test manhattan, umichigan version

Based on https://genome.sph.umich.edu/wiki/Code_Sample:_Generating_Manhattan_Plots_in_R

```
library(lattice)
manhattan.plot<-function(chr, pos, pvalue,
    sig.level=NA, annotate=NULL, ann.default=list(),
    should.thin=T, thin.pos.places=2, thin.logp.places=2,
    xlab="Chromosome", ylab=expression(-log[10](p-value)),
    col=c("gray","darkgray"), panel.extra=NULL, pch=20, cex=0.8,...) {

    if (length(chr)==0) stop("chromosome vector is empty")
    if (length(pos)==0) stop("position vector is empty")
    if (length(pvalue)==0) stop("pvalue vector is empty")

    #make sure we have an ordered factor
    if(!is.ordered(chr)) {
        chr <- ordered(chr)
    } else {
        chr <- chr[,drop=T]
    }

    #make sure positions are in kbp
    if (any(pos>1e6)) pos<-pos/1e6;

    #calculate absolute genomic position
    #from relative chromosomal positions
    posmin <- tapply(pos,chr, min);
    posmax <- tapply(pos,chr, max);
    posshift <- head(c(0,cumsum(posmax)),-1);
    names(posshift) <- levels(chr)
    genpos <- pos + posshift[chr];
    getGenPos<-function(cchr, cpos) {
        p<-posshift[as.character(cchr)]+cpos
        return(p)
    }

    #parse annotations
    grp <- NULL
    ann.settings <- list()
    label.default<-list(x="peak",y="peak",adj=NULL, pos=3, offset=0.5,
        col=NULL, fontface=NULL, fontsize=NULL, show=F)
    parse.label<-function(rawval, groupname) {
        r<-list(text=groupname)
        if(is.logical(rawval)) {
            if(!rawval) {r$show <- F}
        } else if (is.character(rawval) || is.expression(rawval)) {
            if(nchar(rawval)>=1) {
                r$text <- rawval
```

```
        }
    } else if (is.list(rawval)) {
        r <- modifyList(r, rawval)
    }
    return(r)
}

if(!is.null(annotate)) {
    if (is.list(annotate)) {
        grp <- annotate[[1]]
    } else {
        grp <- annotate
    }
    if (!is.factor(grp)) {
        grp <- factor(grp)
    }
} else {
    grp <- factor(rep(1, times=length(pvalue)))
}

ann.settings<-vector("list", length(levels(grp)))
ann.settings[[1]]<-list(pch=pch, col=col, cex=cex, fill=col, label=label.default)

if (length(ann.settings)>1) {
    lcols<-trellis.par.get("superpose.symbol")$col
    lfills<-trellis.par.get("superpose.symbol")$fill
    for(i in 2:length(levels(grp))) {
        ann.settings[[i]]<-list(pch=pch,
            col=lcols[(i-2) %% length(lcols) +1 ],
            fill=lfills[(i-2) %% length(lfills) +1 ],
            cex=cex, label=label.default);
        ann.settings[[i]]$label$show <- T
    }
    names(ann.settings)<-levels(grp)
}
for(i in 1:length(ann.settings)) {
    if (i>1) {ann.settings[[i]] <- modifyList(ann.settings[[i]], ann.default)}
    ann.settings[[i]]$label <- modifyList(ann.settings[[i]]$label,
        parse.label(ann.settings[[i]]$label, levels(grp)[i]))
}
if(is.list(annotate) && length(annotate)>1) {
    user.cols <- 2:length(annotate)
    ann.cols <- c()
    if(!is.null(names(annotate[-1])) && all(names(annotate[-1])!="")) {
        ann.cols<-match(names(annotate)[-1], names(ann.settings))
    } else {
        ann.cols<-user.cols-1
    }
    for(i in seq_along(user.cols)) {
        if(!is.null(annotate[[user.cols[i]]]$label)) {
            annotate[[user.cols[i]]]$label<-parse.label(annotate[[user.cols[i]]]$label,
                levels(grp)[ann.cols[i]])
        }
```

```
        ann.settings[[ann.cols[i]]]<-modifyList(ann.settings[[ann.cols[i]]],
            annotate[[user.cols[i]]])
    }
}
rm(annotate)

#reduce number of points plotted
if(should.thin) {
    thinned <- unique(data.frame(
        logp=round(-log10(pvalue),thin.logp.places),
        pos=round(genpos,thin.pos.places),
        chr=chr,
        grp=grp)
    )
    logp <- thinned$logp
    genpos <- thinned$pos
    chr <- thinned$chr
    grp <- thinned$grp
    rm(thinned)
} else {
    logp <- -log10(pvalue)
}
rm(pos, pvalue)
gc()

#custom axis to print chromosome names
axis.chr <- function(side,...) {
    if(side=="bottom") {
        panel.axis(side=side, outside=T,
            at=((posmax+posmin)/2+posshift),
            labels=levels(chr),
            ticks=F, rot=0,
            check.overlap=F
        )
    } else if (side=="top" || side=="right") {
        panel.axis(side=side, draw.labels=F, ticks=F);
    }
    else {
        axis.default(side=side,...);
    }
 }

#make sure the y-lim covers the range (plus a bit more to look nice)
prepanel.chr<-function(x,y,...) {
    A<-list();
    maxy<-ceiling(max(y, ifelse(!is.na(sig.level), -log10(sig.level), 0)))+.5;
    A$ylim=c(0,maxy);
    A;
}

xyplot(logp~genpos, chr=chr, groups=grp,
    axis=axis.chr, ann.settings=ann.settings,
    prepanel=prepanel.chr, scales=list(axs="i"),
```

```
        panel=function(x, y, ..., getgenpos) {
            if(!is.na(sig.level)) {
                #add significance line (if requested)
                panel.abline(h=-log10(sig.level), lty=2);
            }
            panel.superpose(x, y, ..., getgenpos=getgenpos);
            if(!is.null(panel.extra)) {
                panel.extra(x,y, getgenpos, ...)
            }
        },
        panel.groups = function(x,y,..., subscripts, group.number) {
            A<-list(...)
            #allow for different annotation settings
            gs <- ann.settings[[group.number]]
            A$col.symbol <- gs$col[(as.numeric(chr[subscripts])-1) %% length(gs$col) + 1]
            A$cex <- gs$cex[(as.numeric(chr[subscripts])-1) %% length(gs$cex) + 1]
            A$pch <- gs$pch[(as.numeric(chr[subscripts])-1) %% length(gs$pch) + 1]
            A$fill <- gs$fill[(as.numeric(chr[subscripts])-1) %% length(gs$fill) + 1]
            A$x <- x
            A$y <- y
            do.call("panel.xyplot", A)
            #draw labels (if requested)
            if(gs$label$show) {
                gt<-gs$label
                names(gt)[which(names(gt)=="text")]<-"labels"
                gt$show<-NULL
                if(is.character(gt$x) | is.character(gt$y)) {
                    peak = which.max(y)
                    center = mean(range(x))
                    if (is.character(gt$x)) {
                        if(gt$x=="peak") {gt$x<-x[peak]}
                        if(gt$x=="center") {gt$x<-center}
                    }
                    if (is.character(gt$y)) {
                        if(gt$y=="peak") {gt$y<-y[peak]}
                    }
                }
                if(is.list(gt$x)) {
                    gt$x<-A$getgenpos(gt$x[[1]],gt$x[[2]])
                }
                do.call("panel.text", gt)
            }
        },
        xlab=xlab, ylab=ylab,
        panel.extra=panel.extra, getgenpos=getGenPos, ...
    );
}
```

```
list.files()
```

```
## [1] "gwas-t2m-meta0613-positions-pq.csv" "michigan.Rmd"
## [3] "msa_position_2_ref_position.csv"    "plot_gisaid_gwas-t2m.pdf"
## [5] "plot_gisaid_gwas-t2m.Rmd"           "README.md"
```

```r
tb = read.csv2( "gwas-t2m-meta0613-positions-pq.csv", stringsAsFactors = F)
#tb
```

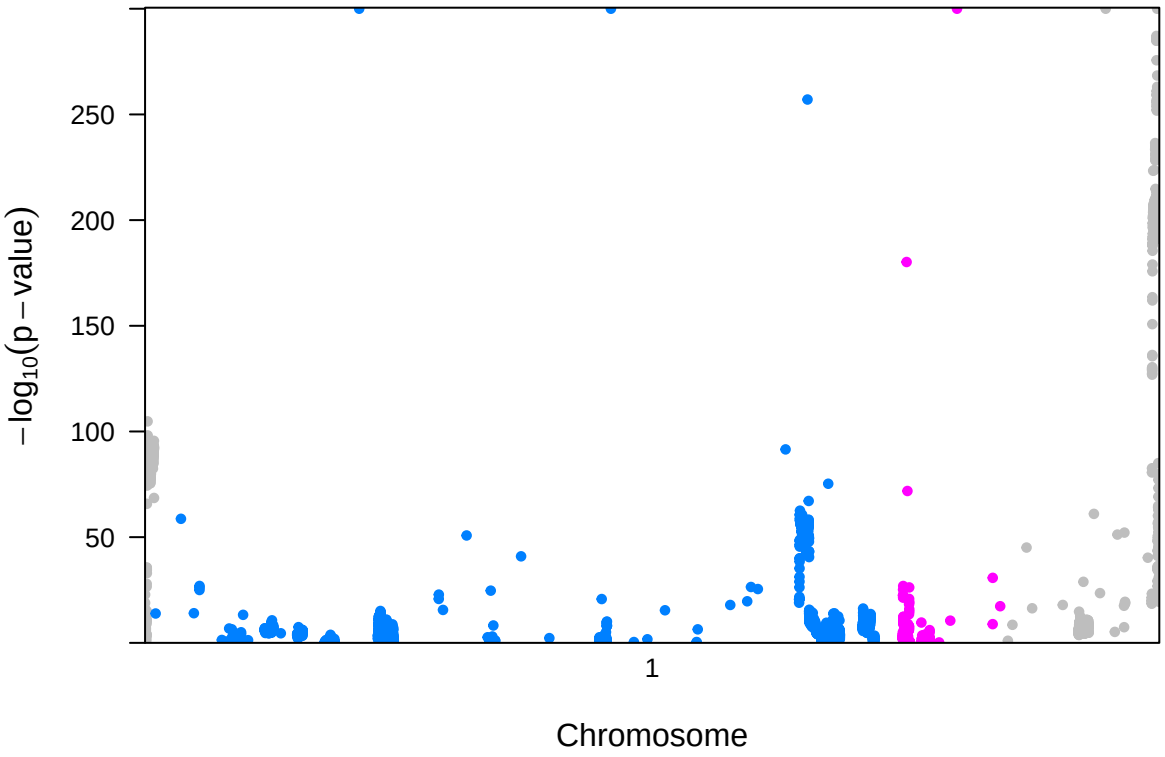
```r
tb_cov = read.csv("msa_position_2_ref_position.csv", row.names = 1)
tb$p = as.numeric(tb$p)
tb$ref_position = tb_cov$ref_positions[ match(tb$msa_position, tb_cov$msa_positions  )]
tb$p[tb$p==0] = 1E-300
summary(tb)
```

```
##   R_position         msa_position         p                  q
##  Length:4210        Min.   :  582   Min.   :0.0000000   Length:4210
##  Class :character   1st Qu.: 8827   1st Qu.:0.0000000   Class :character
##  Mode  :character   Median :28652   Median :0.0000039   Mode  :character
##                     Mean   :23641   Mean   :0.0789487
##                     3rd Qu.:34110   3rd Qu.:0.0581378
##                     Max.   :48402   Max.   :0.9983663
##   ref_position
##  Min.   :    0
##  1st Qu.: 5500
##  Median :19488
##  Mean   :15095
##  3rd Qu.:21489
##  Max.   :29890
```

```r
tb$chr=1
```

```r
# spike 21563   25384

#make annotation factor
ann <- rep(1, length(tb$p))
#print(ann)
ann[with(tb, chr==1 & ref_position>=266 & ref_position< 21555)]<-2
ann[with(tb, chr==1 & ref_position>=21563 & ref_position< 25384)]<-3
#print(ann)
ann<-factor(ann, levels=1:3, labels=c("", "ORF1ab", "Spike"))
#manhattan.plot(tb$chr, tb$ref_position, tb$p)
#draw plot with annotation
manhattan.plot(tb$chr, tb$ref_position, tb$p, annotate=ann)
```


250
200
150
100
50
$-\log_{10}(p-value)$
1
Chromosome