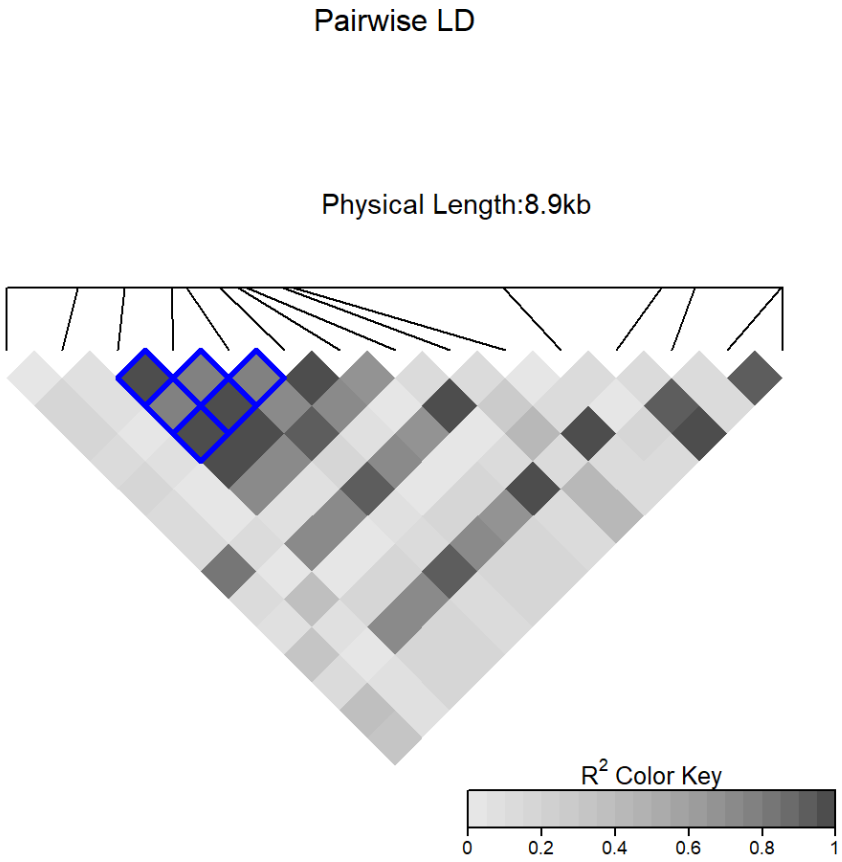
```
      }
      grid.draw(oneRect)
    }
  }
  #
  grid.draw(highlight)
  if(!is.na(a[1]))  upViewport(0)  #back to the root viewport
  else              popViewport()
  invisible(pgon)
}

grid.newpage()

data(CEUData)
# Normal highlight() functionality
MyHeatmap <- LDheatmap(CEUSNP, genetic.distances = CEUDist,
                       color = grey.colors(20), flip = TRUE)
onlyOutline <- LDTest.highlight(MyHeatmap, 1, 15, col = "blue", fill = NA, lwd =3, fli
pOutline = FALSE, crissCross = FALSE)
```



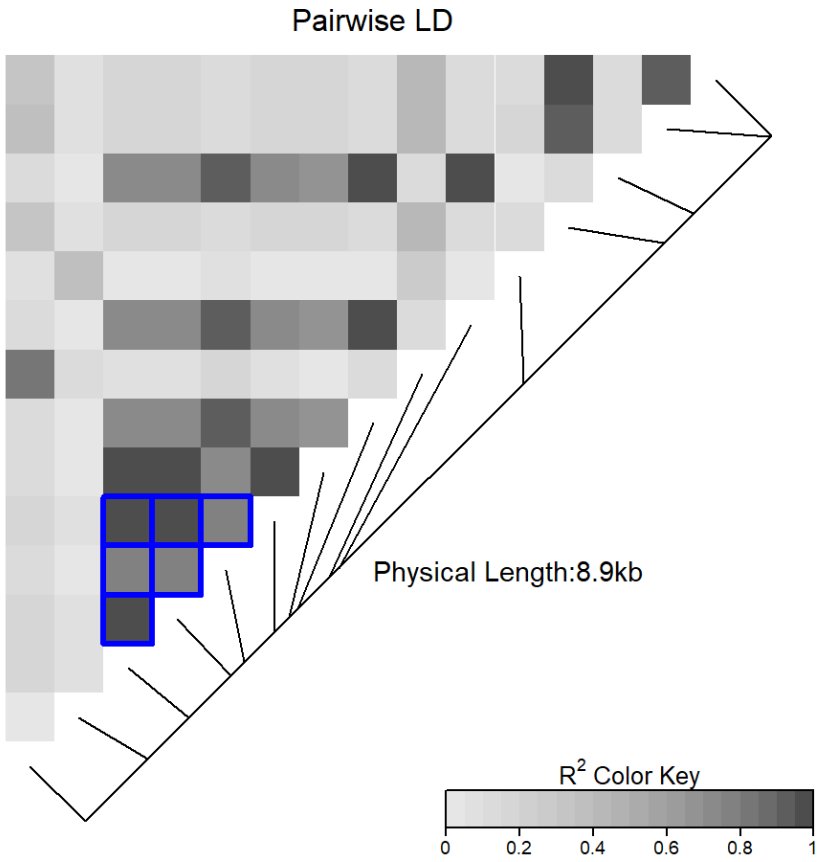
```
# Extended highlight() functionality to wrap all cells
MyHeatmap2 <- LDheatmap(CEUSNP, genetic.distances = CEUDist,
                        color = grey.colors(20), flip = TRUE)
allCellsOutline <- LDTest.highlight(MyHeatmap2, 1, 15, col = "blue", fill = NA, lwd =
3, flipOutline = FALSE, crissCross = TRUE)
```




```
# Now only some of the cells
MyHeatmap3 <- LDheatmap(CEUSNP, genetic.distances = CEUDist,
                        color = grey.colors(20), flip = TRUE)
someCellsOutline <- LDTest.highlight(MyHeatmap3, 3, 6, col = "blue", fill = NA, lwd =
3, flipOutline = FALSE, crissCross = TRUE)
```




```
# Verify that it works for non-flipped
MyHeatmap4 <- LDheatmap(CEUSNP, genetic.distances = CEUDist,
                        color = grey.colors(20), flip = FALSE)
someCellsOutlineNF <- LDTest.highlight(MyHeatmap4, 3, 6, col = "blue", fill = NA, lwd
=3, flipOutline = FALSE, crissCross = TRUE)
```

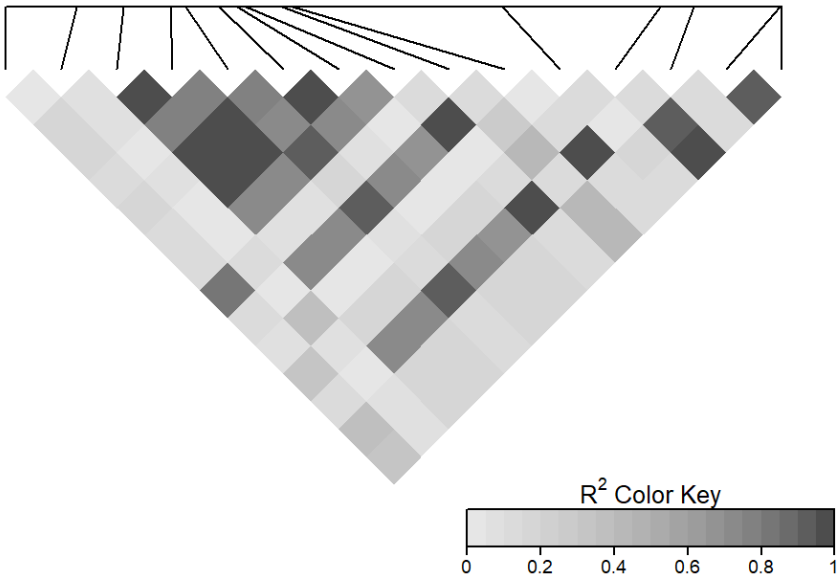


The highlighting of each cell is controlled by the crissCross parameter. A TRUE value dictates the outlining of each cell while a FALSE value dictates only the highlighting of the outer cells, the default behaviour of LDheatmap.highlight().

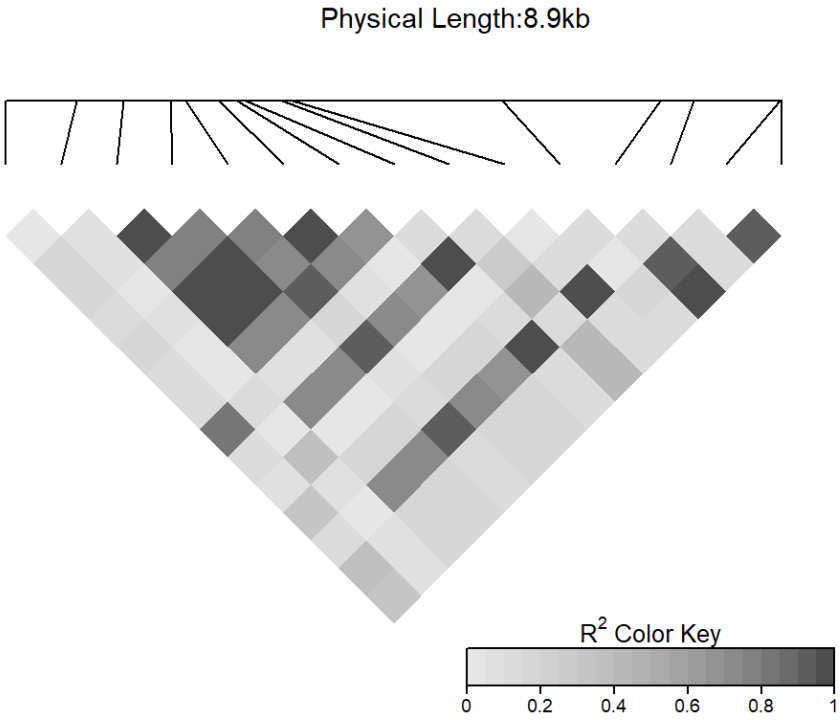
Bug Report 2.5: User requested the ability to control geneMap distance from heatmap. No image or reproducibility of the idea was provided. As such, a generic function was created with the intention of providing the desired feature.

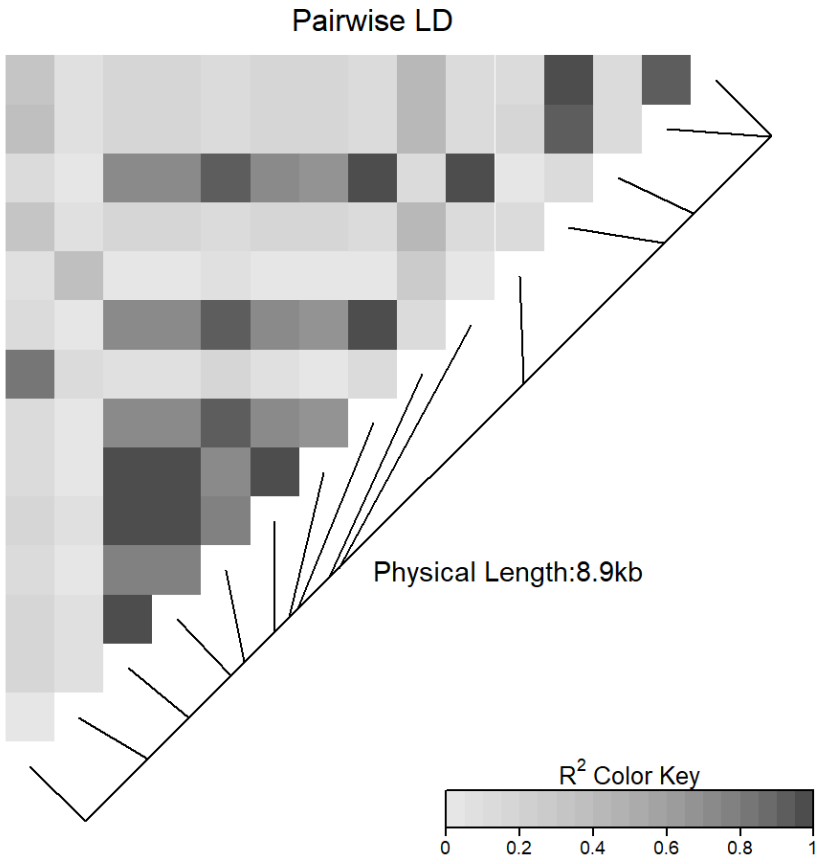
Pairwise LD

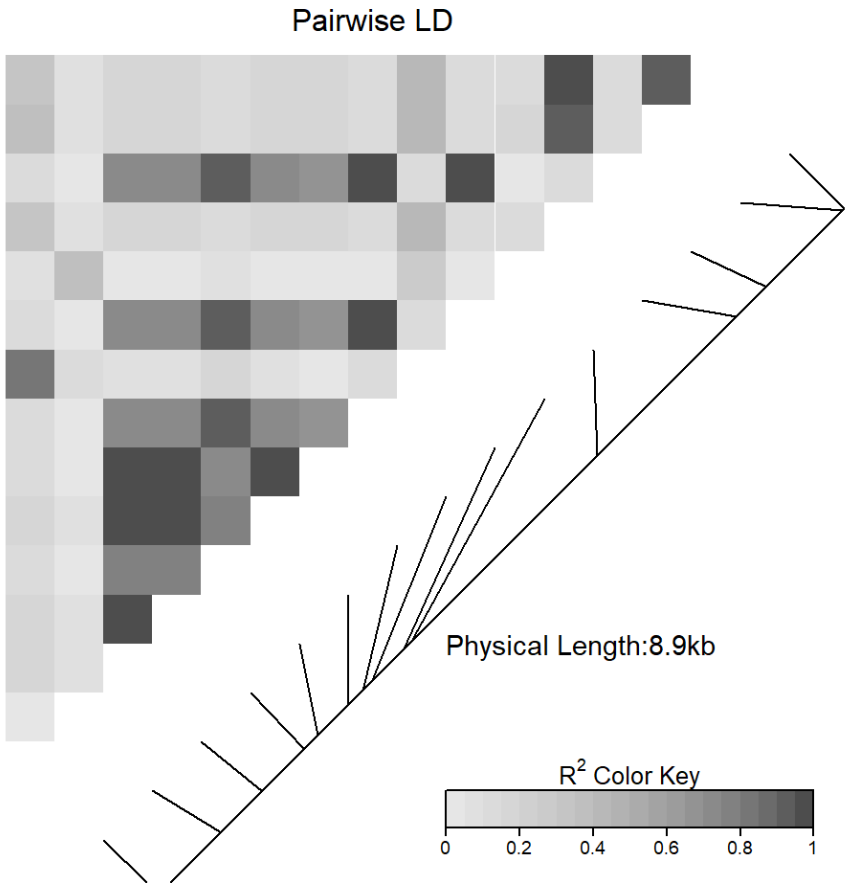
Physical Length:8.9kb



Pairwise LD




Pairwise LD
Physical Length:8.9kb
R2 Color Key
0
0.2
0.4
0.6
0.8
1



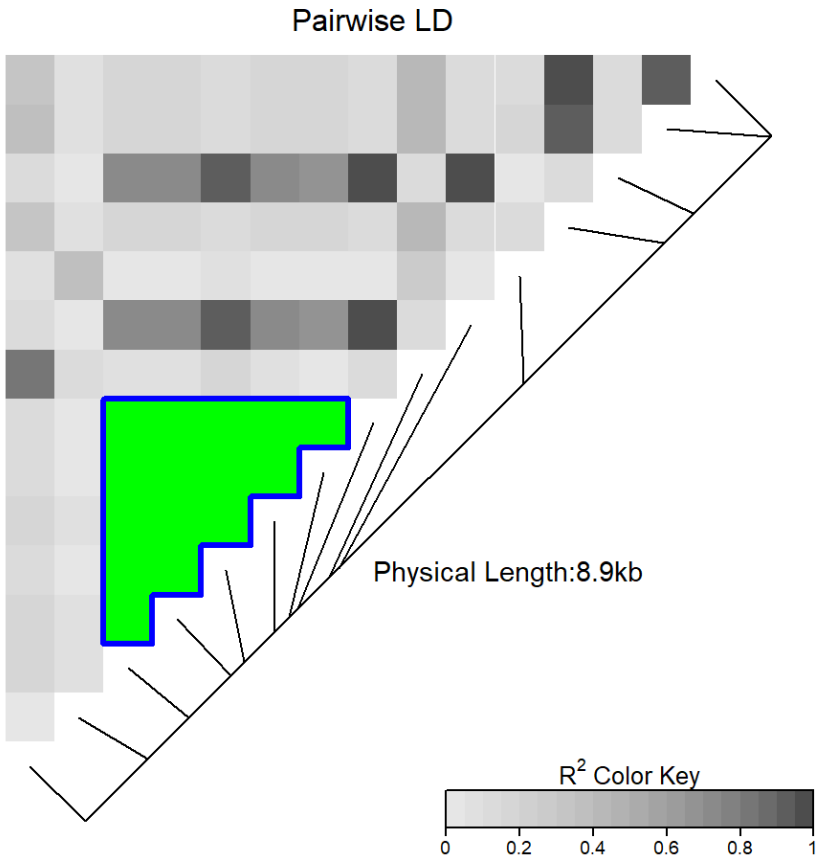
Movement of the genemap is controlled by the distance parameter. Currently trivial character values are used though it can be modified for exact values by changing distance to a numeric argument.

Remaining issues: Last part of this bug report/ request asked for a way to add bp positions (i.e. beginning and end positions at the segment bar). I am unsure of what these are though I can likely implement them with some advice.

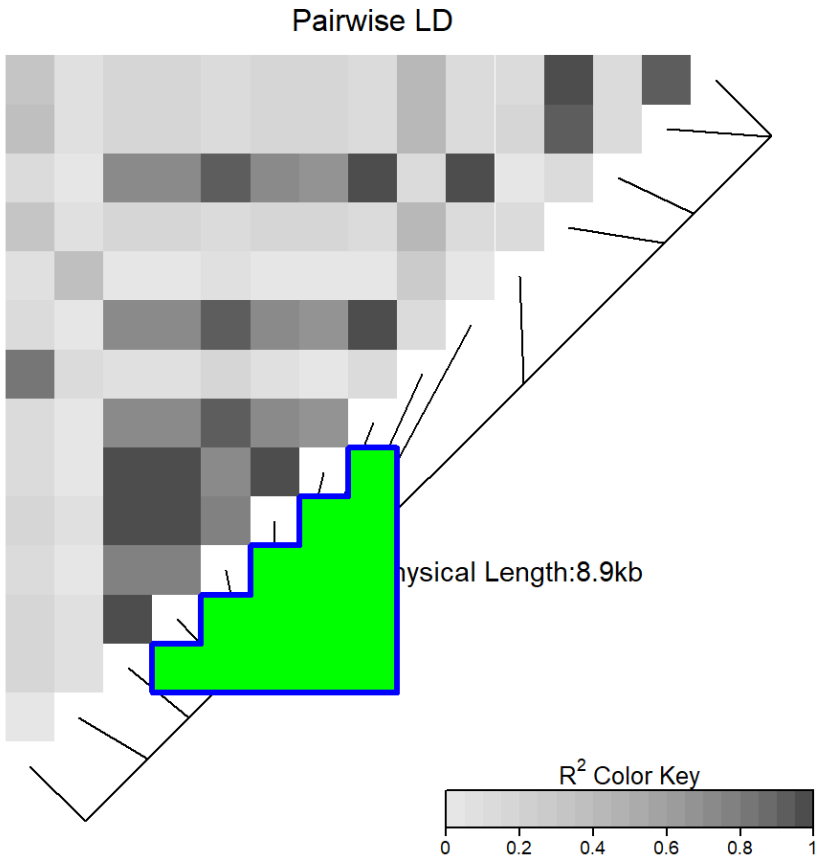
Bug Report 3: When using the highlight function on a flipped graph, the output polygon is on the wrong side of the heatmap.

In testing the bug report, I was unable to duplicate the same problem the user had. Because of this, I implemented a parameter in the highlight function such that any user experiencing this problem can fix with a simple adjustment. Code changes to the highlight function are included in the earlier discussion for Bug Report 2.

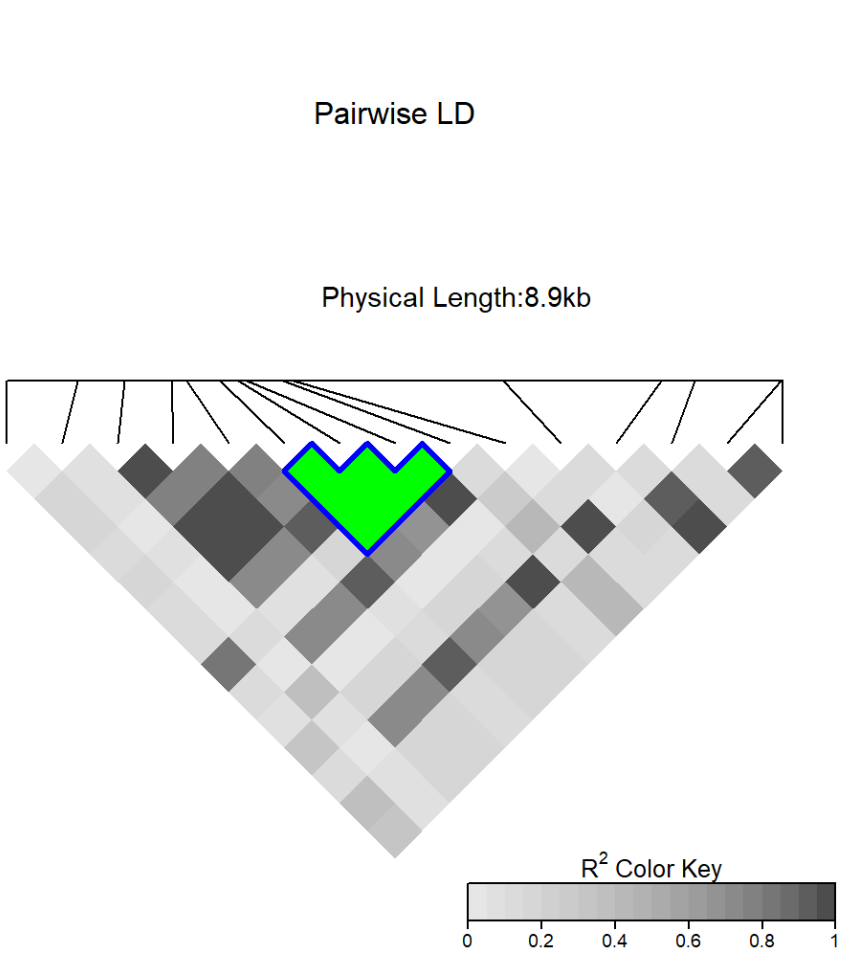
```
# Bug report: highlight function needs to be mirrored when used with flip = TRUE #
# Sample of highlight on normal heatmap
if(!exists("LDTest", mode="function")) source("LDHeatmapTestFunctionMR.R")
if(!exists("LDheatmapMapNew.add", mode="function")) source("LDheatmapHelpers.R")
#library(mvtnorm)
data(CEUData)
tt <- LDheatmap(CEUSNP, genetic.distances=CEUDist)
LDTest.highlight(tt, 3, 8, col="blue", fill="green", lwd=3)
```



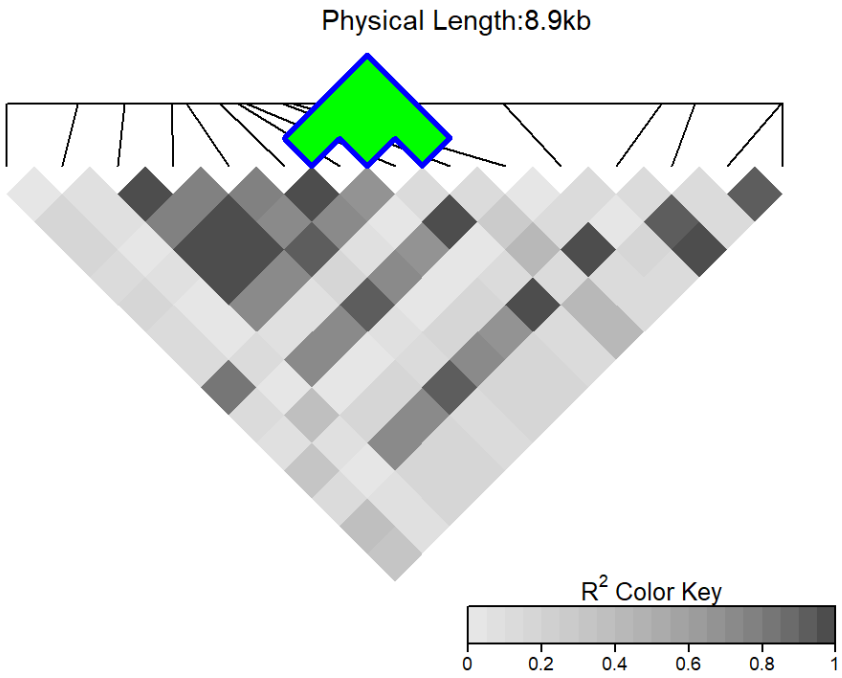

```
tt <- LDheatmap(CEUSNP, genetic.distances=CEUDist)
LDTest.highlight(tt, 3, 8, col="blue", fill="green", lwd=3, flipOutline = TRUE)
```




```
# Sample of highlight on flipped heatmap
ttFlip <- LDheatmap(CEUSNP, genetic.distances = CEUDist, flip = TRUE)
LDTest.highlight(ttFlip, 6, 9, col = "blue", fill = "green", lwd = 3)
```



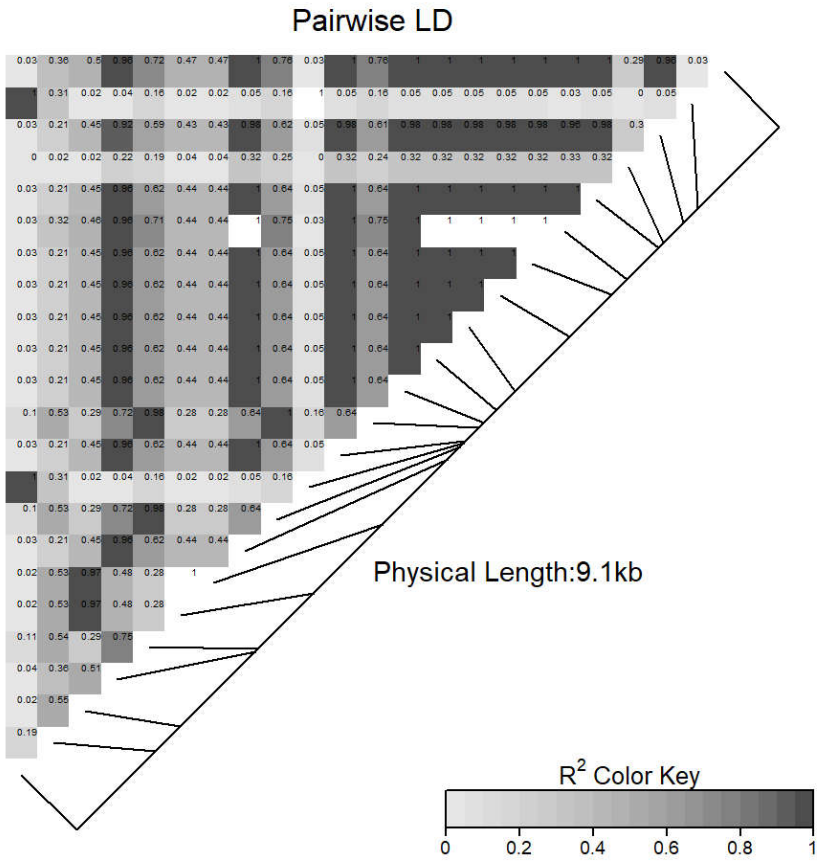

```
# Solution: Add flipOutline parameter such that if a user encounters a flip problem th
ey can change the flipOutline param value to reverse
ttFlip <- LDheatmap(CEUSNP, genetic.distances = CEUDist, flip = TRUE)
LDTest.highlight(ttFlip, 6, 9, col = "blue", fill = "green", lwd = 3, flipOutline = TR
UE)
```



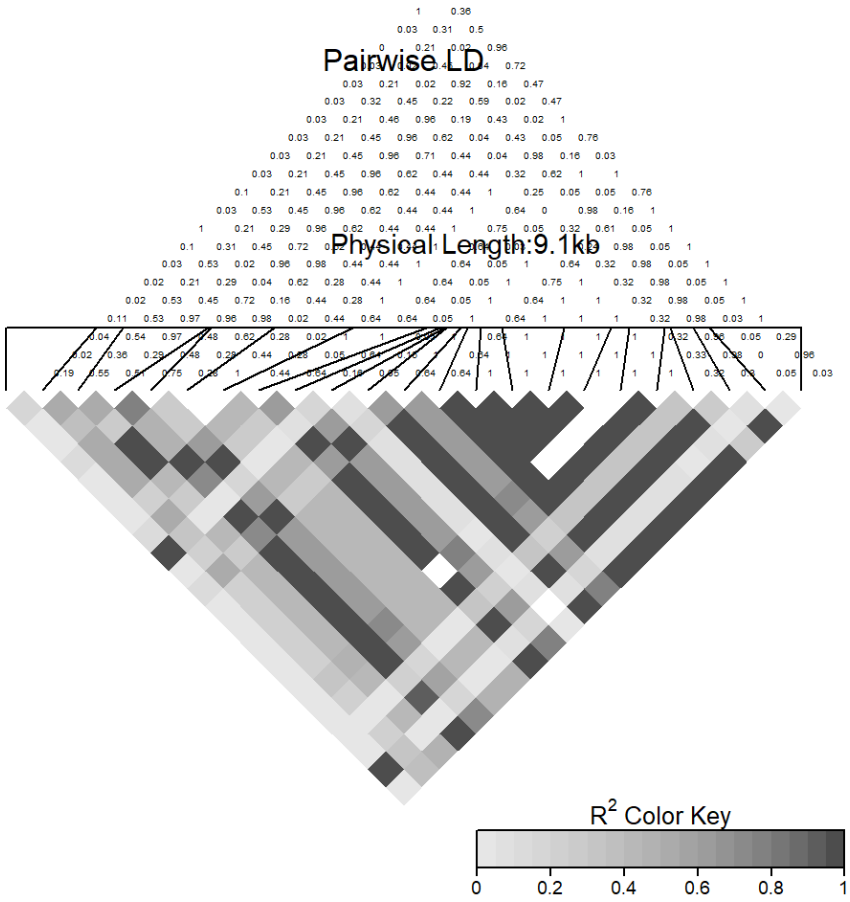
Bug Report 4: When trying to add correlation values to the individual cells, values were not being adequately mapped for flipped graphs.

Recreation of the plotting:

```
data(GIMAP5.CEU)
llText <- LDheatmap(GIMAP5.CEU$snp.data,GIMAP5.CEU$snp.support$Position,flip=FALSE, te
xt = TRUE)
```




```
llText2<- LDheatmap(GIMAP5.CEU$snp.data,GIMAP5.CEU$snp.support$Position,flip=TRUE, tex
t = TRUE)
```



Resolving this problem involved editing the code in the LDheatmap function call. The problematic code is as follows:

```
LDheatmapPlaceholder<-
   function (gdat, genetic.distances=NULL,
             distances="physical", LDmeasure="r", title="Pairwise LD",
             add.map=TRUE, add.key=TRUE, geneMapLocation=0.15,
             geneMapLabelX=NULL, geneMapLabelY=NULL,
             SNP.name=NULL, color=NULL,
             newpage=TRUE, name="ldheatmap", vp.name=NULL,
             pop=FALSE, flip=NULL, text=FALSE)
   {
     # # # #
      ImageText <- NULL
      if (text) ImageText<-makeImageText(dim(LDmatrix)[1],dim(LDmatrix)[2], round(imgL
Dmatrix, digits = 2), name="heatmaptext")
      title <- textGrob(title, 0.5, 1.05, gp=gpar(cex=1.0), name="title")
      if (flip) {
         ImageRect <- editGrob(ImageRect, vp=flipVP)
         if (text)
            ImageText <- editGrob(ImageText, vp=flipVP, rot=45, just="left")
      }
      heatMap <- gTree(children=gList(ImageRect, ImageText, title), name="heatMap")
   }
```

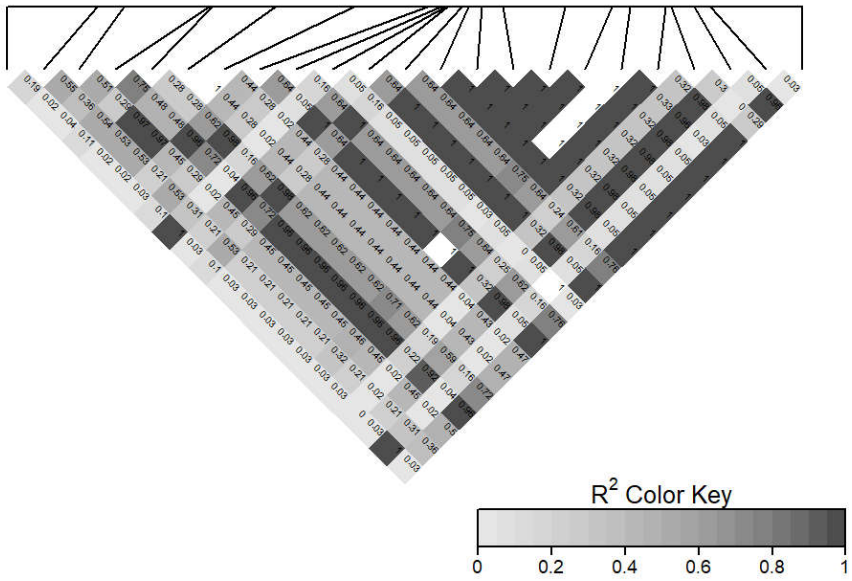
Generating the text for the image through a rotation of 45 degrees in the flipVP viewport causes the text to be displayed above the graph. Modification to using a rotation of 0 degrees seems to alleviate the problem. Code, as follows, is stored in the LDTest function.

```
LDTestPlaceholder <- function(gdat, genetic.distances=NULL,
                          distances="physical", LDmeasure="r", title="Pairwise LD",
                          add.map=TRUE, add.key=TRUE, geneMapLocation=0.15,
                          geneMapLabelX=NULL, geneMapLabelY=NULL,
                          SNP.name=NULL, color=NULL,
                          newpage=TRUE, name="ldheatmap", vp.name=NULL,
                          pop=FALSE, flip=NULL, text=FALSE)
  {
    # # # #
    ImageText <- NULL
    if (text) ImageText<-makeImageText(dim(LDmatrix)[1],dim(LDmatrix)[2], round(imgLDm
atrix, digits = 2), name="heatmaptext")
    title <- textGrob(title, 0.5, 1.05, gp=gpar(cex=1.0), name="title")

    if (flip) {
      ImageRect <- editGrob(ImageRect, vp=flipVP)
      if (text)
        # Added flip = TRUE parameter to better utilize makeImageText() in the flippe
d case
        ImageText <- makeImageText(dim(LDmatrix)[1],dim(LDmatrix)[2], round(imgLDmatri
x, digits = 2), name="heatmaptext", flip = TRUE)
        ImageText <- editGrob(ImageText, vp=flipVP, rot=0, just=c("right", "top"))
    }
  }
```

The change made above involved the adding of a flip parameter to makeImageText() such that the data could be added to the graph in the appropriate order. Adjusted the call of makeImageText() in LDTest() to accomodate for the added parameter.

Demonstration of the improvement:

```
if(!exists("LDTest", mode="function")) source("LDHeatmapTestFunctionMR.R")
if(!exists("LDheatmapMapNew.add", mode="function")) source("LDheatmapHelpers.R")
data(GIMAP5.CEU)
llText <- LDTest(GIMAP5.CEU$snp.data,GIMAP5.CEU$snp.support$Position,flip=TRUE, text
= TRUE)
```



Need to ensure that this function is being run with the LDheatmapHelpers.R version of makeImageText (). Will further verify this test with corner case investigation.

Bug Report 5: User reported being able to successfully add 2 scatterplots above the rotated LDheatmap but was unable to incorporate 3 or more.

Gloria investigated the problem and found that the problem involved an incorrect location adjustment to the vp used for plotting. Her modification changes the contructVP() default location = 0.03 to location = (0.2)*(number of scatterplots - 1) + 0.03. This means that the locations used are 0.03, 0.23, 0.43, … for 1, 2, 3, … scatterplots. Reasoning behind this change is that the height of each scatterplot is 0.2 units and overlap can be prevented by appropriately accounting for it. Functions listed first.

```
LDheatmap.addScatterplot_test1 <- function(LDheatmap, P, height=0.2, ylab=NULL, ylim=N
ULL, type="points",color,pch) {
    if (dim(LDheatmap$LDmatrix)[1] != length(P)) {
    print("Length of vector not equal number of SNPs in LDheatmap")
    return()
  }

  flip <- !is.null(LDheatmap$flipVP)
  vp <- constructVP(LDheatmap$LDheatmapGrob, 0.03, flip)
  vp$height <- unit(height, "npc")
  vp$name <- "associationVP"


  if (is.null(ylim))
    ylim <- c(floor(min(P)), ceiling(max(P)))
  vp$yscale <- ylim
  vp$xscale <- c(min(LDheatmap$genetic.distances), max(LDheatmap$genetic.distances))


  xaxis <- linesGrob(x = vp$xscale, y = 0, default.units = "native",
                     name = "xaxis")
  yaxis <- linesGrob(x = min(LDheatmap$genetic.distances),
                     y = vp$yscale, default.units = "native", name = "yaxis")
  yaxisT <- yaxisGrob(name = "yaxis_ticks", gp = gpar(fontsize = 7))
  ylab <- textGrob(ylab, rot = 90, gp = gpar(fontsize = 9),
                   name = "yaxis_title", x = unit(min(LDheatmap$genetic.distances),
                                                  "native") - unit(10, "millimeters"))
  vpstack <- vp
  if (flip)
    vpstack <- vpStack(LDheatmap$flipVP, vp)
  association <- gTree(children = gList(xaxis, yaxis, yaxisT,
                                        ylab), name = "association", vp = vpstack)

  if (type == "points" || type == "both") {
    graph_points <- pointsGrob(LDheatmap$genetic.distances, P, size = unit(2, "millime
ters"), name = "points",pch=16, gp=gpar(col="red"))
    association <- addGrob(association, graph_points)
  }
  if (type == "lines" || type == "both") {
    graph_lines <- linesGrob(LDheatmap$genetic.distances,
                             P, default.units = "native", name = "lines")
    association <- addGrob(association, graph_lines)
  }
  LDheatmap$LDheatmapGrob <- addGrob(LDheatmap$LDheatmapGrob,
                                     association)
  LDheatmap$LDheatmapGrob <- moveTitles(LDheatmap$LDheatmapGrob,
                                        vp)
  return(LDheatmap)
```

```
}
environment(LDheatmap.addScatterplot_test1) <- asNamespace('LDheatmap')

LDheatmap.addScatterplot_test2 <- function(LDheatmap, P, height=0.2, ylab=NULL, ylim=N
ULL, type="points",color,pch) {
  if (dim(LDheatmap$LDmatrix)[1] != length(P)) {
    print("Length of vector not equal number of SNPs in LDheatmap")
    return()
  }
  flip <- !is.null(LDheatmap$flipVP)
  vp <- constructVP(LDheatmap$LDheatmapGrob, 0.03, flip)
  vp$height <- unit(height, "npc")
  vp$name <- "associationVP"
  if (is.null(ylim))
    ylim <- c(floor(min(P)), ceiling(max(P)))
  vp$yscale <- ylim
  vp$xscale <- c(min(LDheatmap$genetic.distances), max(LDheatmap$genetic.distances))
  xaxis <- linesGrob(x = vp$xscale, y = 0, default.units = "native",
                     name = "xaxis")
  yaxis <- linesGrob(x = min(LDheatmap$genetic.distances),
                     y = vp$yscale, default.units = "native", name = "yaxis")
  yaxisT <- yaxisGrob(name = "yaxis_ticks", gp = gpar(fontsize = 7))
  ylab <- textGrob(ylab, rot = 90, gp = gpar(fontsize = 9),
                   name = "yaxis_title", x = unit(min(LDheatmap$genetic.distances),
                                                  "native") - unit(10, "millimeters"))
  vpstack <- vp
  if (flip)
    vpstack <- vpStack(LDheatmap$flipVP, vp)
  association2 <- gTree(children = gList(xaxis, yaxis, yaxisT,
                                         ylab), name = "association2", vp = vpstack)
  if (type == "points" || type == "both") {
    graph_points <- pointsGrob(LDheatmap$genetic.distances, P, size = unit(2, "millime
ters"), name = "points",pch=16, gp=gpar(col="purple"))
    association2 <- addGrob(association2, graph_points)
  }
  if (type == "lines" || type == "both") {
    graph_lines <- linesGrob(LDheatmap$genetic.distances,
                             P, default.units = "native", name = "lines")
    association2 <- addGrob(association2, graph_lines)
  }
  LDheatmap$LDheatmapGrob <- addGrob(LDheatmap$LDheatmapGrob,
                                     association2)
  LDheatmap$LDheatmapGrob <- moveTitles(LDheatmap$LDheatmapGrob,
                                        vp)
  return(LDheatmap)

}
environment(LDheatmap.addScatterplot_test2) <- asNamespace('LDheatmap')
```

```
LDheatmap.addScatterplot_test3 <- function(LDheatmap, P, height=0.2, ylab=NULL, ylim=N
ULL, type="points",color,pch) {

   if (dim(LDheatmap$LDmatrix)[1] != length(P)) {
    print("Length of vector not equal number of SNPs in LDheatmap")
    return()
  }
  flip <- !is.null(LDheatmap$flipVP)
  vp <- constructVP(LDheatmap$LDheatmapGrob, 0.23, flip)
  vp$height <- unit(height, "npc")
  vp$name <- "associationVP"
  if (is.null(ylim))
    ylim <- c(floor(min(P)), ceiling(max(P)))
  vp$yscale <- ylim
  vp$xscale <- c(min(LDheatmap$genetic.distances), max(LDheatmap$genetic.distances))
  xaxis <- linesGrob(x = vp$xscale, y = 0, default.units = "native",
                     name = "xaxis")
  yaxis <- linesGrob(x = min(LDheatmap$genetic.distances),
                     y = vp$yscale, default.units = "native", name = "yaxis")
  yaxisT <- yaxisGrob(name = "yaxis_ticks", gp = gpar(fontsize = 7))
  ylab <- textGrob(ylab, rot = 90, gp = gpar(fontsize = 9),
                   name = "yaxis_title", x = unit(min(LDheatmap$genetic.distances),
                                                  "native") - unit(10, "millimeters"))
  vpstack <- vp
  if (flip)
    vpstack <- vpStack(LDheatmap$flipVP, vp)
  association3 <- gTree(children = gList(xaxis, yaxis, yaxisT,
                                         ylab), name = "association3", vp = vpstack)
  if (type == "points" || type == "both") {
    graph_points <- pointsGrob(LDheatmap$genetic.distances, P, size = unit(2, "millime
ters"), name = "points",pch=16, gp=gpar(col="green4"))
    association3 <- addGrob(association3, graph_points)
  }
  if (type == "lines" || type == "both") {
    graph_lines <- linesGrob(LDheatmap$genetic.distances,
                             P, default.units = "native", name = "lines")
    association3 <- addGrob(association3, graph_lines)
  }
  LDheatmap$LDheatmapGrob <- addGrob(LDheatmap$LDheatmapGrob,
                                     association3)
  LDheatmap$LDheatmapGrob <- moveTitles(LDheatmap$LDheatmapGrob,
                                        vp)
  return(LDheatmap)
}
environment(LDheatmap.addScatterplot_test3) <- asNamespace('LDheatmap')


LDheatmap.addScatterplot_test4 <- function(LDheatmap, P, height=0.2, ylab=NULL, ylim=N
ULL, type="points",color,pch) {
```

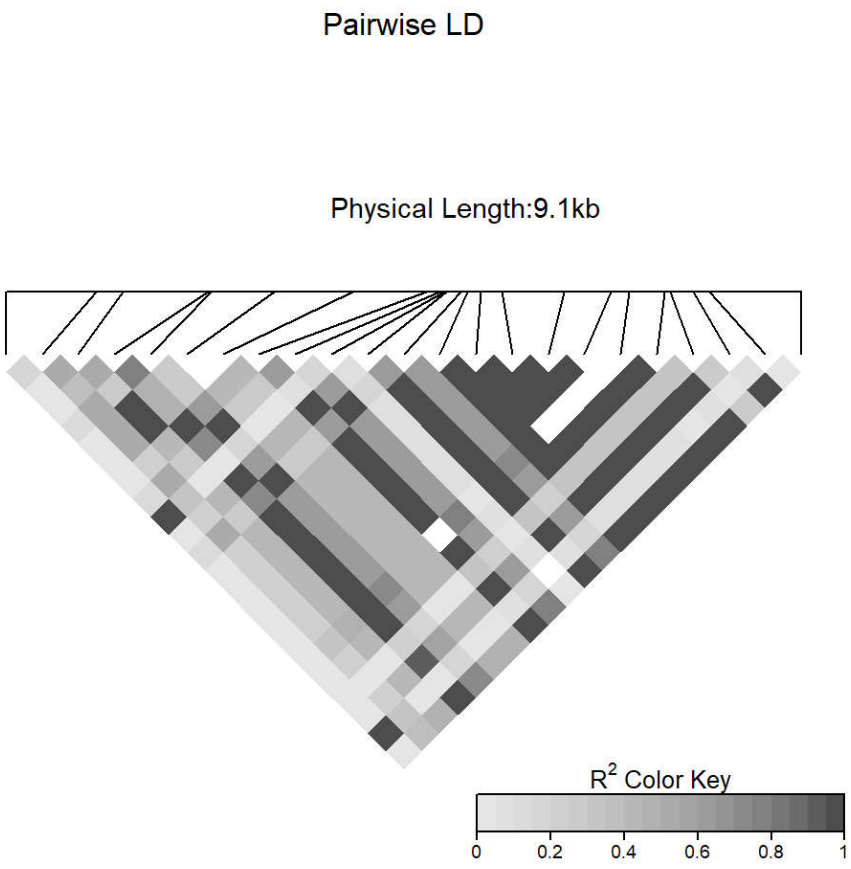
```
   if (dim(LDheatmap$LDmatrix)[1] != length(P)) {
    print("Length of vector not equal number of SNPs in LDheatmap")
    return()
  }
  flip <- !is.null(LDheatmap$flipVP)
  vp <- constructVP(LDheatmap$LDheatmapGrob, 0.43, flip)
  vp$height <- unit(height, "npc")
  vp$name <- "associationVP"
  if (is.null(ylim))
    ylim <- c(floor(min(P)), ceiling(max(P)))
  vp$yscale <- ylim
  vp$xscale <- c(min(LDheatmap$genetic.distances), max(LDheatmap$genetic.distances))
  xaxis <- linesGrob(x = vp$xscale, y = 0, default.units = "native",
                     name = "xaxis")
  yaxis <- linesGrob(x = min(LDheatmap$genetic.distances),
                     y = vp$yscale, default.units = "native", name = "yaxis")
  yaxisT <- yaxisGrob(name = "yaxis_ticks", gp = gpar(fontsize = 7))
  ylab <- textGrob(ylab, rot = 90, gp = gpar(fontsize = 9),
                   name = "yaxis_title", x = unit(min(LDheatmap$genetic.distances),
                                                  "native") - unit(10, "millimeters"))
  vpstack <- vp
  if (flip)
    vpstack <- vpStack(LDheatmap$flipVP, vp)
  association4 <- gTree(children = gList(xaxis, yaxis, yaxisT,
                                         ylab), name = "association4", vp = vpstack)
  if (type == "points" || type == "both") {
    graph_points <- pointsGrob(LDheatmap$genetic.distances, P, size = unit(2, "millime
ters"), name = "points",pch=16, gp=gpar(col="black"))
    association4 <- addGrob(association4, graph_points)
  }
  if (type == "lines" || type == "both") {
    graph_lines <- linesGrob(LDheatmap$genetic.distances,
                             P, default.units = "native", name = "lines")
    association4 <- addGrob(association4, graph_lines)
  }
  LDheatmap$LDheatmapGrob <- addGrob(LDheatmap$LDheatmapGrob,
                                     association4)
  LDheatmap$LDheatmapGrob <- moveTitles(LDheatmap$LDheatmapGrob,
                                        vp)
  return(LDheatmap)
}
environment(LDheatmap.addScatterplot_test4) <- asNamespace('LDheatmap')
```

```
constructVP <- function(LDheatmapGrob, location=0, flip) {

  x0 <- convertX(getGrob(LDheatmapGrob, "diagonal")[[1]][1], "npc", valueOnly=TRUE)
  x1 <- convertX(getGrob(LDheatmapGrob, "diagonal")[[1]][2], "npc", valueOnly=TRUE)
  y0 <- convertX(getGrob(LDheatmapGrob, "diagonal")[[2]][1], "npc", valueOnly=TRUE)
  y1 <- convertX(getGrob(LDheatmapGrob, "diagonal")[[2]][2], "npc", valueOnly=TRUE)
  map_len = sqrt((x1-x0)^2 + (y1-y0)^2)     # genetic map length in npc units

  g_height <- g_x0 <- g_y0 <- 0
  if(!is.null(getGrob(LDheatmapGrob, "transcripts"))) { # if gene track has been plott
ed
    transcriptsVP <- getGrob(LDheatmapGrob, "transcripts")$vp
    if (flip) transcriptsVP <- transcriptsVP[[2]]
    g_x0 <- convertX(transcriptsVP$x, "npc", valueOnly=TRUE)
    g_y0 <- convertX(transcriptsVP$y, "npc", valueOnly=TRUE)
    g_height <- convertX(transcriptsVP$height, "npc", valueOnly=TRUE)
  }

  r_height <- r_x0 <- r_y0 <- 0
  if(!is.null(getGrob(LDheatmapGrob, "recombRate"))) {  # if recombRate track has bee
n plotted
    recombRateVP <- getGrob(LDheatmapGrob, "recombRate")$vp
    if (flip) recombRateVP <- recombRateVP[[2]]
    r_x0 <- convertX(recombRateVP$x, "npc", valueOnly=TRUE)
    r_y0 <- convertX(recombRateVP$y, "npc", valueOnly=TRUE)
    r_height <- convertX(recombRateVP$height, "npc", valueOnly=TRUE)
  }

  m_height <- m_x0 <- m_y0 <- 0
  if(!is.null(getGrob(LDheatmapGrob, "association"))) { # if association scatterplot h
as been plotted
    assocVP <- getGrob(LDheatmapGrob, "association")$vp
    if (flip) assocVP <- assocVP[[2]]
    m_x0 <- convertX(assocVP$x, "npc", valueOnly=TRUE)
    m_y0 <- convertX(assocVP$y, "npc", valueOnly=TRUE)
    m_height <- convertX(assocVP$height, "npc", valueOnly=TRUE)
  }

  # Set the viewport
  if (!flip) {              # flip = FALSE
     angle <- 45
     genome_vp_just <- c("left", "top")
  }else {               # flip = TRUE
     angle <- 45
     genome_vp_just <- c("left", "bottom")
  }
  vp <- viewport(angle=angle, just=genome_vp_just, width=map_len,
    x=min(x0,  g_x0 - g_height*0.8, r_x0 - r_height*0.8, m_x0 - m_height*0.8) - locati
```

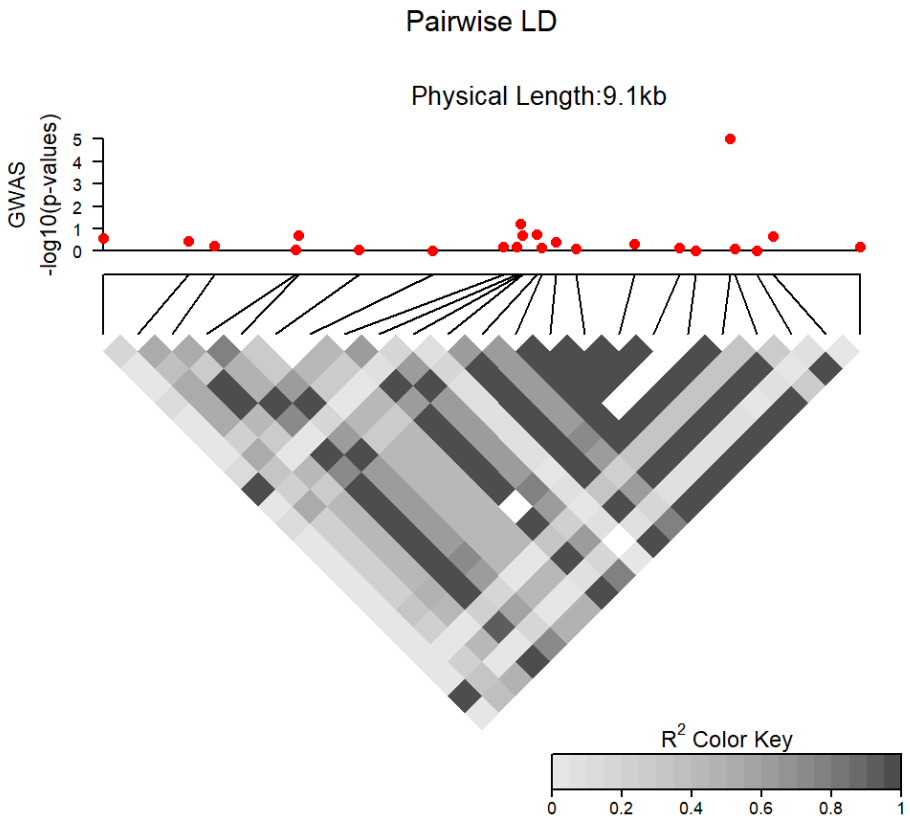
```
on,
    y=max(y0,  g_y0 + g_height*0.8, r_y0 + r_height*0.8, m_y0 + m_height*0.8) + locati
on)
  return (vp)
}

moveTitles <- function(LDheatmapGrob, vp) {

  genemap_title_y <- convertX(getGrob(LDheatmapGrob,"geneMap::title")$y,
                        "npc", valueOnly=TRUE)
  genemap_title_x <- convertX(getGrob(LDheatmapGrob,"geneMap::title")$x,
                        "npc", valueOnly=TRUE)
  flipVP <- getGrob(LDheatmapGrob,"geneMap::diagonal")$vp

  if (is.null(flipVP)) {                    # flip = FALSE
     if (genemap_title_y == 0.3 & genemap_title_x == 0.5)   # user used default settin
g
    LDheatmapGrob <- editGrob(LDheatmapGrob,"geneMap::title", y=unit(0.1, "npc"),
            x=unit(1.1, "npc"), just="right")
     grid.newpage()
     grid.draw(LDheatmapGrob)
     return(LDheatmapGrob)
  }

  # Get top of viewport coordinates in inches on the device
  pushViewport(LDheatmapGrob$vp)
  pushViewport(flipVP)
  vp_trans <- current.transform()
  temp <- c(
convertX(vp$x, "inches", valueOnly=TRUE) - convertX(vp$height, "inches",valueOnly=TRU
E)/sqrt(2),
convertX(vp$y, "inches", valueOnly=TRUE) + convertX(vp$height, "inches", valueOnly=TRU
E)/sqrt(2), 1)
  upViewport()
  tr <- temp %*% vp_trans  # (x, y, 1) on device

  # Get genemap title coordinates in inches on the device
  vp_trans1 <- current.transform()
  genemap_title_y_inch <- convertY(getGrob(LDheatmapGrob,"geneMap::title")$y, "inche
s", valueOnly=TRUE)
  temp1 <- c(0, genemap_title_y_inch, 1)
  tr1 <- temp1 %*% vp_trans1


  # Move gene map title # if necessary
  new_genemap_title_y_inch <- t(solve(t(vp_trans1), t(tr)))[1,2]
  new_genemap_title_y_npc <- convertY(unit(new_genemap_title_y_inch, "inches"), "np
c", valueOnly=TRUE) + 0.05
```

```
  LDheatmapGrob <- editGrob(LDheatmapGrob, "geneMap::title",
    y = unit(new_genemap_title_y_npc, "npc"), just=c("left","bottom"))


  # Move heat map title if necessary
  heatmap_title_y <- convertY(getGrob(LDheatmapGrob,"heatMap::title")$y, "npc", valueO
nly=TRUE)
  genemap_title_height <- convertHeight(grobHeight(getGrob(LDheatmapGrob,"geneMap::tit
le")),
        "npc", valueOnly=TRUE)

  if (heatmap_title_y < new_genemap_title_y_npc + genemap_title_height*3) {
     new_heatmap_title_y <- new_genemap_title_y_npc + genemap_title_height*3
     LDheatmapGrob <- editGrob(LDheatmapGrob, "heatMap::title",
        y = unit(new_heatmap_title_y, "npc"))
  }

  drawLDheatmapGrob(LDheatmapGrob)
  return(LDheatmapGrob)
}

drawLDheatmapGrob <- function(LDheatmapGrob) {
  heatmap_title_y <- convertY(getGrob(LDheatmapGrob,"heatMap::title")$y, "npc", valueO
nly=TRUE)
  vp = viewport(height=1/heatmap_title_y, width=1, y=0.05, just="bottom",
        gp=gpar(cex=1/heatmap_title_y), name="container")
  grid.newpage()
  pushViewport(vp)
  grid.draw(LDheatmapGrob)
  popViewport()
}
```

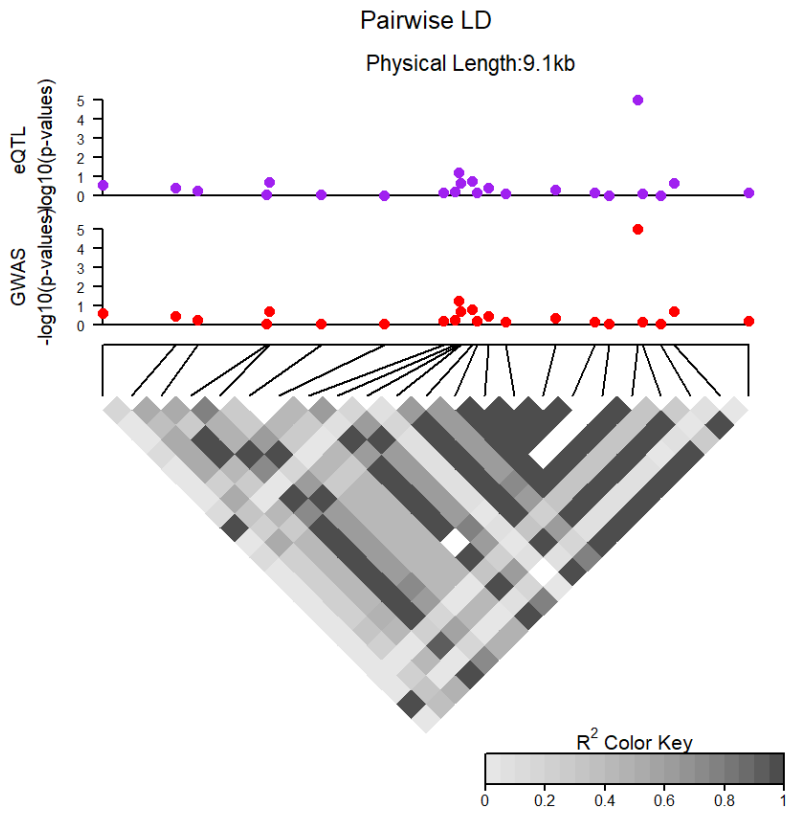
```
data(GIMAP5.CEU)
load(system.file("extdata/addTracks.RData",package="LDheatmap"))
#
ll2 <- LDheatmap(GIMAP5.CEU$snp.data,GIMAP5.CEU$snp.support$Position,flip=TRUE)
# llGenes <- LDheatmap.addGenes(ll2, chr="chr7", genome="hg18")
#
#  grid.newpage()
# grid.draw(llGenes$LDheatmapGrob)
# llGenesRecomb <- LDheatmap.addRecombRate(llGenes, chr="chr7", genome="hg18")
# grid.newpage()
# grid.draw(llGenesRecomb$LDheatmapGrob)

set.seed(1)
atests<-runif(nrow(GIMAP5.CEU$snp.support))
names(atests)<-rownames(GIMAP5.CEU$snp.support)
atests["rs6598"]<-1e-5



llGenesRecombScatter<-LDheatmap.addScatterplot_test1(ll2,-log10(atests), ylab="GWAS\n
-log10(p-values)")
```



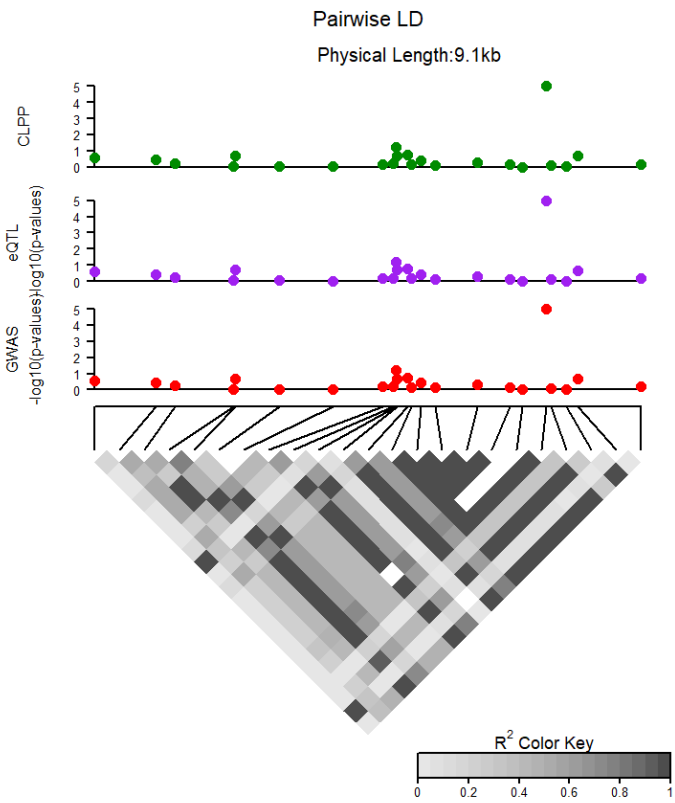
```
llGenesRecombScatter2<-LDheatmap.addScatterplot_test2(llGenesRecombScatter,-log10(atests), ylab="eQTL \n-log10(p-values)")
```

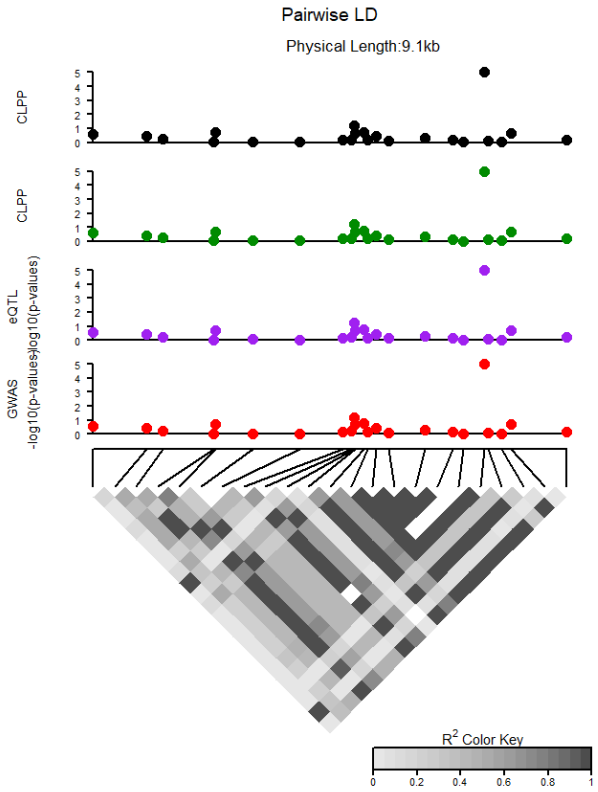



```
#pdf('tmp.pdf',width = 10,height = 8)

llGenesRecombScatter3<-LDheatmap.addScatterplot_test3(llGenesRecombScatter2,-log10(atests), ylab="CLPP")
```




```
llGenesRecombScatter4<-LDheatmap.addScatterplot_test4(llGenesRecombScatter3,-log10(atests), ylab="CLPP")
```


Pairwise LD
Physical Length:9.1kb
CLPP
eQTL
GWAS
-log10(p-values)log10(p-values)
R2 Color Key
0
0.2
0.4
0.6
0.8
1


Pairwise LD
Physical Length:9.1kb
CLPP
CLPP
eQTL
GWAS
-log10(p-values)log10(p-values)
R2 Color Key
0
0.2
0.4
0.6
0.8
1