Gene-based p-values:
To use STAMS, you need to first calculate gene-based p-values. Plink has options for gene-based tests, or you can use another gene-based test. In my work, I prefer VEGAS v1.0, which has an online tool at http://gump.qimr.edu.au/VEGAS/ or can be downloaded and run locally. To use VEGAS, you need to have a homogenous population with a good reference in HapMap.

Dependencies:
My recommendation is to download igraph 0.7.1 from this website: https://cran.r-project.org/src/contrib/Archive/igraph/

This is an older version of igraph. It works very well with STAMS version 1.2, which works universally on all cluster set ups.

If you cannot install igraph 0.7.1 (or earlier), then install igraph 1.0, and use the parallel version of STAMS, version 1.8. There were HUGE overhauls in the data representation and UI between igraph 0.7.1 and igraph 1.0, so don't try running STAMS 1.8 with the old igraph or STAMS 1.2 with the new version. For what it's worth, I use the 0.7.1/STAMS_1.2 setup as my "production line." The parallel version works very well and very fast on small test cases, but takes a little more finagling to get the cluster setup in R, and hasn't been tested on large edgesets.

Working Example:
Next, work through the example in the STAMS documentation interactively so that you have some sense of the workflow. It runs in a few minutes on a laptop. (There's a working example in the STAMS 1.8 help file too.)

Open R and type these things at the prompt (omit the ">"):

```
>install.packages("path/to/stams_1.2.tar.gz", type="source",
repos=NULL)
>library(STAMS)
>?STAMS
```

to pull up the help file. It has a working example that you can run:

```
>data("mapped_data")
```

at the prompt to load the example data. Then just follow along with the help file.

Mapping your data to STRING:

When that works, it is time to map *your* data to STRING ids. There is a function in the STRINGdb package for R that will help with this. Here's how I do it, although you will have to change the path and filenames:

```
input_file=paste("/home/shillenm/wtccc_subgraph/pvalues_all_snps
_", disease, "_vegas.txt-genebased.out.txt", sep='')

print(input_file)
##   LOAD PVALUES
string_db <- STRINGdb$new()
gene_values=read.table(file=input_file, header=TRUE, sep=' ')
# get rid of pvalue = 0 lines and replace with upper bound
gene_values$Pvalue=pmax(gene_values$Pvalue, 1/gene_values$nSims)

# remove pvalue = 1
print("warning, removing genes with a pvalue = 1")
print(length(which(gene_values$Pvalue ==1)))
gene_values=gene_values[-which(gene_values$Pvalue == 1),]

gene_mapped = string_db$map( gene_values, "Gene",
removeUnmappedRows = TRUE )
# check mapping
data3=string_db$add_proteins_description(gene_mapped)
data4=data3[which(data3$Gene==toupper(data3$preferred_name)),]
ensg_genes=data3[grep('ENSG',data3$preferred_name),]
data4=rbind(data4,ensg_genes)
data_mapped=data4[,-which(names(data4)=='annotation')]

# TO SAVE THIS WORK SO THAT YOU CAN LOAD IT UP NEXT TIME
save.image('your_mapped_data.Rdata')
```

I've found that the mapping function is not perfect, and so I double check the mapping by making sure that the gene name is the same as the preferred name. Then there's a whole class of genes that have ENSG in their preferred name, and most of those seem to map correctly out of the box, so I include those too. Some of the genes won't map to STRING_ids, and that's okay.

Once you've mapped your data, use the same workflow as the example code in the STAMS helpfile. It will take 4-5 days to run a full experiment with STAMS 1.2. I have never run it on my laptop—always as a job submitted to a cluster. For STAMS 1.2, request 1 node and at least 32 gigs of memory. STAMS 1.8 will take as many nodes as you give it, but you'll have to set up the cluster in R and pass it to the search function. If using the SLURM job manager, there's a great RSLURM package that makes this easy.

I recommend saving the workspace after searching for lambda, after running the stams_search, and after completion.

Use the chooseModule function to identify the top 1% of modules, and then use those modules as candidate modules for followup study. There are various visualization functions available (see the helpfile) to look at the resulting modules.