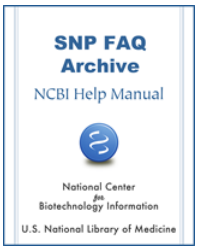
U.S. National Library of Medicine
National Center for Biotechnology Information

NLM Citation: SNP FAQ Archive [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2005-. Using GST to Study Genome Wide Association (GWAS) Data. 2008 Apr 23 [Updated 2010 Jun 15].
Bookshelf URL: https://www.ncbi.nlm.nih.gov/books/



Using GST to Study Genome Wide Association (GWAS) Data

Created: April 23, 2008; Updated: June 15, 2010.

What kind of studies will GST allow me to perform with Genome-Wide Association (GWAS) Data?

There are a number of planned features we intend to include in the GST, such as generation of LD-graphs based on uploaded GWAS data, but these features will not be available anytime soon. For now, the only thing that GST will allow you to do is to display pre-computed results in Gbench. (**04/23/08**)

Loading GWAS Data to GST

GWAS File Types defined

What is the difference between GWAS CHPB, Analysis files, and CHPA files, and which of these would I load into Gbench for study with GST?

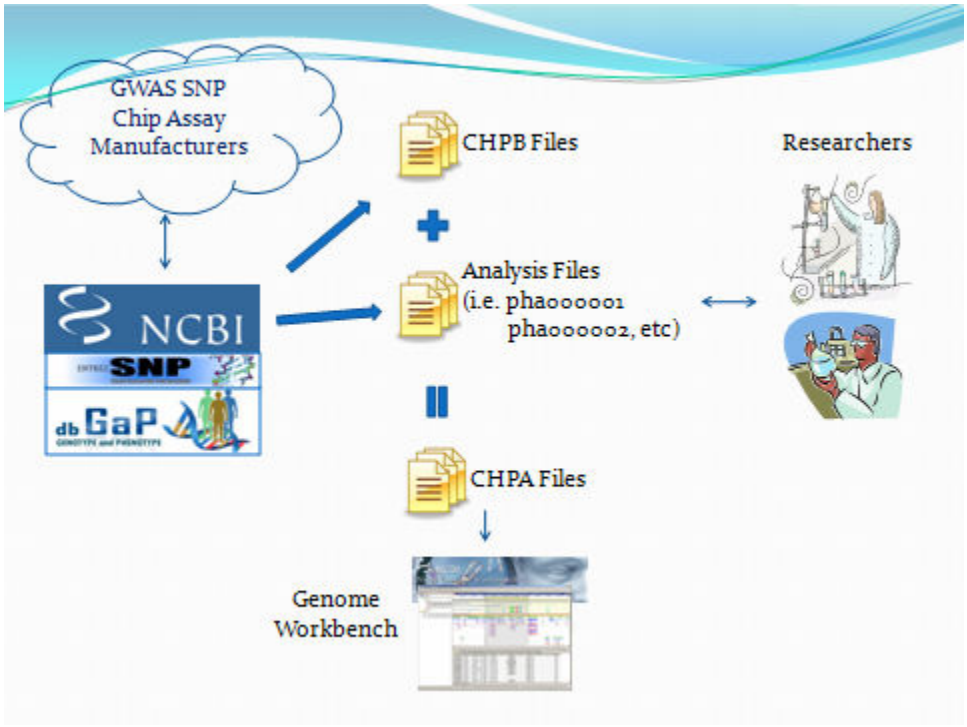
GWAS CHPB files: CHPB files, or chip batch files, contain the SNPs of the GWAS chip for a particular build. These files are generated when dbSNP processes the data submissions of GWAS chips from DNA Chip manufacturers. CHPB files are available online at the dbSNP FTP site, and the rules for the CHPB file format are located in Box 1 of this document.

GWAS Analysis files: dbGAP maintains analysis files of submitted studies, whose file formats vary depending on analysis type. These files are available at the dbGap FTP site organized by submitting institution.

GWAS CHPA files: CHPA files, or chip analysis files, **can be loaded into GBench**. The file format follows the same format as CHPB, except the file extension is .chpa, and an extra column for pvalue is appended at the end. CHPA files are created by taking the CHPB files and the pvalues of an analysis file and merging them using another tool (perl, awk, etc). [see the figure below describing "GWAS Data Flow (File Types)"]

Please Note: As of this date, CHPA files are not published by NCBI and instead the user must separately generate a CHPA file. PERL script examples of how to do generate CHPA files are located in Box 2 of this document. In the future, dbGaP plans to release CHPA files in the same directory as the analysis files.

GWAS DATA Flow (File Types):



(**05/19/08**)

Box 1: CHPB FILE Format Rules.

1. Reference Genome Only
2. Weight 2 and 3 SNPs are included with all positions
3. One file for each chromosome
   a. X and Y get pseudo autosomal region SNPs
   b. Mito is its own file (if they exist on the chip)
   c. Unplaced and NotOn are in a single file
4. Header of file has:
   a. # Submitter Handle
   b. # Batch Id (chip name):
   c. # Genome Build-Assembly:
   d. # dbSNP Build
   e. # Date created
   f. #Version 1 (This will increment on future file format versions)
5. Columns are ordered and are defined below:
   I. ss#
   II. rs#
   III. loc_snp_id
   IV. chrom #
   V. chrom_pos (0-based)
   VI. rsToss_orient
   VII. ssTochrom_orient
   VIII. weight
   IX. bitfield (hexadecimal format, lowest order byte first)
6. File names end in ".chpb"

Box 2: Example PERL Scripts for Merging Analysis File p-values and CHPB Files.

create_chpa.pl

```
#!/usr/bin/perl –w
# Program to combine GWAS chip dumps with analysis files
# Author: Melvin Quintos
# Date:   2008-03-27
# History: -
#

use strict;
use warnings;
use Getopt::Std;

#############################################################################
# Usage
#############################################################################
sub usage()
{
    print STDERR << "EOF";

usage: $0 -o output_file [-r column] [-p column] chpb_file analysis_file

This program takes a GWAS chip batch file and an analysis file and creates a
combined GWAS analysis file (chpa file).

Options:
-h      : this help message
-o      : sets the name of the output file (recommend extension of .chpa)
-r      : rsid column number in analysis file (1-based)
-p      : pvalue column number in analysis file (1-based)

example: $0 -o AREDS_1.chr1.chpa  ILMN_Human-1_chr1.chpb pha000001.v1.p1.chr1.txt

EOF
    exit;
}

#############################################################################
# Global Variables
#############################################################################
my %opts;
my %mapPvals;
my $col_rsid, my $col_pvalue;
my $outputfn;
my $ifn_chpb, my $ifn_analysis;
my @header;

#############################################################################
# Functions
#############################################################################

# Parse command-line options
sub init()
{
    getopts( 'ho:v:r:p:', \%opts ) or usage();
    usage() if $opts{h};
```

Box 2 continues on next page...

Box 2 continued from previous page.

```
}

sub verify_args()
{
    $col_rsid       = ($opts{r}) ? $opts{r} : 1;
    $col_pvalue     = ($opts{p}) ? $opts{p} : 3;
    $outputfn       = $opts{o};

    # validate variables
    if (         (scalar(@ARGV) != 2)            # correct number of args
          or    !(defined($outputfn))            # name of output file is given
          or    !($col_rsid   =~ /^[1-9]\d*$/)    # col is a positive number
          or    !($col_pvalue =~ /^[1-9]\d*$/)    # col is a positive number
       )
    {
        usage();
    }
    else {
        $ifn_chpb       = $ARGV[0];
        $ifn_analysis   = $ARGV[1];
    }
}

# use 1-based column numbering (because it's easier to count that way)
sub get_cols # params: $line, $col_array
{
    my ($line, $col_list) = @_;
    my @result;

    my @items = split(/\t/, $line);
    foreach my $col (@{$col_list})
    {
        push @result, $items[$col-1];  # 1-based columns
    }

    return @result;
}

sub read_chpb_file_version1()
{
    open IFILE, "$ifn_chpb";
    my @cols_of_interest = (2);  # rsid is the 2nd column

    my $reading_header = 1;

    my $line_num = 0;
    foreach my $line (<IFILE>) {
        $line_num++;

        if ($reading_header) {
            if ($line =~ /^ss#/) {
                # Header information stops at row with 'ss#'
                $reading_header = 0;
            }
            else {
```

Box 2 continues on next page...

Box 2 continued from previous page.

```
                push @header, $line;
            }
        }

        # skip all lines that don't have a proper 'ss' number at start of row
        next unless $line =~ /^ss\d+/;
        chomp $line;

        # grab the rsid column
        my ($rsid) = get_cols( $line, \@cols_of_interest );

        if (!defined($rsid)) {
            die "Unknown value in chpb file at line: $line_num\n";
        }

        # trim string, then place in map with a default value of -1
        $rsid =~ s/^\s+|\s+$//g;
        $mapPvals{$rsid} = -1;
    }

    close IFILE;
}

sub read_analysis_file()
{
    open IFILE, "$ifn_analysis";
    my @cols_of_interest = ($col_rsid, $col_pvalue);

    my $reading_header = 1;
    my $line_num = 0;
    foreach my $line (<IFILE>) {
        $line_num++;

        if ($reading_header) {
            if ($line =~ /^# Marker accession/) {
                # Header information stops at row with '# Marker accesssion'
                $reading_header = 0;
            }
            else {
                push @header, $line;
            }
        }

        # skip all lines that are comments or blank lines
        next if $line =~ /^\s*#|^\s*$/;
        chomp $line;

        # for every valid line (e.g. not a comment or blank space)
        my ($rsid, $pval) = get_cols( $line, \@cols_of_interest );

        if (!defined($rsid)) {
            die "Unknown value in file at line: $line_num\n";
        }

        # trim string, then place in map with a default value of -1
```

Box 2 continues on next page...

Box 2 continued from previous page.

```
        $rsid =~ s/^\s+|\s+$//g;
        $mapPvals{$rsid} = $pval;
    }

    close IFILE;
}

# TODO: this is specific to version 1 of chpb file format
sub write_chpa_file()
{
    # open up new file
    open OFILE, ">$outputfn";
    my @cols_of_interest = (2);  # rsid is 2nd column in VERSION=1

    # write header information
    print OFILE @header;
    print OFILE "ss#\trs#\tloc_snp_id\tchrom\tchrom_pos\trsToss_orient
\tssTochrom_orient\tweight\tbitfield\tpvalue\n";

    # read the chpb file again
    open IFILE, "$ifn_chpb";
    foreach my $line (<IFILE>) {
        chomp $line;

        # if the line is a data line, then do a pvalue lookup.
        # otherwise ignore
        if ($line =~ /^ss\d+/) {
            my ($rsid) = get_cols($line, \@cols_of_interest);
            my $pval = $mapPvals{$rsid};
            if (defined($pval) and $pval != -1) {
                $line = "$line\t$pval";
                print OFILE $line, "\n";
            }
        }
    }
    close IFILE;

    close OFILE;

}

##########################################################################
# Program flow
##########################################################################

init();
verify_args();
# TODO: determine which chpb file version we're using
read_chpb_file_version1();
read_analysis_file();
write_chpa_file();

print "Done!\n";

------------------------Example
```

Box 2 continues on next page...

Box 2 continued from previous page.

```
2-------------------------------------------------------------------------
batch_chpa.pl
#!/usr/bin/perl -w
#
# Batch processing of CHPB and analysis files
# Author: Melvin Quintos
# Date:   2008-03-27
# History: -
#

use strict;
use warnings;
use Getopt::Std;
use File::Find;
use Cwd;

#############################################################################
# Usage
#############################################################################
sub usage()
{
    print STDERR << "EOF";

usage: $0 [-d directory] prefix_chpb prefix_analysis

This program goes through and creates chpa data for all analysis and chpb
files in specified directory.

The chpb files must be in the format of <name>chr[1-22,x,y].chpb
The analysis files must be in the format of <name>chr[1-22,x,y].txt

Options:
-h      : this help message
-d      : sets the path to chpb and analysis files (must be in same folder)
          defaults to current directory

example 1: $0 ILMN_Human-1_
example 2: $0 -d path_to_files ILMN_Human-1_

EOF
    exit;
}

#############################################################################
# Global Variables
#############################################################################
my %opts;
my %mapChrChpb;  # map of chromosome to chpb file
my $workdir;
my $initdir;
my $chpb_prefix;

#############################################################################
# Functions
#############################################################################
```

Box 2 continues on next page...

Box 2 continued from previous page.

```
# Parse command-line options
sub init()
{
    getopts( 'hd:', \%opts ) or usage();
    usage() if $opts{h};
}

# setup initial program conditions
sub verify_args()
{
    $workdir = ($opts{d}) ? $opts{d} : '.';
    $initdir = cwd;

    if (scalar(@ARGV)!=1) {
        usage();
    }
    else {
        $chpb_prefix = $ARGV[0];
    }
}

# given an array of files, return only an array of analysis files
sub filter_analysis
{
    my @files;
    foreach my $file (@_) {
        push @files, $file if $file =~ /\.txt$/i;
    }
    @files;
}

# given an array of files, return only an array of chpb files
sub filter_chpb
{
    my @files;
    foreach my $file (@_) {
        # check extension
        if ($file =~ /\.chpb$/i) {
            # check prefix of filename)
            if ( substr($file, 0, length($chpb_prefix)) eq $chpb_prefix) {
                push @files, $file;
            }
        }
    }
    @files;
}

# Store the chpb file by the chromosome it represents
sub process_chpb
{
    my $fn = $_;
    if (-f $fn) {                       # if a file
        if ($fn =~ /(chr.+)\.chpb$/) {  # extract the chromsome name from file
            $mapChrChpb{$1} = $fn;      # store in map
```

Box 2 continues on next page...

Box 2 continued from previous page.

```
        }
    }
}

# Process each analysis file against the proper chromosome
sub process_analysis
{
    my $fn = $_;
    if (-f $fn) {                   # if a file
        if ($fn =~ /(chr.+)\./) {  # extract chomosome name
            # if found in map, then process the files
            my $chpb_file = $mapChrChpb{$1};
            if (defined($chpb_file)) {
                my $new_file = $fn;
                $new_file =~ s/\.txt$/\.chpa/i;
                print "Processing $new_file ... ";
                my $exeCmd = qq{"$initdir/create_chpa.pl" -o $new_file
$chpb_file $fn};
                system($exeCmd);
            }
        }
    }
}

########################################################################
# Program flow
########################################################################

init();
verify_args();

# Process the names of the CHPB files
find({wanted=>\&process_chpb, preprocess=>\&filter_chpb}, $workdir);

# Process the names of analysis files
find({wanted=>\&process_analysis, preprocess=>\&filter_analysis}, $workdir);

print "Done!\n";
```

Loading Private GWAS Data to GST

How do I load private GWAS data (GWAS data my lab has generated) to GBench so I can study it using GST?

1 Create a CHPA file by merging the p-values from your laboratory's analysis files with the corresponding CHPB file(s):
   a. Access your laboratory's analysis files
   b. Download the corresponding CHPB files from the GWAS array folder located in the human _9606 directory of the dbSNP FTP site.
   c. Use an external program, such as PERL, to merge the p-values of your analysis files with the CHPB files to create CHPA files. Click here to see an example PERL scripts for this purpose.

Please Note: the CHPA files you generate will be organized by chromosome

2. Modify the "Name" attribute of your new CHPA file

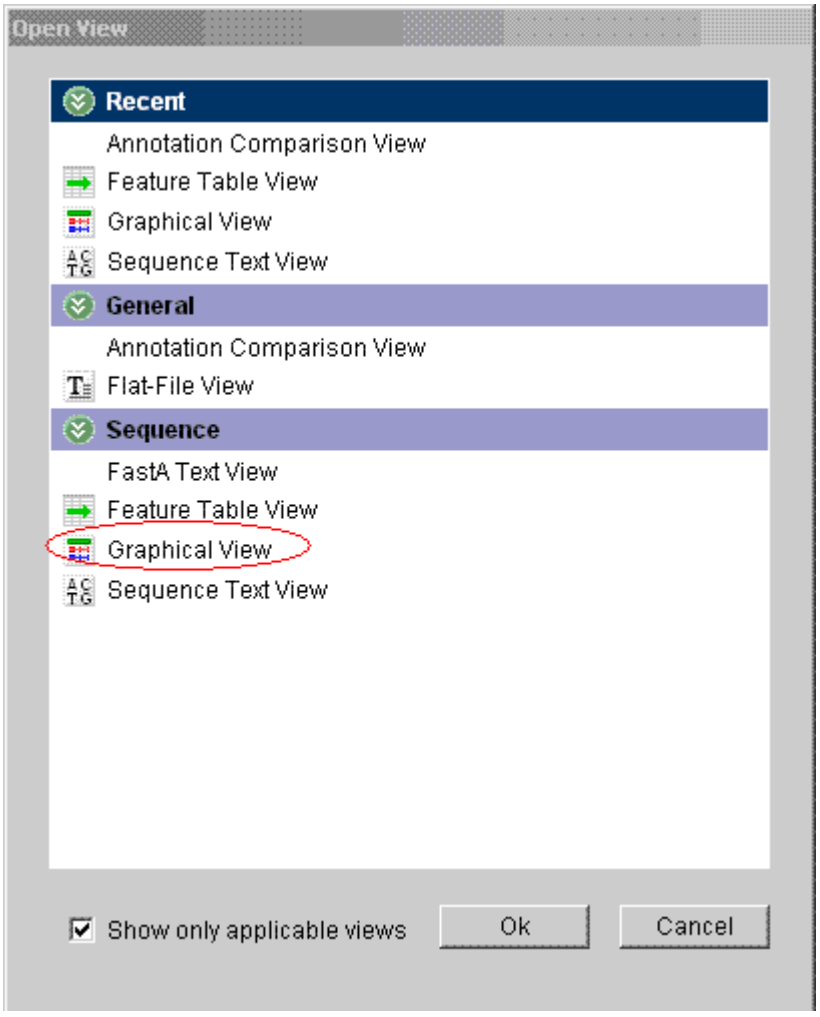
You will need to create a name for your CHPA file. This name will appear above the data track in the graphical view that represents the data in the CHPA file. Make sure this name is meaningful to you so that you can discern the difference between the graphical track created by this CHPA file, and other tracks you may have open with it in the Graphical View.

To modify the "name" field in your CHPA file, all you have to do is open up the CHPA file in a text editor (like notepad or wordpad) and add the comment line "# Name:" followed by your "meaningful" name.

3: Create a View of your Loaded Data

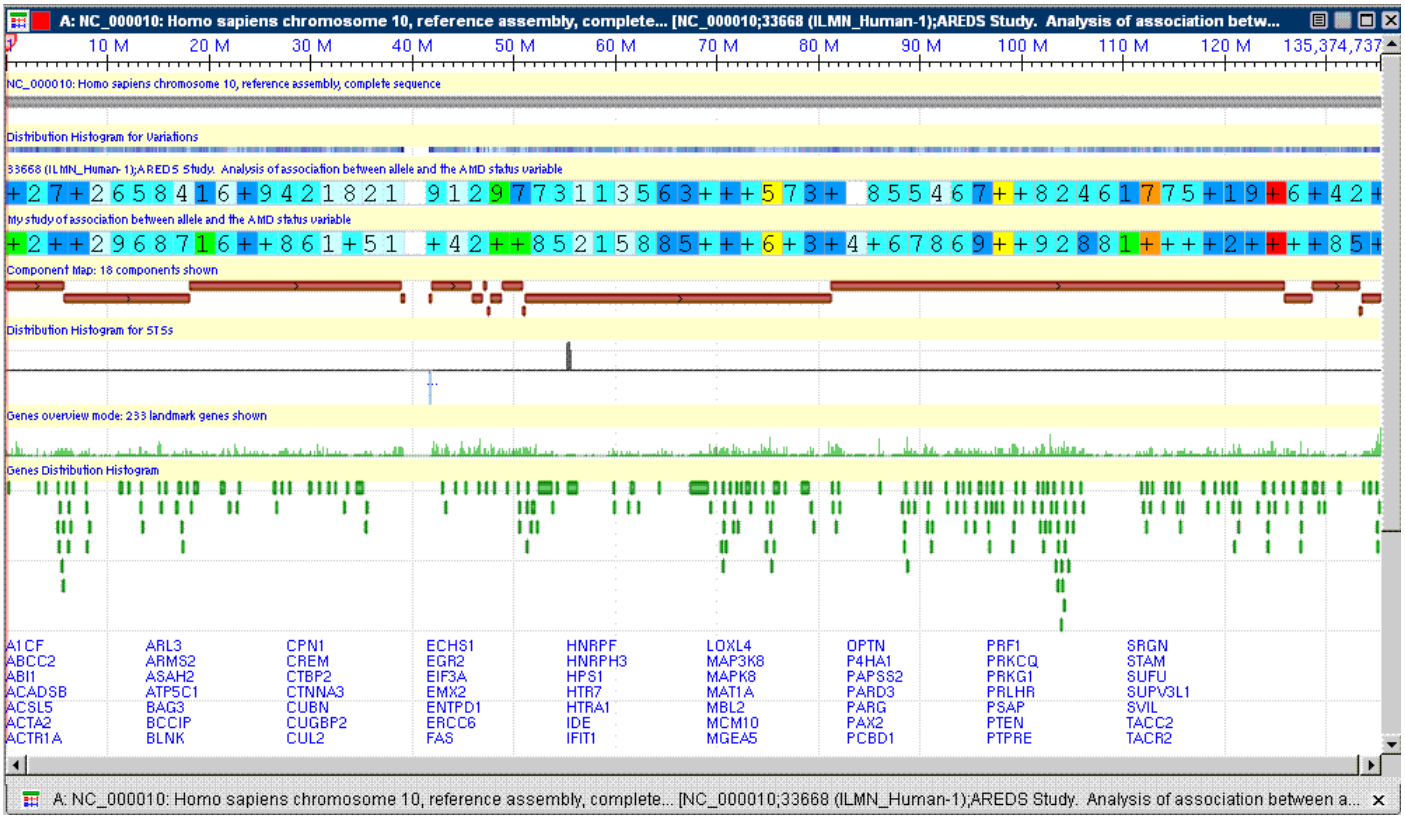
Once loading is complete, your new project will appear in the GBench Project Tree (located in the upper left of the GBench pane) that contains the GWAS data. Each study will generate two nodes in the data tree: one node has a green arrow icon; the other node has a blue mountain icon: There is no discernable difference between the data contained within these nodes.

a. Double click on the green arrow icon for the data for your study.
b. In the "Open View" dialogue box that appears, choose "Graphical View":



c. In the "Graphical View" dialogue box that appears, select the data of your choice and click "OK": The load operation may take a moment to finish.

d. The view that is generated should look similar to the following if you have both a track of your private data and a track of public data displayed:


A: NC_000010: Homo sapiens chromosome 10, reference assembly, complete... [NC_000010;33668 (ILMN_Human-1);AREDS Study. Analysis of association betw...
10 M
20 M
30 M
40 M
50 M
60 M
70 M
80 M
90 M
100 M
110 M
120 M
135,374,737
NC_000010: Homo sapiens chromosome 10, reference assembly, complete sequence
Distribution Histogram for Variations
33668 (ILMN_Human-1);AREDS Study. Analysis of association between allele and the AMD status variable
My study of association between allele and the AMD status variable
Component Map: 18 components shown
Distribution Histogram for STSs
Genes overview mode: 233 landmark genes shown
Genes Distribution Histogram
A1CF
ABCC2
ABI1
ACADSB
ACSL5
ACTA2
ACTR1A
ARL3
ARMS2
ASAH2
ATP5C1
BAG3
BCCIP
BLNK
CPN1
CREM
CTBP2
CTNNA3
CUBN
CUGBP2
CUL2
ECHS1
EGR2
EIF3A
EMX2
ENTPD1
ERCC6
FAS
HNRPF
HNRPH3
HPS1
HTR7
HTRA1
IDE
IFIT1
LOXL4
MAP3K8
MAPK8
MAT1A
MBL2
MCM10
MGEA5
OPTN
P4HA1
PAPSS2
PARD3
PARG
PAX2
PCBD1
PRF1
PRKCQ
PRKG1
PRLHR
PSAP
PTEN
PTPRE
SRGN
STAM
SUFU
SUPV3L1
SVIL
TACC2
TACR2
A: NC_000010: Homo sapiens chromosome 10, reference assembly, complete... [NC_000010;33668 (ILMN_Human-1);AREDS Study. Analysis of association between a...

(**05/22/08**)