Annotated Manhattan plots and QQ plots for GWAS using R, Revisited

Stephen D. Turner

http://www.StephenTurner.us/

Monday, April 25, 2011

Last year I showed you how to create manhattan plots, and later how to highlight regions of interest, using ggplot2 in R. The code was slow, required a lot of memory, and was difficult to maintain and modify.

I finally found time to rewrite the code using base graphics rather than ggplot2. The code is now much faster, and if you're familiar with base R's plot options and graphical parameters, most of these can now be passed to the functions to tweak the plots' appearance. The code also behaves differently depending on whether you have results for one or more than one chromosome.

Here's a quick demo.

First, either copy and paste the code from GitHub, or run the following command in R to source the function from the web:

```
source("http://www.StephenTurner.us/qqman.r")
```

Next, load some GWAS results, and take a look at the relevant columns. This is standard output from PLINK's --assoc option. This may take a minute or so (~ 20MB file):

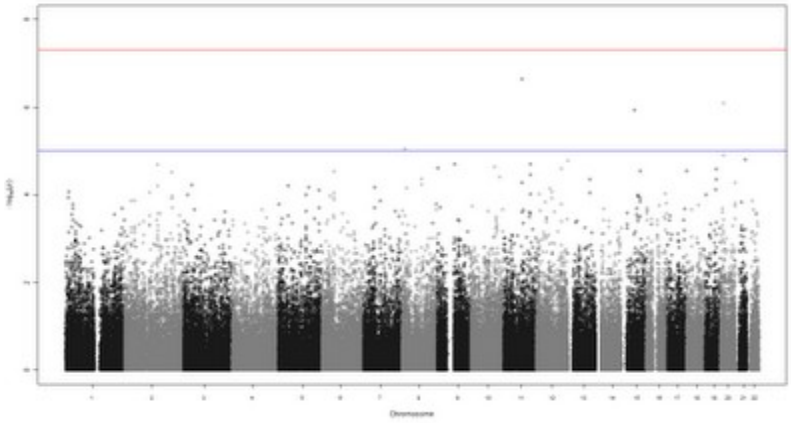
```
results <-
read.table("http://www.StephenTurner.us/plinkresults.assoc",T)

head(subset(results, select=c(SNP, CHR, BP, P)))
```

The manhattan function assumes you have columns named SNP, CHR, BP, and P, corresponding to the SNP name (rs number), chromosome number, genomic coordinate, and p-value. Missing values (where regression failed to converge, for instance) should be recoded NA before reading into R. Do this with a quick sed command. Here's what the data looks like:

```
         SNP CHR        BP       P
1 rs10495434   1 235800006 0.62220
2  rs6689417   1  46100028 0.06195
3  rs3897197   1 143700035 0.10700
4  rs2282450   1 202300047 0.47280
5   rs567279   1  66400050      NA
6 rs11208515   1  64900051 0.53430
```

Now, create a basic manhattan plot (click the image to enlarge):

```
manhattan(results)
```



If you type args(manhattan) you can see the options you can set. Here are a few:

colors: this is a character vector specifying the colors to cycle through for coloring each point. Here's a PDF chart of R's color names.

ymax: this is the y-axis limit. If ymax="max" (default), the y-axis will always be a little bit higher than the most significant -log10(p-value). Otherwise you can set this value yourself.

cex.x.axis: this can be used to shrink the x-axis labels by setting this value less than 1. This is handy if some of the tick labels aren't showing up because the plot region is too small.

limitchromosomes: you can limit which chromosomes you want to display. By default this restricts the plot to chromosomes 1-23(x).

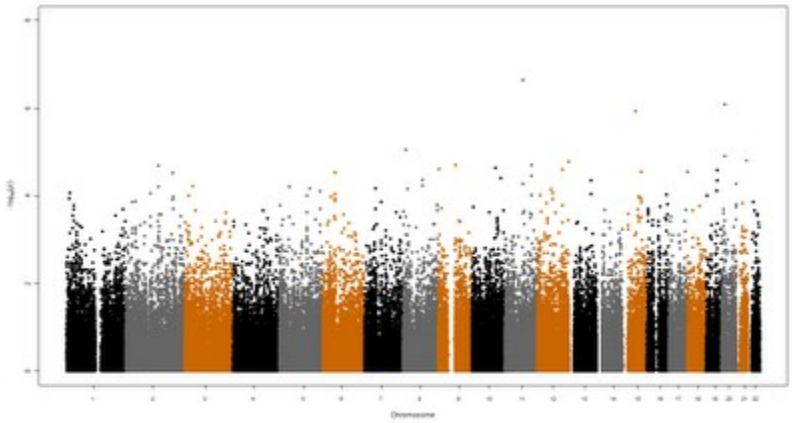
suggestiveline and **genomewideline**: set these to FALSE if you don't want threshold lines, or change the thresholds yourself.

annotate: by default this is undefined. If you supply a character vector of SNP names (e.g. rs numbers), any SNPs in the results data frame that also show up here will be highlighted in green by default. example below.

... : The dot-dot-dot means you can pass most other plot or graphical parameters to these functions (e.g. main, cex, pch, etc).

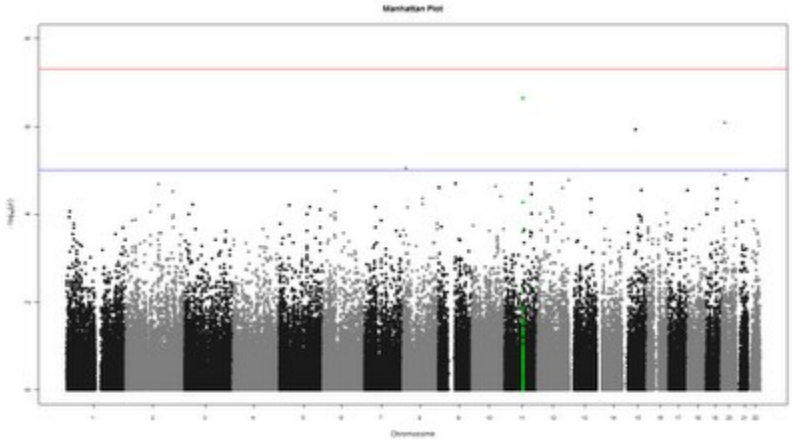
Make a better looking manhattan plot. Change the plot colors, point shape, and remove the threshold lines:

```
manhattan(results, colors=c("black","#666666","#CC6600"),
pch=20, genomewideline=F, suggestiveline=F)
```



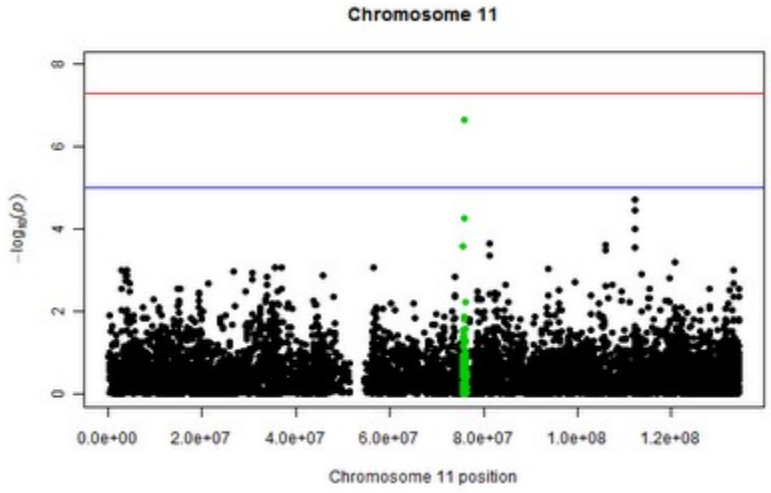
Now, read in a text file with SNP names that you want to highlight, then make a manhattan plot highlighting those SNPs, and give the plot a title:

```
snps_to_highlight <-
scan("http://www.StephenTurner.us/snps.txt", character())

manhattan(results, annotate=snps_to_highlight, pch=20,
main="Manhattan Plot")
```



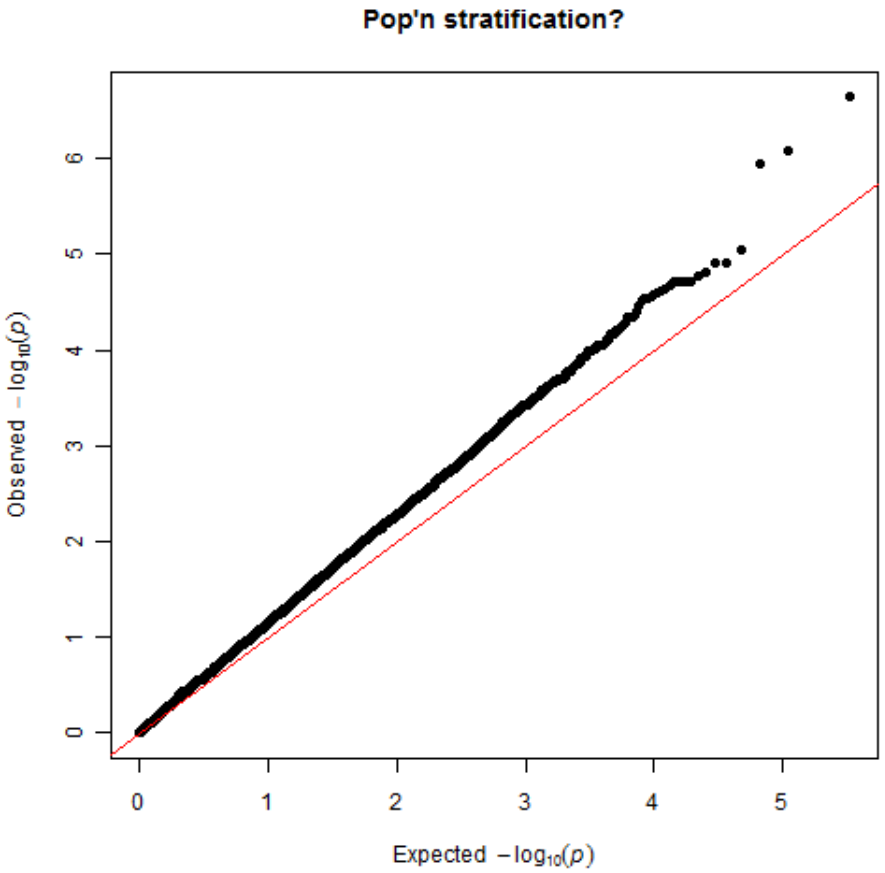
Finally, zoom in and plot only the results for chromosome 11, still highlighting those results. Notice that the x-axis changes from chromosome to genomic coordinate.

```
manhattan(subset(results, CHR==11), pch=20,
annotate=snps_to_highlight, main="Chromosome 11")
```



Finally, make a quantile-quantile plot of the p-values. To make a basic qq-plot of the p-values, pass the qq() function a vector of p-values:

```
qq(results$P)
```



Perhaps we should have made the qq-plot first, as it looks like we might have some unaccounted-for population stratification or other bias.

The code should run much faster and use less memory than before. All the old functions that use ggplot2 are still available, now prefixed with "gg." Please feel free to use, modify, and redistribute, but kindly link back to this post. Here's the code:

```
# Stephen Turner
# http://StephenTurner.us/
# http://GettingGeneticsDone.blogspot.com/
# See license at http://gettinggeneticsdone.blogspot.com/p/copyright.html

# Last updated: Tuesday, April19, 2011
# R code for making manhattan plots and QQ plots from plink output files.
# manhattan() with GWAS data this can take a lot of memory, recommended for
use on 64bit machines only, for now.
# Altnernatively, use bmanhattan() , i.e., base manhattan. uses base
graphics. way faster.


## This is for testing purposes.
# set.seed(42)
# nchr=23
# nsnps=1000
# d=data.frame(
#     SNP=sapply(1:(nchr*nsnps), function(x) paste("rs",x,sep='')),
#     CHR=rep(1:nchr,each=nsnps),
#     BP=rep(1:nsnps,nchr),
#       P=runif(nchr*nsnps)
# )
# annotatesnps <- d$SNP[7550:7750]

# manhattan plot using base graphics
manhattan = function(dataframe, colors=c("gray10", "gray50"), ymax="max",
cex.x.axis=1, limitchromosomes=1:23, suggestiveline=-log10(1e-5),
genomewideline=-log10(5e-8), annotate=NULL, ...) {

    d=dataframe
    if (!("CHR" %in% names(d) & "BP" %in% names(d) & "P" %in% names(d)))
stop("Make sure your data frame contains columns CHR, BP, and P")

    if (any(limitchromosomes)) d=d[d$CHR %in% limitchromosomes, ]
    d=subset(na.omit(d[order(d$CHR, d$BP), ]), (P>0 & P<=1)) # remove na's,
sort, and keep only 0<P<=1
    d$logp = -log10(d$P)
    d$pos=NA
    ticks=NULL
    lastbase=0
    colors <- rep(colors,max(d$CHR))[1:max(d$CHR)]
    if (ymax=="max") ymax<-ceiling(max(d$logp))
    if (ymax<8) ymax<-8

    numchroms=length(unique(d$CHR))
    if (numchroms==1) {
```

```
        d$pos=d$BP
        ticks=floor(length(d$pos))/2+1
    } else {
        for (i in unique(d$CHR)) {
                if (i==1) {
                        d[d$CHR==i, ]$pos=d[d$CHR==i, ]$BP
                } else {
                        lastbase=lastbase+tail(subset(d,CHR==i-1)$BP, 1)
                        d[d$CHR==i, ]$pos=d[d$CHR==i, ]$BP+lastbase
                }
                ticks=c(ticks, d[d$CHR==i, ]$pos[floor(length(d[d$CHR==i, ]
$pos)/2)+1])
        }
    }

    if (numchroms==1) {
        with(d, plot(pos, logp, ylim=c(0,ymax), ylab=expression(-log[10]
(italic(p))), xlab=paste("Chromosome",unique(d$CHR),"position"), ...))
    }   else {
        with(d, plot(pos, logp, ylim=c(0,ymax), ylab=expression(-log[10]
(italic(p))), xlab="Chromosome", xaxt="n", type="n", ...))
        axis(1, at=ticks, lab=unique(d$CHR), cex.axis=cex.x.axis)
        icol=1
        for (i in unique(d$CHR)) {
            with(d[d$CHR==i, ],points(pos, logp, col=colors[icol], ...))
            icol=icol+1
        }
    }

    if (!is.null(annotate)) {
        d.annotate=d[which(d$SNP %in% annotate), ]
        with(d.annotate, points(pos, logp, col="green3", ...))
    }

    if (suggestiveline) abline(h=suggestiveline, col="blue")
    if (genomewideline) abline(h=genomewideline, col="red")
}

# Base graphics qq plot
qq = function(pvector, ...) {
    if (!is.numeric(pvector)) stop("D'oh! P value vector is not numeric.")
        pvector <- pvector[!is.na(pvector) & pvector<1 & pvector>0]
    o = -log10(sort(pvector,decreasing=F))
        #e = -log10( 1:length(o)/length(o) )
        e = -log10( ppoints(length(pvector) ))
        plot(e,o,pch=19,cex=1, xlab=expression(Expected~~-log[10](italic(p))),
ylab=expression(Observed~~-log[10](italic(p))), xlim=c(0,max(e)),
ylim=c(0,max(o)), ...)
        abline(0,1,col="red")
}


### OLD GGPLOT2 CODE ###

# manhattan plot using ggplot2
```

```
gg.manhattan = function(dataframe, title=NULL, max.y="max", suggestiveline=0,
genomewideline=-log10(5e-8), size.x.labels=9, size.y.labels=10, annotate=F,
SNPlist=NULL) {
library(ggplot2)
    if (annotate & is.null(SNPlist)) stop("You requested annotation but
provided no SNPlist!")
        d=dataframe
        #limit to only chrs 1-23?
        d=d[d$CHR %in% 1:23, ]
        if ("CHR" %in% names(d) & "BP" %in% names(d) & "P" %in% names(d) ) {
                d=na.omit(d)
                d=d[d$P>0 & d$P<=1, ]
                d$logp = -log10(d$P)
                d$pos=NA
                ticks=NULL
                lastbase=0
                #new 2010-05-10
                numchroms=length(unique(d$CHR))
                if (numchroms==1) {
                        d$pos=d$BP
                } else {

                        for (i in unique(d$CHR)) {
                                if (i==1) {
                                        d[d$CHR==i, ]$pos=d[d$CHR==i, ]$BP
                                }       else {
                                        lastbase=lastbase+tail(subset(d,CHR==i-
1)$BP, 1)
                                        d[d$CHR==i, ]$pos=d[d$CHR==i, ]
$BP+lastbase
                                }
                                ticks=c(ticks, d[d$CHR==i, ]
$pos[floor(length(d[d$CHR==i, ]$pos)/2)+1])
                        }
                        ticklim=c(min(d$pos),max(d$pos))

                }
                mycols=rep(c("gray10","gray60"),max(d$CHR))
                if (max.y=="max") maxy=ceiling(max(d$logp)) else maxy=max.y
                if (maxy<8) maxy=8
                if (annotate) d.annotate=d[as.numeric(substr(d$SNP,3,100)) %in
% SNPlist, ]
                if (numchroms==1) {
                        plot=qplot(pos,logp,data=d,ylab=expression(-log[10]
(italic(p))), xlab=paste("Chromosome",unique(d$CHR),"position"))
                }       else {
                        plot=qplot(pos,logp,data=d, ylab=expression(-log[10]
(italic(p))) , colour=factor(CHR))
                        plot=plot+scale_x_continuous(name="Chromosome",
breaks=ticks, labels=(unique(d$CHR)))
                        plot=plot+scale_y_continuous(limits=c(0,maxy),
breaks=1:maxy, labels=1:maxy)
                        plot=plot+scale_colour_manual(value=mycols)
                }
                if (annotate)  plot=plot + geom_point(data=d.annotate,
colour=I("green3"))
                plot=plot + opts(legend.position = "none")
```

```
                plot=plot + opts(title=title)
                plot=plot+opts(
                        panel.background=theme_blank(),
                        panel.grid.minor=theme_blank(),
                        axis.text.x=theme_text(size=size.x.labels,
colour="grey50"),
                        axis.text.y=theme_text(size=size.y.labels,
colour="grey50"),
                        axis.ticks=theme_segment(colour=NA)
                )
                if (suggestiveline)
plot=plot+geom_hline(yintercept=suggestiveline,colour="blue", alpha=I(1/3))
                if (genomewideline)
plot=plot+geom_hline(yintercept=genomewideline,colour="red")
                plot
        }       else {
                stop("Make sure your data frame contains columns CHR, BP, and
P")
        }
}

gg.qq = function(pvector, title=NULL, spartan=F) {
        library(ggplot2)
        o = -log10(sort(pvector,decreasing=F))
        #e = -log10( 1:length(o)/length(o) )
        e = -log10( ppoints(length(pvector) ))
        plot=qplot(e,o, xlim=c(0,max(e)), ylim=c(0,max(o))) +
stat_abline(intercept=0,slope=1, col="red")
        plot=plot+opts(title=title)
        plot=plot+scale_x_continuous(name=expression(Expected~~-log[10]
(italic(p))))
        plot=plot+scale_y_continuous(name=expression(Observed~~-log[10]
(italic(p))))
        if (spartan) plot=plot+opts(panel.background=theme_rect(col="grey50"),
panel.grid.minor=theme_blank())
        plot
}

gg.qqman = function(data="plinkresults") {
        myqqplot = ggqq(data$P)
        mymanplot = ggmanhattan(data)
        ggsave(file="qqplot.png",myqqplot,w=5,h=5,dpi=100)
        ggsave(file="manhattan.png",mymanplot,width=12,height=9,dpi=100)
}

gg.qqmanall= function(command="ls *assoc") {
        filelist=system(command,intern=T)
        datalist=NULL
        for (i in filelist) {datalist[[i]]=read.table(i,T)}
        highestneglogp=ceiling(max(sapply(datalist, function(df) max(na.omit(-
log10(df$P))))))
        print(paste("Highest -log10(P) = ",highestneglogp),quote=F)
        start=Sys.time()
        for (i in names(datalist)) {
                myqqplot=ggqq(datalist[[i]]$P, title=i)
                ggsave(file=paste("qqplot-",    i, ".png", sep=""),myqqplot,
width=5, height=5,dpi=100)
```

```
                mymanplot=ggmanhattan(datalist[[i]], title=i,
max.y=highestneglogp)
                ggsave(file=paste("manhattan-", i, ".png",
sep=""),mymanplot,width=12,height=9,dpi=100)
        }
        end=Sys.time()
        print(elapsed<-end-start)
}
```

view raw ggman.r This Gist brought to you by GitHub.