

Genomes, Browsers and Databases

Peter Schattner

Department of Biomolecular Engineering

University of California

Santa Cruz, CA

Outline

Why is this tutorial important?

The number of molecular biology databases continues to explode. Presently, few problems in any area of genomic molecular biology can be addressed without analyzing data stored in these databases. However, these databases are located in many different locations and often use non-standard data formats requiring specialized data parsers. As a result integrating and comparing data from multiple biological databases is difficult and tedious.

The genome databases at UCSC, Ensembl and NCBI offer solutions to this problem by integrating data from multiple databases in a uniform and standardized manner. However, effectively using these databases also has a considerable learning curve, especially if one wants to query multiple genomic regions in an automated manner rather than simply analyzing individual genes via an interactive browser. This tutorial is intended to help students and researchers climb this learning curve more expeditiously.

The tutorial should be useful for both biology and bioinformatics students and researchers. Biologists will learn how to extract a wider range of relevant annotations for their genes of interest from the browsers. Bioinformaticians will learn how to access the underlying browser databases to perform automated, large scale queries across entire genomes. As important, both groups will gain an appreciation for the methods by which the browsers and their databases are constructed so that they are prepared to take advantage of new features and enhancements that are continually being incorporated into these important tools.

What does the tutorial cover?

The tutorial discusses interactive, batch and automated querying of the three major genome databases (UCSC, Ensembl and NCBI). For pedagogical and time constraint reasons, most of the focus is on a single database, the UCSC genome browser database.

The tutorial begins with a background section that provides motivation why browsers have become essential tools for biologists and bioinformaticians. This section also introduces the three major browsers, compares some of their specific strengths and limitations and explains my motivation for focusing primarily on the UCSC Browser.

The second section describes using the browser in interactive web-based mode, which is the most common, and easiest, method of accessing these resources. Interactive mode for the UCSC Browser is described, including many of its features and controls. This is done in the context of a simple biological example.

Part Three includes a more “behind the scenes” view of some of the tools used to build the browser, the understanding of which is necessary to appreciate the resources browsers offer, as well as their limitations. Topics include genome assemblies and database builds and local and whole-genome sequence alignment methodologies.

Part Four introduces batch querying of browser databases and why this can be an important tool. Using the UCSC Browser as an example, batch querying using SQL, the Table Browser and Galaxy are all introduced. Details of file and table formats are discussed as well as potential pitfalls that can fool the novice. The previously used biological query example is extended to a situation requiring batch querying. Batch querying in Ensembl is also briefly described.

Finally in Part Five, automated program-based querying of the browser database is described. Topics include when program-based querying is advantageous, tradeoffs between remote login and mirror site development and installation issues. In the last section a simple, but complete, working C program for automated querying is presented.

CONTENTS

PART 1: INTRODUCTION TO GENOME BROWSERS	5
PART 2: INTERACTIVE QUERYING ON THE UCSC BROWSER.....	6
PART 3: ASSEMBLIES, ALIGNMENTS ETC. - A GLIMPSE INSIDE.....	6
PART 4: BROWSER/DATABASE BATCH QUERYING.....	8
PART 5: AUTOMATED QUERYING PROCEDURES.....	12
SAMPLE DATABASE QUERYING PROGRAM.....	15
FINDING MORE INFORMATION.....	19
REFERENCES.....	20

INTRODUCTION TO GENOME BROWSERS

Integrated genome browsers have become essential tools for the analysis of genomic data. They offer the ability to visualize disparate annotations of genes and other genomic locations from single or multiple species in ways that are simply not possible with previous tools. These tools have become especially important recently because the number of individual biological databases is growing rapidly. Moreover, many of these “databases” are downloadable as flat-files only, meaning that searching them is slow or else local relational databases need to be set up. Also, differing data formats are used, requiring the use of multiple data parsers. Finally complex queries require integrating data from multiple databases.

As an example, say you found a synonymous codon polymorphism in a possible disease gene and wanted to check its properties such as: is it in dbSNP? Does it occur in any known EST? Is the site conserved in other vertebrates? It is possible to answer these questions without an integrated browser but it would require finding and using multiple different resources (dbSnp, Genbank, BLAST, etc) each with its own idiosyncrasies and learning curves. With one of the browsers such queries can be accomplished in a simple and straightforward manner.

There are three main browsers: Ensembl, NCBI MapViewer and UCSC. They have more similarities than differences. Choosing one or the other is pretty subjective. But it is probably most useful to pick one you like and stick mainly with that one, since they each have their own learning curve. For this tutorial I focus primarily on the UCSC Browser because it has several strengths:

- Strong comparative genomics capabilities
- Fast –especially when doing sequence searches with BLAT
- (Essentially) a single “view” from single base-pair to entire chromosome
- UCSC system is well suited for batch and automated querying.
- The UCSC Browser is comprehensive, especially for non-protein coding regions
- Frequent annotation updates (Genbank/Refseq daily / ESTs weekly).

And, in truth, probably the main reason for my choosing the UCSC Browser is that it is the one with which I have by far the most experience.

INTERACTIVE QUERYING ON THE UCSC BROWSER

Interactive Querying on the UCSC Browser is straightforward and can be described in four simple steps. First you need to go to: <http://genome.ucsc.edu>. Next choose a genome and an assembly. Then pick a genomic region of interest. Finally specify what annotations you want

Browser annotations are via “tracks” along the chromosome. Scores of annotation tracks can be selected. To help the user navigate among them, they are divided into “Groups”. Examples of track types include chromosome descriptions, gene annotations, local paired alignments (mRNAs, ESTs), comparative genomics annotations, annotations of species variations. New tracks are being added all the time. Some recent tracks include ENCODE annotations, Retroposed genes, snoRNAs / miRNAs, RNA fold predictions, segmental duplications, and Affymetrix full chromosome transcriptional data. For a more detailed recent update see Hinrichs et al in the References.

In addition to the track data there is a wealth of other data available in the UCSC browser, which can be found in “Details” pages, the GeneSorter, the Proteome Browser, VisiGene, BLAT and the isPCR tool. Examples of the use of these tools and resources are given in the tutorial slides. For more detailed examples of interactive browser use with the UCSC Browser see the Openhelix website (www.openhelix.com).

ASSEMBLIES, ALIGNMENTS AND ALL THAT

Two fundamental issues that are important to understanding the data presented at the UCSC Browser are how the database is “built” and how the alignments of sequence tracks of differing sequence data from other species are carried out.

In contrast to some systems that describe genomic locations in terms of physical “contig” locations, the UCSC browser uses actual chromosome coordinates. As a result, each new assembly of a genome will change the positions of most features (i.e. annotations). Consequently after a genome reassembly the entire database needs to be rebuilt. A browser utility exists to convert coordinates between assemblies.

One source of confusion is that new builds may not have all the tracks of a previous build. This is partly because the “build” process is not completely automated and because tracks that are no longer considered important may not be rebuilt in new assemblies. Moreover many annotations are updated between builds (in some cases nightly). This can be confusing when comparing with results obtained previously from the same build.

Much of the power of the UCSC Browser comes from its comparative genomics tools, which in turn are based on its powerful alignment tools. For speed purposes, all alignments (except for those resulting from user-initiated BLAT queries) are precomputed and stored. However, powerful as it is, the UCSC alignment-tool suite has its limitations and those limitations will be reflected in the alignments presented in the browser, and therefore need to be understood.

For local paired alignments, UCSC uses BLAT, translated BLAT and BLASTZ. BLAT and translated BLAT tools are extremely fast but lower sensitivity than BLAST especially for cross-species (i.e. xeno) alignments. BLASTZ is similar to BLASTn with an optimised scoring algorithm for cross species comparisons. Local multiple alignments are performed with MULTIZ (an extension of BLASTZ) and cross species conservation of multiple alignments is scored with phastCons, a phylogenetic Hidden Markov Model tool.

For paired genomic alignments, UCSC uses “chains and nets”. This approach has the advantages of being not dependent on high quality gene annotations and having support for segmental duplications and inversions. Nets and chains are described in more detail in the paper by Kent (2003) in the References. Genomic multiple alignments are generated by the “threaded blockset aligner” (TBA) tool. TBA starts with local multiple alignment “seeds” generated by MULTIZ which are then linked together (“threaded”) to form longer alignments. TBA currently cannot handle segmental duplications and inversions. For more detail on TBA see the paper by Blanchette in the references.

BROWSER/DATABASE BATCH QUERYING WITH THE UCSC BROWSER

Interactive querying is difficult if you want to study numerous “interesting” genomic regions. Querying each region interactively is tedious, time-consuming and error prone. For example, we can extend the example used in Part II and suppose you have found *hundreds* of candidate polymorphisms and you want to know which of them are in dbSNP or overlap known ESTs or are at sites conserved in other vertebrates.

To answer such questions efficiently requires batch querying of the underlying genome database(s). In the UCSC Browser, data for each assembly are typically stored in a separate database and auxiliary data, e.g. gene ontology (GO) data, are stored in yet other databases. These databases may have hundreds of tables, many with millions of entries. The conventional way of querying a relational database is via “Structured Query Language” (SQL). However with tools such as the Table Browser, you can query the database without using SQL.

Nevertheless, even with the Table Browser, you need some understanding of the underlying track, table and file formats. Table formats describe how data is stored in the (relational) databases. Track formats describe how the data is presented on the browser. File formats describe how the data is stored in “flat files” in conventional computer files. Finally, for understanding the underlying the computer code you will need to learn about the “C” structures which hold the data in the source code.

These classes of formats can be confusing because they are typically similar to one another but may differ in subtle ways. However, at least in the case of the UCSC Browser, the situation is simplified because there are utility programs available for converting files to tracks to tables and to C- structures and back again. In some cases the files are actually converted automatically between table (SQL) format and code (C) format by a dedicated program called autoSQL. autoSQL is described in detail at <http://www.linuxjournal.com/article/5949>.

The principle file and table formats are: BED (“Browser extensible data”) – for gene and chromosome annotations, PSL (“Pattern space layout”) – for pair-wise alignments, MAF (“Multiple alignment format”) – for multiple alignments and WIG (“Wiggle format”) - for numerical data. Although the design of these structures is

quite logical, they do have some subtleties (especially for representing negative strand data) that can create pitfalls for those not used to them. More detailed descriptions of database tables can be found at:

<http://genome.ucsc.edu/goldenPath/gdbDescriptions.html>

In most cases, using SQL is not necessary to obtain the information you need from the Browser Database. The Table Browser can accomplish this task for you. Retrieving specific subsets of data can be accomplished with the table filtering and intersection tools. Moreover the data can be retrieved in a variety of useful output formats. A particularly useful output format is as a custom track, which then can be displayed on the browser or intersected with other tables. Direct SQL querying of the UCSC databases is also supported. This can be accomplished either via the Table Browser itself, through the public UCSC mirror database at genome-mysql.cse.ucsc.edu or by setting up one's own mirror database.

GALAXY

The Galaxy Website (<http://g2.bx.psu.edu>) was developed for several reasons. Ultimately its intent is to provide an easy interface to sequence and data manipulation tools (a la SRS or the UCSD Biology Workbench) that are capable of being applied to genomic data. It offers varied output formats and is intended to work with data from multiple browsers / databases.

First released in 2005, Galaxy is still somewhat of a “work in progress”. To date, it supports the UCSC Table Browser, EBI EnSmart and NHGRI databases and offers only a few sequence manipulation tools (e.g. GC%, Ka/ Ks calculations). However, Galaxy does already offer an interface to the UCSC Table Browser that is arguably more “user friendly” than UCSC’s, especially in cases where table intersection, union or similar manipulations need to be performed.

Galaxy is likely to add more useful tools in the near future and is probably worth monitoring as an alternative entry point to the browser databases for batch querying. More details on Galaxy can be found in the paper by Giardine et al in the references.

BROWSER “GOTCHAS”

Although the UCSC Databases are generally configured in a very straightforward and logical manner there are situations where the system interface works in an unexpected manner to those inexperienced with it. Some of these situations reflect intrinsic difficulties or ambiguities in describing genomic sequence data. Others are specific to the conventions used in the UCSC system.

General pitfalls in interpreting stored genomic data include the fact that a gap in an alignment of an mRNA to the genome may not be intron, but rather an insertion in the genome sequence or a deletion in the mRNA. Also sequence differences between an mRNA and the genome may not represent polymorphisms, but rather sequencing artifacts. Another potential source of confusion is that the number of blocks in an alignment between an mRNA and the genome may not be the same as the number of “blocks” (i.e. exons) in the gene that is predicted to be represented by that mRNA.

Subtleties to watch out for that are relatively specific to the UCSC data representation include the fact that UCSC uses a “half open” numbering system and that the data is stored internally starting at position “0” while it is displayed as if it started at position “1”. Issues of speed cause the database to use various indexing fields, including “bin” fields and MAF index fields for searching in external files that can cause confusion if one is not expecting to see them. Also, block (e.g. exon) and strand data is stored in different ways in different kinds of tables. Although these varying kinds of representations all make sense once they are brought to one’s attention, they can be quite puzzling if one is not prepared for them.

To find out more information about these aspects of the UCSC data representation as well as to find several other examples of possibly unexpected behavior within the browser database, a good place to look are the “Frequently Asked Questions” section of the browser documentation located at: <http://www.genome.ucsc/FAQ/>

ENSEMBL’S APPROACH TO BATCH QUERIES AND CUSTOM TRACKS

Performing batch queries and creating custom tracks is also possible in Ensembl. Specifically Ensembl’s version of the UCSC Table Browser is called “BioMart”. It is

more “gene oriented” than the Table Browser with somewhat different features. In particular, BioMart offers a tight interface with the R/Bioconductor project for the analysis of microarray data.

The Ensembl track display system uses “DAS” (Distributed Annotation System) a widely used system for using multiple remotely located servers for displaying genomic annotation information. Local DAS client software integrates the results from the various servers. DAS is intended to be more scalable, since maintaining tracks is not responsibility of single group. To find out what annotations are available from the Ensembl DAS system, the user can check the DAS registry at: <http://das.sanger.ac.uk/registry>

AUTOMATED QUERYING PROCEDURES

Using the Table Browser or Galaxy or BioMart is still a partly interactive process. Consequently, when performing multiple, large scale queries this approach becomes time-consuming and error prone. Moreover, complex data analyses often require performing data manipulation in software, anyway, and it may be more efficient to integrate this analysis with the data retrieval. In such cases, it is often desirable to be able to perform database querying in a fully automated manner.

A typical case arises by modifying our polymorphism-analysis example. Specifically, assume that instead of experimental data, we have a computer algorithm to *predict* candidate disease polymorphisms and we want to know whether the predicted polymorphisms are in dbSNP or in known ESTs or at vertebrate conserved sites. Furthermore, let's assume that our computer algorithm has several adjustable parameters and each time we change them we would get a new list of putative biological polymorphisms. We will NOT want to interactively perform all the Table Browser table-intersections every time we modify a parameter. Although this specific example is made-up, it is not unlike more realistic ones, such as characterizing the introns of genes that host snoRNAs (see Schattner et. al in the References) or characterizing regions of extreme codon conservation (see Schattner and Diekhans)

Although fully automated database querying is very powerful, it does require certain prerequisites not needed by the interactive and batch-querying methods described so far. Specifically you'll need: general programming skills, database programming skills, and direct (SQL) access to a browser database.

In principle automated database querying is feasible with any of the three major browser databases. The choice of which to use is likely to be largely motivated by which computer and database languages you are most comfortable with. Although for commercial users, licensing considerations may also be of importance.

In this tutorial, I discuss automated querying of the UCSC database, which uses the C language and the MySQL database architecture. My reasons for focusing on the UCSC database are (besides that it is the one with which I am most familiar) the availability of an extraordinary code base on which to build writing ones own

programs, the comprehensive nature of the database and the fact that “C” code is fast.

Without question, for me the biggest advantage of using the UCSC database is the availability of the code in the kent source tree. The code is clearly written, extensively tested and fast. It is also open source so you can learn from it and modify it for your own use and free for academic, government and personal use (core routines are even free for commercial use).

With the kent source code, many important utilities are usable “right out of the box”. Built-in library functions provide almost any sequence and data manipulation capability you might want. These library functions are located primarily in the “lib” and “hg/lib” subdirectories of kent/src/. Plenty of code examples illustrate exactly how to use the library effectively. There are CGI-based programs to perform all the sequence and data manipulations performed by the browser. In general, if the browser performs some data manipulation, with a little detective work, you can find the code to insert in your program. Sometimes the appropriate program can be identified by simply looking after the “cgi-bin” in the web address as in genome.ucsc.edu/cgi-bin/hgTracks. Moreover, many browser cgi-programs can be run in stand-alone mode. You just need to give it the proper arguments, which are stored in the “CART”. Current CART arguments can be examined by running: <http://genome.cse.ucsc.edu/cgi-bin/cartDump>.

The browser code is largely object oriented. A “C” structure is defined for each type of track and table. Associated functions (i.e. methods) are defined to implement the various manipulations that can be performed on the data in the structure. In addition, functions are available for loading and writing data to and from the database.

However, before you can run automated queries on the UCSC database you need a database-copy on which to run them. This can be either the public UCSC database at genome-mysql.cse.ucsc.edu or one’s own mirror database. The advantages of using the public database are that it does not require up to 1.2 Tbyte or more of disk space and there is no database installation or maintenance required. (You will need to modify the .hg.conf files and the library routine sqlConn.c because the public database does not use a password whereas the routines in the library do.) In contrast, having your own mirror has the advantages of not being a shared resource. Consequently, you can run it as heavily as you like. Performance doesn’t

depend on usage by others. And the database can be modified / customized to meet your specific needs.

Detailed instructions for setting up a mirror of the UCSC database and browser can be found at: <http://genome.ucsc.edu/admin/mirror.html>. If you have more than 1.2 TBytes of free space then simply following the instructions there should work fine. However, if you have less available space, it works perfectly well to only install the databases for the species you need to work on. In fact, you may well choose to only install a subset of the tables and files of a database. Also you do not need to install Apache or any CGI or HTML files if you are only doing database querying.

In any case you will need to download and compile the kent source code. It's a good idea to compile the kent code with the "debug" option enabled (not the default). This is not because the kent code is buggy (it is not) but rather because it makes it easier to track problems in your own code later using an interactive debugger such as gdb.

Once you are configured for remote access to the public database or have installed a local mirror of the databases you will be querying, writing programs to actually do the database querying is straightforward. In the next section is a "toy" program, demonstrating both approaches. The program is pretty simple, but it illustrates the main ideas, is complete and does work (tested under Mac OS X and linux). The source and input data files can be found at: <http://www.soe.ucsc.edu/~schattner/> (Or if you want to try them by cutting and pasting them, be careful of "line breaks" that might have been introduced during reformatting).

```

/* gbdExample - illustrates accessing data from public UCSC
 * database or a locally installed mirror or downloaded file */
#include "common.h"
#include "options.h"
#include "jksql.h"
#include "bed.h"
#include "binRange.h"
#include "genePred.h"
#include "genePredReader.h"
#include "hdb.h"

/*****Globals*****/
struct slDouble *overlapList = NULL;
struct slDouble *otherList = NULL;

/*****Globals*****/
void usage()
/* Explain usage and exit. */
{
errAbort(
    "gbdExample - find median length of introns overlapping ranges in
input file\n"
    "usage:\n"
    "    gbdExample db dbTable myBedFile method\n"
    "        where db is the database name \n"
    "        where dbTable is tableFileName in 'file' mode or else\n"
    "            name of table to use in 'public' or 'localDb' modes
\n"
    "        where myBedFile is a bed file of genomic ranges \n"
    "        where method is either 'public' or 'localDb' or 'file' \n"
    "\n");
}

/*****Globals*****/
void binKeeperGpHashFree(struct hash **hash)
/* adapted from binKeeperPslHashFree in pslPseudo.c */
{
if (*hash != NULL)
{
    struct hashEl *hashEl = NULL;
    struct hashCookie cookie = hashFirst(*hash);
    while ((hashEl = hashNext(&cookie)) != NULL)
    {
        struct binKeeper *bk = hashEl->val;
        struct binElement *elist = NULL, *el = NULL;;
        elist = binKeeperFindAll(bk) ;
    }
}
}

```

```

        for (el = elist; el != NULL ; el = el->next)
        {
            struct genePred *gp = el->val;
            genePredFree(&gp);
        }
        binKeeperFree(&bk);
    }
    hashFree(hash);
}

/*****/
struct hash *readGpToBinKeeper(char *gpFileName)
/* adapted from readPslToBinKeeper in psl.c */
{
#define MAX_CHROM_SIZE 400000000
struct binKeeper *bk;
struct genePred *gp;
struct lineFile *pf = lineFileOpen(gpFileName , TRUE);
struct hash *hash = newHash(0);
char *row[21] ;
int genePredLineCtMin = 10;
while (lineFileNextRow(pf, row, genePredLineCtMin))
{
    gp = genePredLoad(row);
    if (hashLookup(hash, gp->chrom) == NULL)
    {
        bk = binKeeperNew(0, MAX_CHROM_SIZE);
        hashAdd(hash, gp->chrom, bk);
    }
    bk = hashMustFindVal(hash, gp->chrom);
    binKeeperAdd(bk, gp->txStart, gp->txEnd, gp);
}
lineFileClose(&pf);
return hash;
}

/*****/
struct genePred *bkToGenePreds(struct hash *gpHash, char *chrom, int
start, int end)
/* */
{
    struct genePred *gpList = NULL;
    struct genePred *gp;
    struct binKeeper *bk = hashFindVal(gpHash, chrom);
    struct binElement *el, *elist = binKeeperFind(bk, start, end) ;

```



```

for (el = elist; el != NULL ; el = el->next)
{
    gp = el->val;
    if (gp != NULL)
    {
        slSafeAddHead(&gpList, gp);
    }
}
slFreeList(&elist);
return gpList;
}

/*****/
struct sqlConnection *getHgdbtestConn(char *db)
/* Read .hg.conf and return connection. */
{
    char *host = "genome-mysql.cse.ucsc.edu";
    char *user = "genome";
    char *password = NULL;
    hSetDbConnect(host,db,user,password);
    return sqlConnectRemote(host, user,password, db);
}

/*****/
int genePredLongestCmp(const void *va, const void *vb)
/* Compare to sort based sizes of txEnd - txStart, largest first. */
{
    const struct genePred *a = ((struct genePred **)va);
    const struct genePred *b = ((struct genePred **)vb);
    int lengthA = a->txEnd - a->txStart;
    int lengthB = b->txEnd - b->txStart;
    int dif = lengthB - lengthA;
    return dif;
}

/*****/
void doOneGene(struct genePred *gp, int qStart, int qEnd)
/* get intron statistics for longest gene in range */
{
    int i, intronStart, intronEnd;
    for (i=1; i< gp->exonCount; ++i)
    {
        intronStart = gp->exonEnds[i - 1];
        intronEnd = gp->exonStarts[i];
        double intronLength = (double) (intronEnd - intronStart);
        struct slDouble *slIntronLength = slDoubleNew(intronLength);
        if (positiveRangeIntersection(qStart, qEnd, intronStart,
intronEnd))

```

```

        {
            slSafeAddHead(&overlapList, slIntronLength);
        }
    else
    {
        slSafeAddHead(&otherList, slIntronLength);
    }
}

}

/*****/
void doOneBed(struct bed *bed, struct sqlConnection *conn,
             char *geneTable, struct hash *gpHash)
/* get intron statistics for longest gene in range */
{
    int bStart = bed->chromStart;
    int bEnd = bed->chromEnd;
    struct genePred *gp = NULL;
    if (gpHash == NULL)
        gp = genePredReaderLoadRangeQuery(conn, geneTable, bed->chrom,
        bStart, bEnd, NULL);
    else
        gp = bkToGenePreds(gpHash, bed->chrom, bStart, bEnd);
    slSort(&gp, genePredLongestCmp);
    if (gp == NULL)
    {
        errAbort("No gene found in %s overlapping %s:%d-%d\n",
            geneTable, bed->chrom, bStart, bEnd);
    }
    doOneGene(gp, bStart, bEnd);
    if (gpHash == NULL)
        genePredFreeList(&gp);
}

/*****/
void processBedFile(char *bedFile, struct sqlConnection *conn,
                  char *geneTable, struct hash *gpHash)
/* Read file and process */
{
    struct bed *bedList=NULL, *bed=NULL;
    bedList = bedLoadAll(bedFile);
    for(bed = bedList; bed != NULL; bed = bed->next)
    {
        doOneBed(bed, conn, geneTable, gpHash);
    }
}

```

```

printf("Median value of lengths of overlapping introns = %f\n",
slDoubleMedian(overlapList));
printf("Median value of lengths of other introns = %f\n",
slDoubleMedian(otherList));
bedFreeList(&bedList);
}

/*****/
/* gbdExample.c */
int main(int argc, char *argv[])
/* Find median value of lengths of introns overlapping ranges in
input file
* and compare with lengths of other introns in those genes
* Program reads 'bed file' of genomic regions and
* extracts longest gene overlapping each region. For each
* gene, lengths of introns overlapping the region as well
* as those not overlapping the region are computed. Medians
* of each set of intron lengths is printed out.
* Program is compiled with:
gcc -g -Wall -Werror -I${KENTSRC}/inc -I${KENTSRC}/hg/inc -o
gbdExample gbdExample.c ${KENTSRC}/lib/${MACHTYPE}/jkhgap.a
${KENTSRC}/lib/${MACHTYPE}/jkweb.a ${MYSQLLIBS} -lm
* where $KENTSRC is the local location of the kent source
* tree, $MYSQLLIBS is location of the local MySQL libraries
* and $MACHTYPE is the machine type environmental variable
* Once compiled and linked, the program is run as e.g.:
./gbdExample sacCer1 sgdGene myYeastBedFile localDb
* or
./gbdExample hg17 refGene myHumanBedFile public
* or
./gbdExample sacCer1 sgdGene.txt myYeastBedFile file
* where the first argument is the db to use (this parameter is
* ignored in 'file' mode, the second argument is the name of
* the db or file gene table, third program argument is the location
* file of (bed) locations to be screened for intron lengths,
* and the fourth argument indicates whether to use
* the public UCSC database at genome-mysql.cse.ucsc.edu
* or a locally installed mirror or a downloaded file.
*/
{
char *db = argv[1];
char *geneTable = argv[2];
char *bedFile = argv[3];
char *method = argv[4];
if (argc != 5)
usage();

```

```

struct sqlConnection *conn = NULL;
struct hash *gpHash = NULL;
if (sameWord(method, "file"))
    gpHash = readGpToBinKeeper(geneTable);
else
{
    conn = sameWord(method, "public") ?
        getHgdbtestConn(db) : sqlConnect(db);
}
processBedFile(bedFile, conn, geneTable, gpHash);
slFreeList(&overlapList);
slFreeList(&otherList);
sqlDisconnect(&conn);
binKeeperGpHashFree(&gpHash);
return 0;
}

```

FINDING MORE INFORMATION

There are numerous excellent sources of additional material describing the three browsers and their underlying databases. Unfortunately much of this material is distributed in numerous articles and web pages that are located in different places. A good place to start is Openhelix at www.openhelix.com. They have excellent free on-line tutorials and powerpoint presentations that cover much of the material from Part II of the present tutorial (i.e. on interactive browsing with the UCSC browser) in more detail than I do.

Additional tutorials, userguides and FAQs can be found at the respective browser websites:

genome.cse.ucsc.edu/goldenPath/help/hgTracksHelp.html

www.ensembl.org/Docs

www.ncbi.nlm.nih.gov/mapview/static/MapViewHelp.html

To understand the methods underlying the design and development of the browsers themselves, one needs to go to the original research literature and subsequent reviews. Recent articles that I have found most helpful in understanding the workings of the browsers are listed in the References section.

To *really* understand the detailed workings of the browsers, it is often necessary to go to the code itself. This code is typically open source and relatively well documented. For the UCSC Browser, all the code is in the kent “source tree” which can be freely downloaded at <http://www.soe.ucsc.edu/~kent/src/>. If you want to see in detail how a large database, like hg17, is built, look at `makeHg17.doc` in `kent/src/hg/makeDb/`.

Finally, you may want to join a mailing list where you can ask questions. The UCSC Browser mailing list is at: <http://www.cse.ucsc.edu/mailman/listinfo/genome>

ACKNOWLEDGEMENTS

I am indebted to Mark Diekhans and Hiram Clawson for having spent large amounts of time patiently explaining the workings of the UCSC Browser to me. I thank Ewan Birney, Deanna Church, Xose Fernandez, Fan Hsu, Bob Kuhn, Daryl Thomas and

David Wheeler for answering questions about the features of the various browsers and databases and to Daryl Thomas and Fan Hsu for assistance in developing some of the slide presentation material. And finally I am of course grateful to the three browser teams for creating such outstanding and useful tools in the first place.

REFERENCES

Browser Comparisons

Baxevanis, A.D. (2003) Using genomic databases for sequence-based biological discovery. *Mol Med*, **9**, 185-192.

Furey, T.S. (2006) Comparison of human (and other) genome browsers. *Hum Genomics*, **2**, 266-270.

Ensembl

Birney, E. (2003) Ensembl: a genome infrastructure. *Cold Spring Harb Symp Quant Biol*, **68**, 213-215.

Birney, E., Andrews, D., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cox, T., Cunningham, F., Curwen, V., Cutts, T. *et al.* (2006) Ensembl 2006. *Nucleic Acids Res*, **34**, D556-561.

Birney, E., Andrews, T.D., Bevan, P., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cuff, J., Curwen, V., Cutts, T. *et al.* (2004) An overview of Ensembl. *Genome Res*, **14**, 925-928.

Curwen, V., Eyraas, E., Andrews, T.D., Clarke, L., Mongin, E., Searle, S.M. and Clamp, M. (2004) The Ensembl automatic gene annotation system. *Genome Res*, **14**, 942-950.

Durinck, S., Moreau, Y., Kasprzyk, A., Davis, S., De Moor, B., Brazma, A. and Huber, W. (2005) BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics*, **21**, 3439-3440.

Hammond, M.P. and Birney, E. (2004) Genome information resources - developments at Ensembl. *Trends Genet*, **20**, 268-272.

Kasprzyk, A., Keefe, D., Smedley, D., London, D., Spooner, W., Melsopp, C., Hammond, M., Rocca-Serra, P., Cox, T. and Birney, E. (2004) EnsMart: a generic system for fast and flexible access to biological data. *Genome Res*, **14**, 160-169.

MapViewer

Wheeler, D.L., Barrett, T., Benson, D.A., Bryant, S.H., Canese, K., Chetvernin, V.,

Church, D.M., DiCuccio, M., Edgar, R., Federhen, S. *et al.* (2006) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, **34**, D173-180.

Wheeler, D.L., Barrett, T., Benson, D.A., Bryant, S.H., Canese, K., Church, D.M., DiCuccio, M., Edgar, R., Federhen, S., Helmberg, W. *et al.* (2005) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, **33**, D39-45.

UCSC

Blanchette, M., Kent, W.J., Riemer, C., Elnitski, L., Smit, A.F., Roskin, K.M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E.D. *et al.* (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res*, **14**, 708-715.

Hinrichs, A.S., Karolchik, D., Baertsch, R., Barber, G.P., Bejerano, G., Clawson, H., Diekhans, M., Furey, T.S., Harte, R.A., Hsu, F. *et al.* (2006) The UCSC Genome Browser Database: update 2006. *Nucleic Acids Res*, **34**, D590-598.

Hsu, F., Kent, W.J., Clawson, H., Kuhn, R.M., Diekhans, M. and Haussler, D. (2006) The UCSC Known Genes. *Bioinformatics*, **22**, 1036-1046.

Hsu, F., Pringle, T.H., Kuhn, R.M., Karolchik, D., Diekhans, M., Haussler, D. and Kent, W.J. (2005) The UCSC Proteome Browser. *Nucleic Acids Res*, **33**, D454-458.

Karolchik, D., Hinrichs, A.S., Furey, T.S., Roskin, K.M., Sugnet, C.W., Haussler, D. and Kent, W.J. (2004) The UCSC Table Browser data retrieval tool. *Nucleic Acids Res*, **32**, D493-496.

Kent, W.J., Baertsch, R., Hinrichs, A., Miller, W. and Haussler, D. (2003) Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. *Proc Natl Acad Sci U S A*, **100**, 11484-11489.

Kent, W.J., Hsu, F., Karolchik, D., Kuhn, R.M., Clawson, H., Trumbower, H. and Haussler, D. (2005) Exploring relationships and mining data with the UCSC Gene Sorter. *Genome Res*, **15**, 737-741.

Galaxy


Giardine B., Riemer C., Hardison R.C., *et al.* (2005) Galaxy: a platform for interactive large-scale genome analysis. *Genome Res*, **15**, 1451-1455.

Applications


Schattner, P. and Diekhans, M. (2006) Regions of extreme synonymous codon selection in mammalian genes. *Nucleic Acids Res*, **34**, 1700-1710.

Schattner P., Barberan-Soler S. and Lowe T.M. (2006) A computational screen for mammalian pseudouridylation guide H/ACA RNAs. *RNA*, **12**, 15-25.

14th Annual International Conference On Intelligent Systems For Molecular Biology



I/MB 2006 Fortaleza, Brazil
August 6-10, 2006
and 2nd Annual AB³C Conference: X-Meeting




Genomes, Browsers and Databases: Tools for Automated Data Integration Across Multiple Genomes

Peter Schattner
University of California, Santa Cruz
schattner@cse.ucsc.edu

I/MB 2006 Fortaleza, Brazil August 6-10, 2006

Overview


- *Introduction to Genome Browsers*
- *Basics of the UCSC Browser*
- *Assemblies, Alignments and all that*
- *Batch Browser/Database Querying*
- *Automated Procedures for Database Querying*



98


“Disclaimer”

- *I have tried to be fair in my descriptions of browser and database tools and capabilities, but any evaluation of features like ease of use or tool utility are inherently subjective.*
- *All opinions are solely mine, from the perspective of an “end user”, and may well not be those of the UCSC Browser Team (of which I am not a member.)*



99

Part I - Introduction to Genome Browsers



100

What browsers provide

- *Data visualization*
- *Annotation of genes and genomic locations*
- *Comparisons / alignments of genes and genomic locations*
- *Interactive and automated access to integrated databases*

Why integrated browsers are important

- Visualizing data from multiple sources simultaneously can be critical to understanding.
- The number of databases is growing rapidly.
- Many “databases” are downloadable as flat-files only.
 - Searching is slow or
 - Local relational databases need to be set up
- Differing data formats are used.
 - Consequently, multiple data parsers are often required.
- Complex queries require integrating data from multiple databases.

A simple example

You have found a synonymous codon polymorphism in a possible disease gene and you want to know:

- Is the polymorphism in dbSNP?
- Does it occur in any known EST?
- Is the site conserved in other vertebrates?
- Is it near any “LINE” repeat sequences?
- Is the exon involved alternatively spliced?

What we might do without a browser

- Go to dbSNP & search for SNPs at that location.
- Blast Genbank and retrieve ESTs and parse for polymorphism.
- Blast Genbank and cross-species (xeno) MRNAs and parse for conservation.
- Retrieve sequence near location and check for presence of LINEs (eg with BLAST).
- Go to ASDB (Alternatively Spliced Database) and check for evidence of alternative splicing.

This approach is slow and tedious. Browsers enable a much better way...

An introduction to the browsers and their relative strengths and limitations

The three genome browsers

- There are three main browsers:
 - Ensembl
 - NCBI MapViewer
 - UCSC
- At first glance their main distinguishing features are:
 - MapViewer is arranged vertically.
 - Ensembl has multiple (22) different “Views”.
 - UCSC has a single “View” for (almost) everything.

Choosing a browser

- In general, the browsers have more similarities than differences.
- Also, the development teams are competitive (in a cooperative way); if a site doesn't have a feature now, it may have it soon.
- But the browsers do have different strengths. In particular, some species are covered by only one browser.
- It's probably best to find one browser you like and stick with it for most tasks.

NCBI MapViewer

MapViewer Home

NCBI Map Viewer

Search: for

Click on the organism name to go to the genome view

Vertebrates

Mammals

BLAST *Bos taurus* (cow)

BLAST *Canis familiaris* (dog)

BLAST *Felis catus* (cat)

BLAST *Homo sapiens* (human) Build 36

BLAST *Homo sapiens* (human) Build 35

BLAST *Mus musculus* (mouse) Build 36

BLAST *Mus musculus* (mouse) Build 35

BLAST *Ovis aries* (sheep)

BLAST *Pan troglodytes* (chimpanzee)

BLAST *Rattus norvegicus* (rat)

BLAST *Sus scrofa* (pig)

Other Vertebrates

BLAST *Danio rerio* (zebrafish)

BLAST *Gallus gallus* (chicken)

Invertebrates

Insects

BLAST *Anopheles gambiae* (mosquito)

BLAST *Apis mellifera* (honey bee)

BLAST *Drosophila melanogaster* (fruit fly)

BLAST *Tribolium castaneum* (red flour beetle)

Nematode

BLAST *Caenorhabditis elegans* (nematode)

Echinoderms

BLAST *Strongylocentrotus purpuratus* (purple sea urchin)

Plants BLAST Search all plant maps

BLAST *Arabidopsis thaliana* (thale cress)

BLAST *Avena sativa* (oat)

BLAST *Glycine max* (soybean)

BLAST *Hordeum vulgare* (barley)

BLAST *Lycopersicon esculentum* (tomato)

BLAST *Mannihot esculenta* (cassava)

BLAST *Oryza sativa* (rice)

BLAST *Triticum aestivum* (wheat)

BLAST *Zea mays* (corn)

Fungi BLAST

BLAST *Aspergillus fumigatus*

BLAST *Candida glabrata*

BLAST *Cryptococcus neoformans*

BLAST *Debaryomyces hansenii*

BLAST *Encephalitozoon cuniculi*

BLAST *Eremothecium gossypii*

BLAST *Gibberella zeae*

BLAST *Kluyveromyces fragilis*

BLAST *Magnaporthe oryzae*

BLAST *Neurospora crassa*

BLAST *Saccharomyces cerevisiae* (baker's yeast)

BLAST *Schizosaccharomyces pombe* (fission yeast)

BLAST *Ustilago maydis*

BLAST *Yarrowia lipolytica*

<http://www.ncbi.nlm.nih.gov/mapview/>

109

MapViewer Master Map

Region Displayed: 43M-48M bp

Download/View Sequence/View

Gene	Symbol	Q	LinkOut	E	Cyto	Description
GALNACT-2	+	sv pr dl mm hm ccds	C	10q11.21	chondroitin sulfate GalNACT-2	
RASGEF1A	+	sv pr dl mm hm ccds	C	10q11.21	RasGEF domain family, member 1A	
FXDY4	+	sv pr dl mm hm ccds	C	10q11.21	FXDY domain containing ion transport regulator 4	
HNRPF	+	OMIM sv pr dl mm hm ccds	C	10q11.21-q11.22	heterogeneous nuclear ribonucleoprotein F	
ZNF239	+	OMIM sv pr dl mm hm	C	10q11.22-q11.23	zinc finger protein 239	
ZNF485	+	sv pr dl mm hm ccds	C	10q11.21	zinc finger protein 485	
ZNF32	+	OMIM sv pr dl mm hm ccds	C	10q22-q25	zinc finger protein 32 (KOX 30)	
HNRPA3P1	+	sv dl mm	C	10q11.21	heterogeneous nuclear ribonucleoprotein A3 pseudogene 1	
CXCL12	+	OMIM sv pr dl mm hm ccds	C	10q11.1	chemokine (C-X-C motif) ligand 12 (stromal cell-derived factor 1)	
C10orf127	+	sv pr dl mm	C	10q11.21	chromosome 10 open reading frame 127	
RASSF4	+	sv pr dl mm hm ccds	C	10q11.21	Ras association (RalGDS/AF-6) domain family 4	
C10orf10	+	sv pr dl mm ccds	C	10q11.21	chromosome 10 open reading frame 10	
ZNF22	+	OMIM sv pr dl mm hm ccds	C	10q11	zinc finger protein 22 (KOX 15)	
LOC338579	+	sv pr dl mm	C	10q11.21	hypothetical protein LOC338579	
OR6D1P	+	sv dl mm	C	10q11.21	olfactory receptor, family 6, subfamily D, member 1 pseudogene	
OR13A1	+	sv pr dl mm hm	C	10q11.21	olfactory receptor, family 13, subfamily A, member 1	

110

Selecting tracks on MapViewer

Organism: [Help](#)

Chromosome: Region Shown:

Available Maps: Assembly:

Maps Displayed (left to right):

---Sequence Maps---

Ab initio

Assembly

BES Clone

Clone

Component

Contig

CpG Island

estBt

[] [Celera] ugHs

[] [Celera] Gene

[R] [Celera] Phenotype

More Options:

☐ Show Connections ☒ Verbose Mode

Compress Map: Auto Compress if > px

Page Length:

Thumbnail View: ☒ default (ideogram) ☐ master

111

MapViewer strengths

- Good coverage of plant and fungal genomes.
- Close integration with other NCBI tools and databases, such as Model Maker, trace archives or Celera assemblies.
- Vertical view enables convenient overview of regional gene descriptions.
- Discontiguous MEGABLAST is probably the most sensitive tool available for cross-species sequence queries.
- Ability to view multiple assemblies (e.g. Celera and reference) simultaneously.

112

MapViewer limitations

- Little cross-species conservation or alignment data.
- Inability to upload custom annotations and data.
- Limited capability for batch data access.
- Limited support for automated database querying.
- Vertical view makes base-pair level annotation cumbersome.



113

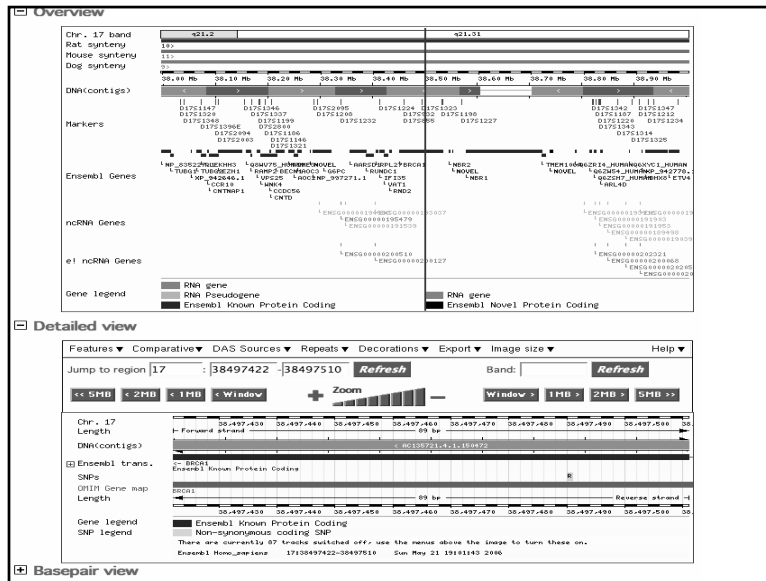
Ensembl



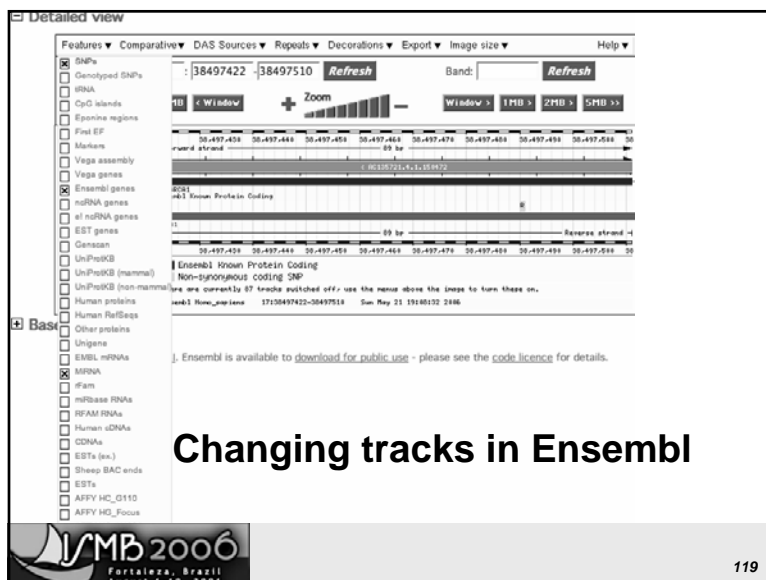
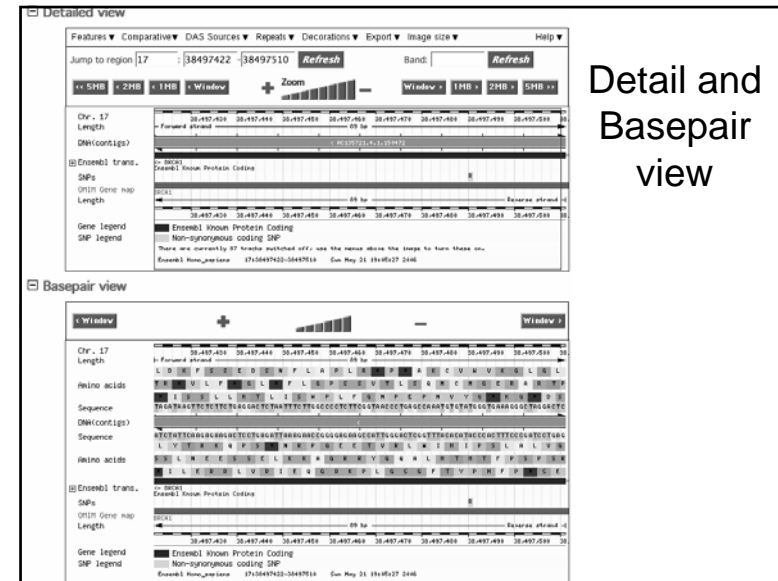
114

http://www.ensembl.org/

115

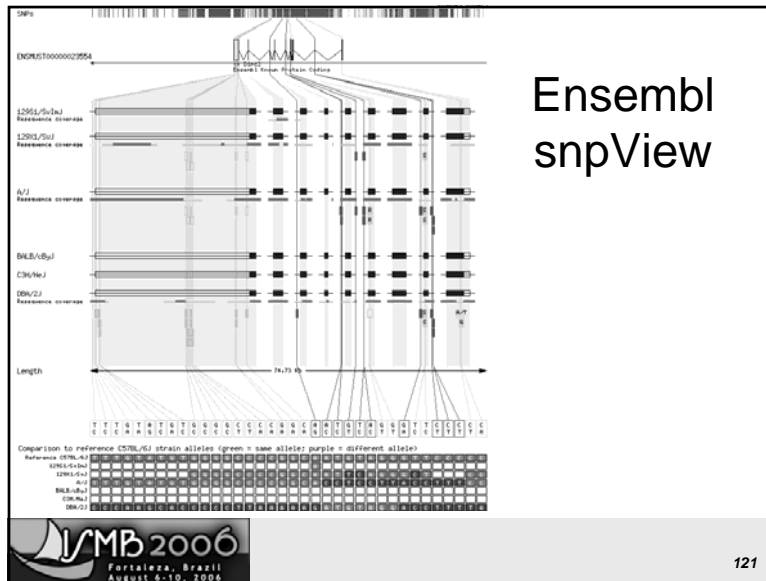


Detail and Basepair view



Ensembl strengths (I)

- Multiple view levels shows genomic context.
- Some annotations are more complete and/or are more clearly presented (e.g. snpView of multiple mouse strain data.)
- Possible to create query over more than one genome database at a time (with BioMart).



Ensembl strengths (II)

- Batch and automated querying well supported and documented (especially for perl and java).
- API (programmer interface) is designed to be identical for all databases in a release.
- Ensembl tends to be more “community oriented” - using standard, widely used tools and data formats.
- All data and code are completely free to all.

Ensembl is “community oriented”

- *Close alliances with Wormbase, Flybase, SGD*
- *“support for easy integration with third party data and/or programs” – BioMart*
- *Close integration with R/ Bioconductor software*
- *More use of community standard formats and programs, e.g. DAS, GFF/GTF, Bioperl*

(Note: UCSC also supports GFF/GTF and is compatible with R/Bioconductor and DAS, but UCSC tends to use more “homegrown” formats, e.g. BED, PSL, and tools.)

Ensembl limitations

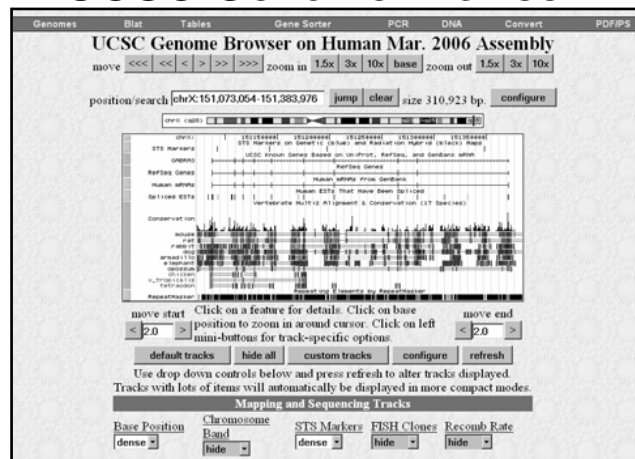
- Limited data quantifying cross-species sequence conservation.
- Batch queries for intergenic regions with BioMart are difficult.
- BioMart offers less complete access to database than UCSC Table Browser. (However, the user interface to BioMart is easier.)

UCSC Genome Browser



125

UCSC Genome Browser



127

Strengths of the UCSC Browser (I)

For this course I will be focusing primarily on the UCSC Browser for several reasons:

- Strong comparative genomics capabilities.
- Fast response
 - sequence searches performed with BLAT.
 - code is written in speed-optimized C.
 - Multiple indexing and non-normalized tables for fast database retrieval.
- (Essentially) single “view” from single base-pair to entire chromosome.
- Easiest interface for loading custom annotations.



128

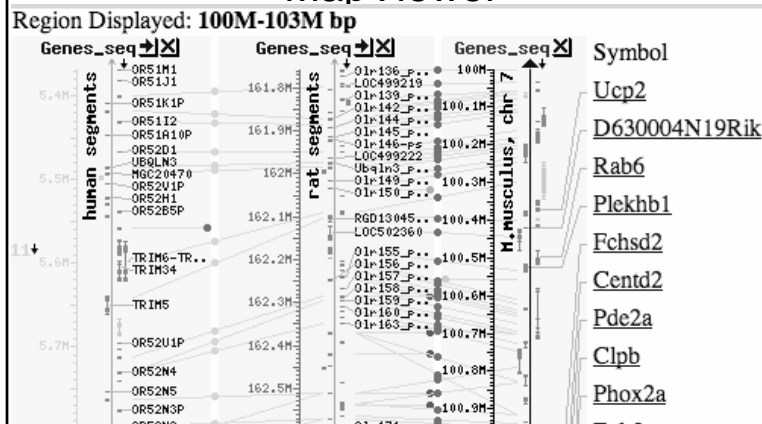
UCSC Browser Strengths (II)

- Well suited for batch and automated querying of both gene and intergenic regions.
- Comprehensive: tends to have the most species, genes and annotations.
- Annotations frequently updated (Genbank/Refseq daily / ESTs weekly).
- Able to find “similar” genes easily with GeneSorter.
- Rapid access to *in situ* images with VisiGene.

UCSC browser limitations

- Lack of “overview” mode can make it harder to see genomic context.
- Syntenic regions cannot be viewed simultaneously.
- Cross species sequence queries with BLAT are often insensitive.
- Comprehensiveness of database can make user interface intimidating.
- Code access for commercial users requires licensing.

Human, mouse, rat synteny in MapViewer



Part II – Interactive Querying on the UCSC Browser

Getting data from the UCSC browser

- You first need to go to:
<http://genome.ucsc.edu>
- Next choose a genome and an assembly.
- Then pick a region.
- Finally specify what annotations you want.



133

Request:	Genome Browser Response:
chr7	Displays all of chromosome 7
20p13	Displays region for band p13 on chr 20
chr3:1-1000000	Displays first million bases of chr 3, counting from p arm telomere
D16S3046	Displays region around STS marker D16S3046 from the Genethon/Marshfield maps. Includes 100,000 bases on each side as well.
RH18061;RH80175	Displays region between STS markers RH18061;RH80175. Includes 100,000 bases on each side as well.
AA205474	Displays region of EST with GenBank accession AA205474 in BRCA1 cancer gene on chr 17
AC008101	Displays region of clone with GenBank accession AC008101
AF083811	Displays region of mRNA with GenBank accession number AF083811
PRNP	Displays region of genome with HUGO Gene Nomenclature Committee identifier PRNP
NM_017414	Displays the region of genome with RefSeq identifier NM_017414
NP_059110	Displays the region of genome with protein accession number NP_059110
pseudogene mRNA	Lists transcribed pseudogenes, but not cDNAs
homeobox caudal	Lists mRNAs for caudal homeobox genes
zinc finger	Lists many zinc finger mRNAs
kruppel zinc finger	Lists only kruppel-like zinc fingers
huntington	Lists candidate genes associated with Huntington's disease
zabier	Lists mRNAs deposited by scientist named Zabier
Evans,J.E.	Lists mRNAs deposited by co-author J.E. Evans

Request:	Genome Browser Response:
chr7	Displays all of chromosome 7
20p13	Displays region for band p13 on chr 20
chr3:1-1000000	Displays first million bases of chr 3, counting from p arm telomere
D16S3046	Displays region around STS marker D16S3046 from the Genethon/Marshfield maps. Includes 100,000 bases on each side as well.
RH18061;RH80175	Displays region between STS markers RH18061;RH80175. Includes 100,000 bases on each side as well.
AA205474	Displays region of EST with GenBank accession AA205474 in BRCA1 cancer gene on chr 17
AC008101	Displays region of clone with GenBank accession AC008101
AF083811	Displays region of mRNA with GenBank accession number AF083811
PRNP	Displays region of genome with HUGO Gene Nomenclature Committee identifier PRNP
NM_017414	Displays the region of genome with RefSeq identifier NM_017414
NP_059110	Displays the region of genome with protein accession number NP_059110
pseudogene mRNA	Lists transcribed pseudogenes, but not cDNAs
homeobox caudal	Lists mRNAs for caudal homeobox genes
zinc finger	Lists many zinc finger mRNAs
kruppel zinc finger	Lists only kruppel-like zinc fingers
huntington	Lists candidate genes associated with Huntington's disease
zabier	Lists mRNAs deposited by scientist named Zabier
Evans,J.E.	Lists mRNAs deposited by co-author J.E. Evans

Ways to specify location

- Directly (*chr5:1000000-2000000*)
- Via name or term (e.g. *BRCA1*)
 - Many, more complex query terms are possible
 - See the Genome “Gateway” page for examples.
- Via BLAT query



136



Genome Browser Navigation

Navigation, Species, and Assembly

Zoom and Position

Mapping and Sequence

Base Position, Chromosome Band, STS markers, Gap

Genes and Gene Predictions

Known Genes, RefSeq, Ensembl, Acembly, Genscan

mRNAs and ESTs

Human mRNAs from GenBank, Human ESTs that have been spliced

Gene Expression and Regulation

GNF Ratios on Affymetrix GeneChips

Comparative Genomics

Multiple Alignments, Conservation Scores, Chain / Net

Variation and Repeats

Start / End Position Adjustment

Reset / Hide / Refresh

Color Key for Chain / Net / Self tracks

Modified from D. Thomas, IEEE CSB Tutorial (2004)

Track overview

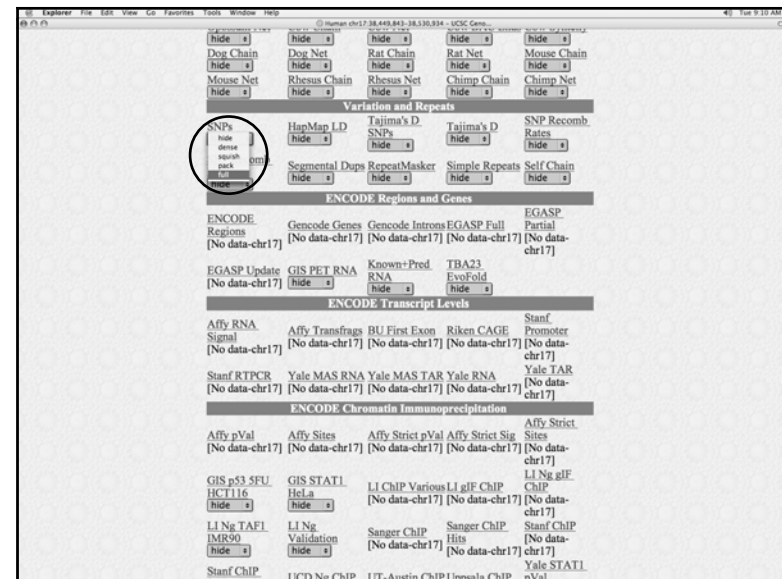
- Annotations are via “tracks” along the chromosome.
- Annotations are divided in track “Groups.”
- Track availability will vary depending on the organism and assembly.
- Scores of tracks exist and new tracks are being added all the time (see Hinrichs, NAR 2006 for a recent update).

Track types:

- *Chromosome descriptions*
- *Genes & gene predictions*
- *Gene annotations*
- *Local paired alignments (mRNAs, ESTs)*
- *Comparative genomics*
 - Genomic paired alignments
 - Genomic multiple alignments
- *Sequence variations*

Recently added tracks

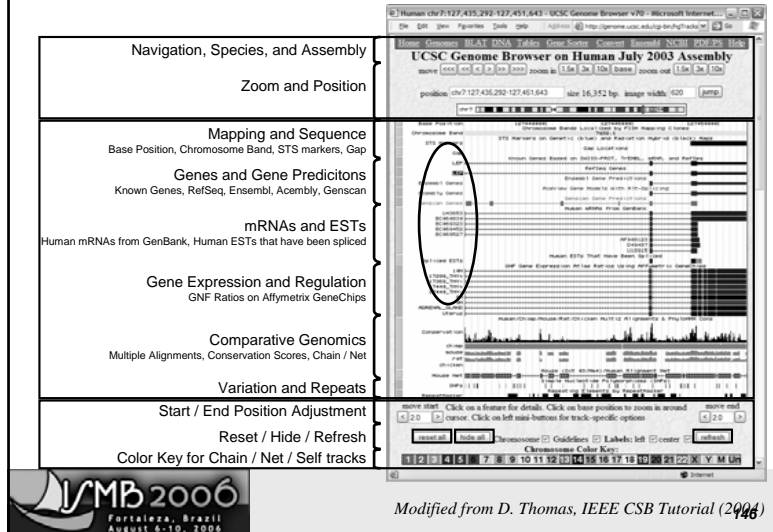
- *ENCODE annotations*
- *Consensus CDS annotations*
- *Retroposed genes*
- *snoRNAs / miRNAs*
- *RNA fold predictions*
- *Segmental duplications*
- *Copy number polymorphisms*
- *Affymetrix full chromosome transcriptional data*
- *Mammalian Gene Consortium data*



Annotation “Details”

- Every annotation has an associated “Details” page.
- What is included depends on:
 - Genome
 - Assembly
 - Track type
 - Data available

Genome Browser Navigation



The screenshot shows the UCSC Genome Browser interface for Human July 2003 Assembly. The interface includes a navigation bar at the top with links like Home, Genomes, BLAT, DNA, Tables, Gene Sorter, Content, Ensembl, NCBI, and PDB. Below this, a navigation panel on the left lists various tracks and their functions:

- Navigation, Species, and Assembly:** Zoom and Position
- Mapping and Sequence:** Base Position, Chromosome Band, STS markers, Gap
- Genes and Gene Predictions:** Known Genes, RefSeq, Ensembl, Acembly, Genscan
- mRNAs and ESTs:** Human mRNAs from GenBank, Human ESTs that have been spliced
- Gene Expression and Regulation:** GNF Ratios on Affymetrix GeneChips
- Comparative Genomics:** Multiple Alignments, Conservation Scores, Chain / Net
- Variation and Repeats:** Start / End Position Adjustment, Reset / Hide / Refresh, Color Key for Chain / Net / Self tracks

The main display area shows a genomic track with various data points, including a red circle highlighting a specific region. The bottom of the interface includes a chromosome color key and a navigation bar with buttons like 'Home', 'Genomes', 'BLAT', 'DNA', 'Tables', 'Gene Sorter', 'Content', 'Ensembl', 'NCBI', and 'PDB'.

Modified from D. Thomas, IEEE CSB Tutorial (2004)

Details Page of a Known Gene



The screenshot shows the Human Gene LEP Description and Page Index page. The page includes a navigation bar at the top with links like Home, Genomes, Genome Browser, Gene Sorter, BLAT, PCR, Tables, FAQ, and Help. Below this, the page title is "Human Gene LEP Description and Page Index". The main content area includes a description of the gene, a list of quick links to various databases and tools, and a section for comments and description text from SwissProt.

Human Gene LEP Description and Page Index

Description: leptin (obesity homolog, mouse)
Representative mRNA: BC060550 **Protein:** P41159 (OB_HUMAN)
RefSeq Summary: This gene is similar to the mouse obesity gene (ob). The protein encoded by this gene is secreted by white adipocytes. In the mouse study, mutations in this gene are linked to severe and morbid obesity.

Page Index: Quick Links | SwissProt Comments | Sequence | Microarray | Protein Structure
 Other Species | GO Annotations | mRNA Descriptions | Pathways | Methods

Quick Links to Tools and Databases

Genome Browser	Protein Browser	Gene Sorter	SwissProt	LocustLink	Entrez Gene
PubMed	OMIM	GeneLink	GeneCards	CGAP	Stanford SOURCE
Jackson Labs					

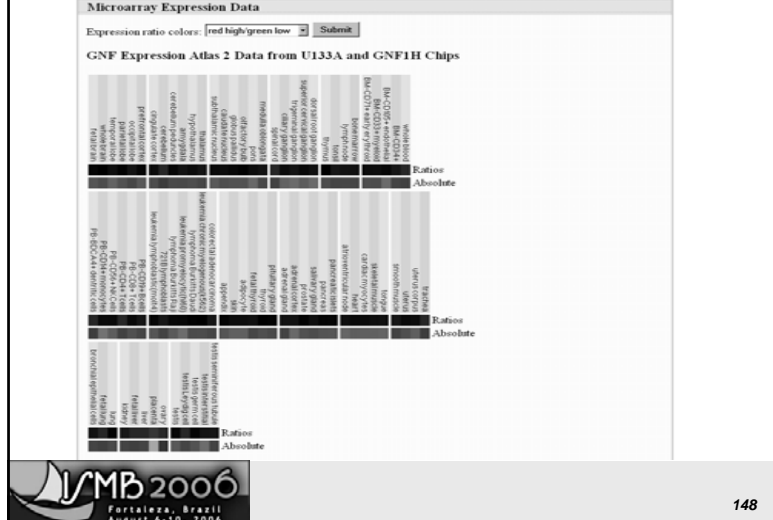
Comments and Description Text from SwissProt

ID: OB_HUMAN
DESCRIPTION: Leptin precursor (Obesity factor) (Obese protein).
FUNCTION: May function as part of a signaling pathway that acts to regulate the size of the body fat depot. An increase in the level of LEP may act directly or indirectly on the CNS to inhibit food intake and/or regulate energy expenditure as part of a homeostatic mechanism to maintain constancy of the adipose mass.
SUBUNIT: Interacts with SIGLEC6.
SUBCELLULAR LOCATION: Secreted.
DISEASE: Defects in LEP may be a cause of autosomal recessive obesity [MIM:601663].
SIMILARITY: Belongs to the leptin family.
DATABASE: NAME=RefSeq Systems' cytokine mini-review: LEP.
WWW: http://www.ncbi.nlm.nih.gov/seq/seqbuilder.asp?bodyid=2137.

Sequence

Genomic (chr7:127,475,283-127,491,632): mRNA (many differ from genome): Protein (167 aa)

Known Gene Details (continued)



The screenshot shows the Microarray Expression Data page for the LEP gene. The page includes a navigation bar at the top with links like Home, Genomes, Genome Browser, Gene Sorter, BLAT, PCR, Tables, FAQ, and Help. Below this, the page title is "Microarray Expression Data". The main content area includes a table of expression data for various tissues and cell lines, with a color scale for expression ratio (red for high, green for low).

Microarray Expression Data

Expression ratio colors: red high/green low Submit

GNF Expression Atlas 2 Data from U133A and GNF1H Chips

Gene	U133A	GNF1H	Ratio	Absolute
LEP	High	Low	High	Low
OB	High	Low	High	Low
OBP	High	Low	High	Low
OBP2	High	Low	High	Low
OBP3	High	Low	High	Low
OBP4	High	Low	High	Low
OBP5	High	Low	High	Low
OBP6	High	Low	High	Low
OBP7	High	Low	High	Low
OBP8	High	Low	High	Low
OBP9	High	Low	High	Low
OBP10	High	Low	High	Low
OBP11	High	Low	High	Low
OBP12	High	Low	High	Low
OBP13	High	Low	High	Low
OBP14	High	Low	High	Low
OBP15	High	Low	High	Low
OBP16	High	Low	High	Low
OBP17	High	Low	High	Low
OBP18	High	Low	High	Low
OBP19	High	Low	High	Low
OBP20	High	Low	High	Low
OBP21	High	Low	High	Low
OBP22	High	Low	High	Low
OBP23	High	Low	High	Low
OBP24	High	Low	High	Low
OBP25	High	Low	High	Low
OBP26	High	Low	High	Low
OBP27	High	Low	High	Low
OBP28	High	Low	High	Low
OBP29	High	Low	High	Low
OBP30	High	Low	High	Low
OBP31	High	Low	High	Low
OBP32	High	Low	High	Low
OBP33	High	Low	High	Low
OBP34	High	Low	High	Low
OBP35	High	Low	High	Low
OBP36	High	Low	High	Low
OBP37	High	Low	High	Low
OBP38	High	Low	High	Low
OBP39	High	Low	High	Low
OBP40	High	Low	High	Low
OBP41	High	Low	High	Low
OBP42	High	Low	High	Low
OBP43	High	Low	High	Low
OBP44	High	Low	High	Low
OBP45	High	Low	High	Low
OBP46	High	Low	High	Low
OBP47	High	Low	High	Low
OBP48	High	Low	High	Low
OBP49	High	Low	High	Low
OBP50	High	Low	High	Low
OBP51	High	Low	High	Low
OBP52	High	Low	High	Low
OBP53	High	Low	High	Low
OBP54	High	Low	High	Low
OBP55	High	Low	High	Low
OBP56	High	Low	High	Low
OBP57	High	Low	High	Low
OBP58	High	Low	High	Low
OBP59	High	Low	High	Low
OBP60	High	Low	High	Low
OBP61	High	Low	High	Low
OBP62	High	Low	High	Low
OBP63	High	Low	High	Low
OBP64	High	Low	High	Low
OBP65	High	Low	High	Low
OBP66	High	Low	High	Low
OBP67	High	Low	High	Low
OBP68	High	Low	High	Low
OBP69	High	Low	High	Low
OBP70	High	Low	High	Low
OBP71	High	Low	High	Low
OBP72	High	Low	High	Low
OBP73	High	Low	High	Low
OBP74	High	Low	High	Low
OBP75	High	Low	High	Low
OBP76	High	Low	High	Low
OBP77	High	Low	High	Low
OBP78	High	Low	High	Low
OBP79	High	Low	High	Low
OBP80	High	Low	High	Low
OBP81	High	Low	High	Low
OBP82	High	Low	High	Low
OBP83	High	Low	High	Low
OBP84	High	Low	High	Low
OBP85	High	Low	High	Low
OBP86	High	Low	High	Low
OBP87	High	Low	High	Low
OBP88	High	Low	High	Low
OBP89	High	Low	High	Low
OBP90	High	Low	High	Low
OBP91	High	Low	High	Low
OBP92	High	Low	High	Low
OBP93	High	Low	High	Low
OBP94	High	Low	High	Low
OBP95	High	Low	High	Low
OBP96	High	Low	High	Low
OBP97	High	Low	High	Low
OBP98	High	Low	High	Low
OBP99	High	Low	High	Low
OBP100	High	Low	High	Low

Known Gene Details (continued)

Protein Domain and Structure Information

InterPro Domains: [Graphical view of domain structure](#)
 IPR009079 - Pore-helical cytokine
 IPR009065 - Obesity factor

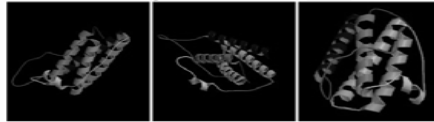
Pfam Domains:
 PF02024 - Leptin

Protein Data Bank (PDB) 3-D Structure



LAXS - X-ray

ModBase Predicted Comparative 3D Structure on P41159



The pictures above may be empty if there is no ModBase structure for the protein. The ModBase structure frequently covers just a fragment of the protein. You may be asked to log onto ModBase the first time you click on the pictures. It is simplest after logging in to just click on the picture again to get to the specific info on that model.



149

Known Gene Details (continued)

Homologous Genes in Other Species (BLASTP Best Hit)

Mouse	Rat	Zebrafish	D. melanogaster	C. elegans	S. cerevisiae
Gene Browser	Gene Browser	No homolog	No homolog	No homolog	No homolog
Gene Details	Gene Details				
Gene Sorter	Gene Sorter				
Jackson Labs	RGD				
Protein Sequence	Protein Sequence				
Alignment	Alignment				

Gene Ontology (GO) Annotations with Structured Vocabulary

Molecular Function:
 GO:0005178 hormone activity

Biological Process:
 GO:0006112 energy reserve metabolism
 GO:0007165 signal transduction
 GO:0007262 cell-cell signaling

Cellular Component:
 GO:0005578 extracellular
 GO:0005612 extracellular space

Descriptions from all associated mRNAs.

U34653 - Human obese protein (ob) mRNA, complete cds.
 R326038 - Homo sapiens leptin (obesity homolog, mouse), mRNA (cDNA clone MGC 71704 IMAGE 30333193), complete cds.
 AF081123 - Homo sapiens obese protein (ob) mRNA, complete cds.
 U34657 - Homo sapiens obese mRNA, complete cds.
 U11815 - Human obese (ob) mRNA, complete cds.
 R326032 - Homo sapiens leptin (obesity homolog, mouse), mRNA (cDNA clone MGC 96838 IMAGE 7262097), complete cds.
 R326045 - Homo sapiens leptin (obesity homolog, mouse), mRNA (cDNA clone MGC 96908 IMAGE 7262109), complete cds.
 R326057 - Homo sapiens leptin (obesity homolog, mouse), mRNA (cDNA clone MGC 96912 IMAGE 7262121), complete cds.

Biochemical and Signalling Pathways

KEGG - Kyoto Encyclopedia of Genes and Genomes
 hsa04500 - Cytokine-cytokine receptor interaction - Homo sapiens
 hsa04630 - Jak-STAT signaling pathway - Homo sapiens

BioCarta from NCI Cancer Genome Anatomy Project
 h_leptinPathway - Reversal of Insulin Resistance by Leptin

Quick look at other features of the UCSC browser

- *GeneSorter*
- *Protein Browser*
- *VisiGene*
- *isPCR*



151

The Gene Sorter

- Finds genes that are “closely related” to specified gene.
- “Closeness” can be specified by:
 - Protein homology
 - Expression patterns
 - Chromosome location
 - Gene function (GO annotations) etc.
- The gene properties displayed by the Gene Sorter are also highly configurable.



152

Gene Sorter Interface

UCSC Human Gene Sorter


genome: Human assembly: July 2003 search: BC050072 Go!

sort by: Expression (GNF Atlas2) configure filter (now off) display: 25 output: sequence text

Info Sorting Types for Gene Sorter

[Return to Main Page](#)

Type	Description
Expression (GNF Atlas2)	Difference in Expression with Selected Gene According to GNF Gene Expression Atlas2
Expression (GNF Atlas1)	Difference in Expression with Selected Gene
Protein Homology	Blastp E-value with Selected Gene
Pfam Similarity	Number of Pfam Domains Shared with Selected Gene
Gene Distance	Distance in Base Pairs from Selected Gene
Chromosome	Ordered by Chromosome Position
Name Similarity	Number of Leading Characters in Name that Match Selected Gene Name
Alphabetical	Alphabetical Order of Name
GO Similarity	Number of Shared Gene Ontology Terms

 Fortaleza, Brazil August 6-10, 2006 153


Typical Gene Sorter Results

Home UCSC Human Gene Sorter

genome: Human assembly: July 2003 search: BC050072 Go!

sort by: Expression (GNF Atlas2) configure filter (now off) display: 25 output: sequence text

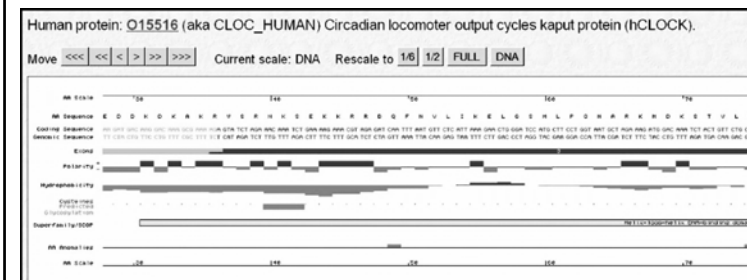
#	Name	whole brain	amygdala	bone marrow	PE-CD4+ T cells	adipocyte	heart	pancreas	kidney	ovary	testis	E-Value	Genome Position	Description
1	FOXP1B										3e-159	chr14 27,226,998	Similar to forkhead box O1.	
2	LHX2										n/a	chr9 122,161,094	LIM homeobox 2	
3	SYT1										n/a	chr12 78,230,531	synaptotagmin I	
4	LASS1										n/a	chr19 18,854,131	LAG1 longevity assurance homolog 1 (S. cerevisiae)	
5	EGR4										n/a	chr2 73,493,876	early growth response 4	
6	HPCA										n/a	chr1 32,858,645	hippocalcin	
7	DOCK3										n/a	chr3 51,026,030	DOCK3	
8	ARNT2										n/a	chr15 78,509,325	Hypothetical protein	

 Fortaleza, Brazil August 6-10, 2006 154

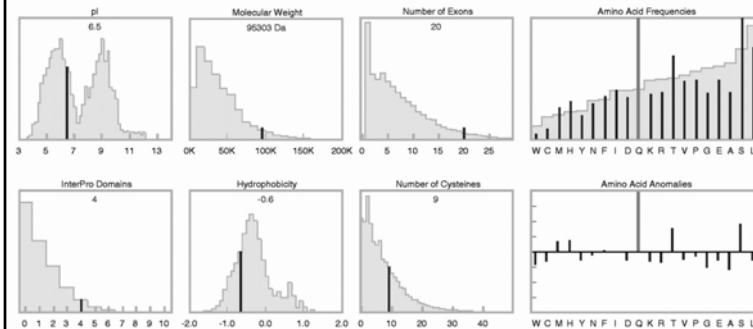
Proteome Browser

- The Proteome Browser displays annotations that are protein specific.
- Its format is analagous to Ensembl's gene- and protein-oriented "Views".

Proteome Browser - amino acid structure



Proteome Browser - physical / chemical properties



Modified from F. Hsu, NAR (2005) 157

Protein structure and links

UCSC links:

- Genome Browser - AB002332
- Gene Details Page - AB002332
- Gene Sorter - AB002332

InterPro Domains: Graphical view of domain structure

- IPR001092 - MH_basic
- IPR001097 - Nrc_translocat
- IPR001610 - PAC
- IPR000014 - PAC

Pfam Domains:

- PF00010 - Helix-loop-helix DNA-binding domain
- PF00703 - PAC motif
- PF00003 - PAC domain

ModBase Predicted Comparative 3D Structure on Q15516

Front Top Side

The pictures above may be empty if there is no ModBase structure for the protein. The ModBase structure frequently covers just a fragment of the protein. You may be asked to log onto ModBase the first time you click on the pictures. It is simplest after logging in to just click on the picture again to get to the specific info on that model.

Pathways:

- BioCarta - circadianPathway - Circadian Rhythms
- KEGG - hsa04710 - Circadian rhythm - Homo sapiens



Modified from F. Hsu, NAR (2003) 158

VisiGene

VisiGene Image Browser

VisiGene is a virtual microscope for viewing *in situ* images. These images show where a gene is used in an organism, sometimes down to cellular resolution. With VisiGene users can retrieve images that meet specific search criteria, then interactively zoom and scroll across the collection.

Good search terms include gene symbols, authors, years, body parts, organisms, GenBank and UniProt accessions, Known Gene descriptive terms, Theiler stages for mice, and Nieuwkoop/Faber stages for frogs. The wildcard characters * and ? work with gene symbols; otherwise the full word must match.

Sample queries

Request:	VisiGene Response:
nkx2-2	Displays images associated with the gene nkx2-2
hoxa*	Displays images of all genes in the Hox-A cluster (Note: * works only at the end of the word)
NM_007492	Displays images associated with accession NM_007492
theiler 22	Displays all images that show Theiler stage 22
vgPb_16	Displays images associated with VisiGene probe ID 16
allen institute	Displays all images from the Allen Brain Atlas
mouse	Displays all mouse images
xenopus	Displays all images associated with frogs of genus Xenopus
mouse midbrain	Displays mouse images that show expression in the midbrain
smith jc 1994	Displays images contributed by scientist J.C. Smith in 1994

Images Available

The following image collections are currently available for browsing:

- High-quality high-resolution images of eight-week-old male mouse sagittal brain slices with reverse-complemented mRNA hybridization probes from the Allen Brain Atlas, courtesy of the Allen Institute for Brain Science
- Mouse *in situ* images from the Jackson Labs Gene Expression Database (GXD) at MGI
- Transcription factors in mouse embryos from the Mahoney Center for Neuro-Oncology

VisiGene Hoxa9

163 images match

Mouse Hoxa9

Mouse Hoxa10

Mouse Hoxa10

Mouse Hoxa11

Mouse Hoxa13

source: Mahoney Lab Reference: Mouse brain organization revealed through direct genome-scale TF expression analysis

Year: 2004 Contributors: Gray P.A., Fu H., Luo P., Zhao Q., Yu J., Ferrari A., Tenzen T., Yuk D., Tsung E.F., Cai Z., Alberta J.A., Cheng L., Liu Y., Stenman J.M., Valerius M.T., Billings N., Kim H.A., Greenberg M.E., McMahon A.P., Rowitch D.H., Stiles C.D., Ma

In-Silico PCR Input

[Home](#)[Genomes](#)[Blat](#)[Tables](#)[Gene Sorter](#)[FAQ](#)[Help](#)

UCSC In-Silico PCR

Genome:
Human

Assembly:
May 2004

Forward Primer:

Reverse Primer:

submit

Max Product Size: 10000

Min Perfect Match: 15

Min Good Match: 15

Flip Reverse Primer: ☐

About In-Silico PCR

In-Silico PCR searches a sequence database with a pair of PCR primers, using an indexing strategy for fast performance.

Configuration Options

Genome and Assembly - The sequence database to search.

Forward Primer - Must be at least 15 bases in length.

Reverse Primer - On the opposite strand from the forward primer. Minimum length of 15 bases.

Max Product Size - Maximum size of amplified region.

Min Perfect Match - Number of bases that match exactly on 3' end of primers. Minimum match size is 15.

Min Good Match - Number of bases on 3' end of primers where at least 2 out of 3 bases match.

Flip Reverse Primer - Invert the sequence order of the reverse primer and complement it.

About

When successful, the search returns a sequence output file in fasta format containing all sequence in the database that lie between and include the primer pair. The fasta header describes the region in the database and the primers. The fasta body is capitalized in areas where the primer sequence matches the database sequence and in lower-case elsewhere. Here is an example:

```
>chr22:31,000,551-31,001,506  TAAACGATTGATGATGATGAATAGG  CCATGATGGCTGCTAAAGCAGCTGC
TAAACGATTGATGATGATGAATAGGCGGGTGGCGCGGGTGGGGTGG
gaactgcagagaagaagcagggtcggttcataaacaagctttgtgtccocaa
tatgcacagctgaagtcttccaggggtgatgtgtgagcagcaggtgaggttaag
taccacaaacactccctcagaaacactccctcactccctcagaggttaaaataaa
```

- We now return to our example of using the browser to characterize the region around the putative disease polymorphism.
- We need to:
 - Find the specific region with BLAT
 - Set up the tracks to display SNPs, ESTs, repeats, alternative splicing patterns and alignments with other vertebrates
 - Download EST sequences or view with browser
 - View the results



Human BLAT Search

BLAT Search Genome

Genome: Assembly: Query type: Sort output: Output type:

Paste in a query sequence to find its location in the genome. Multiple sequences may be searched if separated by lines starting with '>' followed by the sequence name.

File Upload: Rather than pasting a sequence, you can choose to upload a text file containing the sequence.

Upload sequence:

164

snp, repeat and conservation results at polymorphism site



Part III

Assemblies, alignments and all that - a glimpse inside the browsers

Part III - overview

- Assemblies, builds and tracks
- Local alignment tools
- Genomic alignment tools
- Preview, Auxiliary and Development Browsers

Assemblies, builds and tracks

- The UCSC browser uses chromosome coordinates.
- As a result, each new assembly of a genome will change the positions of most features (i.e. annotations).
- Consequently after a genome reassembly the entire database needs to be rebuilt.

Assemblies, builds and tracks

- A browser utility exists to convert coordinates between assemblies.
- One source of confusion is that new builds may not have all the tracks of a previous build.
- Moreover many annotations are updated between builds (in some cases daily). This can be confusing when comparing with results obtained previously from the same build.

Lift Genome Annotations

This tool converts genome coordinates and genome annotation files between assemblies. The input data can be pasted into the text box, or uploaded from a file.

Original Genome:
 Original Assembly:
 New Genome:
 New Assembly:

Minimum ratio of bases that must remap:
 Minimum chain size in target:
 Minimum hit size in query:
 Allow multiple output regions: ☐
 Min ratio of alignment blocks/exons that must map:
 If thickStart/thickEnd is not mapped, use the closest mapped base: ☐

For descriptions of the supported data formats, see the bottom of this page.

Data Format:

Paste in data:

```

chr1 28654492 28654699 E1 206 +
chr3 39427528 39427722 E2 194 +
chr3 187987769 187987944 E3 175 +
chr5 138642347 138642587 U19 240 +
chr2 232145995 232146172 U23 177 -
chr16 1952954 1953128 U64 174 -
chr9 127290314 127290490 U65 176 -
chr1 93018276 93018449 U66 173 +
chr17 7421976 7422153 U67 177 +
chr19 17834376 17834549 U68 173 +
  
```

Submit

Clear

UCSC Alignment Algorithms

- Much of the power of the UCSC Browser comes from its comparative genomics tools.
- These in turn come from its alignment tools, including tools for:
 - Paired and multiple sequence alignment
 - Local and genomic alignment
- Except for user-initiated BLAT queries, all alignments are precomputed and stored.

Local alignment tools

UCSC Tools for local paired alignments are:

- BLAT / translated BLAT
 - Very fast
 - Lower sensitivity than BLASTZ/ Discontiguous MegaBLAST for xeno sequences
- BLASTZ - similar to BLASTn with different scoring

Local alignment (continued)

- Local paired alignments are used for:
 - mRNA /EST annotations (BLAT)
 - Seeds for genomic alignments (BLASTZ)
 - Initial clusterings for multiple alignments (BLASTZ)
- Multiple-species local alignments are performed with MULTIZ (an extension of BLASTZ).
- Cross species conservation of multiple alignments is scored with phastCons.



173

Genomic alignment tools

- Importance – chromosomal evolution, homolog identification
- Synteny – traditional method, gene order
- For paired genomic alignments, UCSC uses “chains and nets”
 - Not dependent on high quality gene annotations
 - Important for genomes with limited annotation
 - Support for segmental duplications and inversions



174

Chains

- A chain is an alignment assembled from smaller, local alignments that have been linked (“chained”) together.
- Chains tolerate larger gaps than conventional alignments.
- Chains are created with the axtChain program from BLASTZ alignments.



175

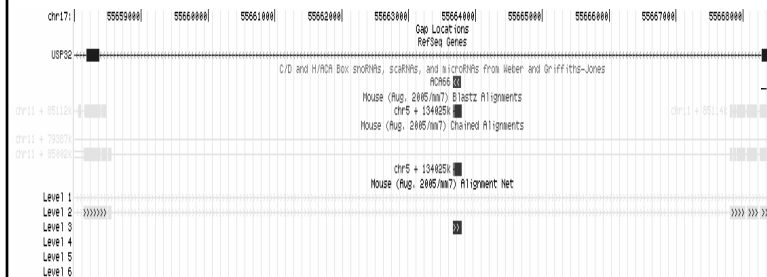
Nets

- Nets are generated by filtering chains such that each nucleotide is covered by at most one chain.
- chainNet program picks the “best” set of chains created by axtChain program to create the net track.
- Nets and chains are described in more detail in Kent PNAS v100 (2003).



176

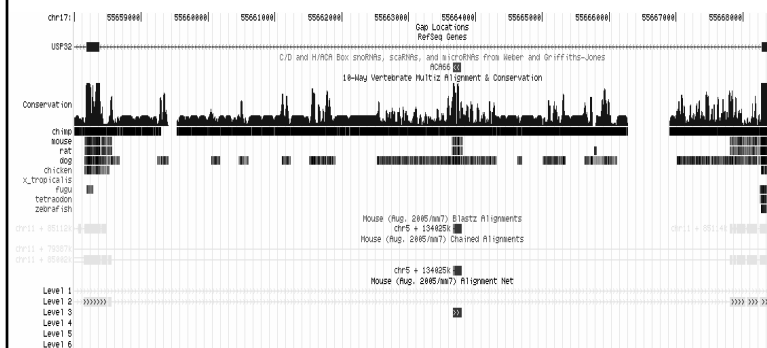
Chains and nets - example



Genomic Multiple Alignments

- Genomic multiple alignments are generated by the “threaded blockset aligner” (TBA) tool (Blanchette Genome Research 2004).
- TBA starts with local multiple alignment “seeds” generated by MULTIZ which are then linked together (“threaded”) to form longer alignments.
- TBA currently can not handle segmental duplications and inversions.

Chains, nets and MULTIZ alignments



Preview, Auxiliary and Development Browsers

Ensembl's Pre! and Genome Reviews browsers

- Ensembl's "Pre!" browsers provide previews, with limited annotations, of genomes being added to the Ensembl site.
- Ensembl's "Genome Reviews" Auxiliary Browser provides access to Ensembl's bacterial and archaeal browsers.
- Both Pre! And Genome Reviews browsers are accessible from the Ensembl home page.



181

The UCSC development browser and why NOT to use it

- <http://genome-test.cse.ucsc.edu> is the development site for the UCSC browser.
- Superficially, the site looks quite similar to the main UCSC genome browser.
- Moreover, in function, the site *seems* similar to the Ensembl Pre! Browser.
- However, the site is slow, can be confusing, is not well documented and is not supported for the external user.



182

The UCSC development browser and why NOT to use it (continued)

- Moreover, data and features have been much less tested, and data can disappear from the browser without any notice.
- Indeed, a leading UCSC developer has said that "everything on genome-test should be considered broken unless proven otherwise."



183

...except in very special circumstances

- However, the test site often covers species and assemblies not yet available on the main site, for example:
 - As of March 2006, UCSC main site covered 34 species. Test site had 80.
 - In particular, the test site had 40 bacterial and archaeal genomes. The main site had none.
 - Test site had ancestral "Boreoeutherian" genome.
 - Most recent human assembly on main site was from May 2004. Test site had the build from March 2006 (which has since been added to the main UCSC browser).



184

...except in very special circumstances (continued)

- genome-test has more annotation tracks.
- Although most are experimental and of little outside interest, occasionally interesting annotation tracks can be found on genome-test before they appear on the main browser.

The “bottom line” is that if your species is not available or you want a “sneak peek” at future annotations, genome-test may be useful. Just remember that the data may be transient, and the site is unsupported and has limited server resources so access it “gently”.



185

Part IV – Browser/Database Batch Querying



186

Batch querying overview

- Introduction / motivation
- UCSC table browser
- Custom tracks and frames
- Galaxy and direct SQL database querying
- A batch query example
- UCSC Database “gotchas”
- Batch querying on Ensembl



187

Why batch querying

- Interactive querying is difficult if you want to study *numerous* “interesting” genomic regions.
- Querying each region interactively is:
 - Tedious
 - Time-consuming
 - Error prone



188

Batch querying examples

- As an example, say you have found *one hundred* candidate polymorphisms and you want to know:
 - Are they in dbSNP?
 - Do they occur in any known ESTs?
 - Are the sites conserved in other vertebrates?
 - Are they near any "LINE" repeat sequences?

Of course you could repeat the procedures described in Part II one hundred times but that would get "old" very fast...



189

Other examples

- Other examples include characterizing multiple:
 - Non-coding RNA candidates
 - ultra-conserved regions
 - introns hosting snoRNA genes



190

Browsers and databases

- Each of the genome browsers is built on top of multiple relational databases.
- Typically data for each genome assembly are stored in a separate database and auxiliary data, e.g. gene ontology (GO) data, are stored in yet other databases.
- These databases may have hundreds of tables, many with millions of entries.



191

The UCSC Table Browser

- For batch queries, you need to query the browser databases.
- The conventional way of querying a relational database is via "Structured Query Language" (SQL).
- However with the Table Browser, you can query the database without using SQL.



192

Browser Database Formats

Nevertheless, even with the Table Browser, you need some understanding of the underlying track, table and file formats.

- Table formats describe how data is stored in the (relational) databases.
- Track formats describe how the data is presented on the browser.
- File formats describe how the data is stored in “flat files” in conventional computer files.
- Finally, for understanding the underlying the computer code (as we will do in the last part of this tutorial) you will need to learn about the “C” structures which hold the data in the source code.



193

Database formats and autoSQL

- Programs in the kent source tree make converting among table,file,track and “C” formats easier.
- In particular, the autoSQL program takes a general specification to automatically create C and SQL code to convert between C structures and SQL tables.
- autoSQL is described in detail at:

<http://www.linuxjournal.com/article/5949>



194

Main UCSC Data Formats

- GFF/GTF
- BED (Browser Extensible Data)
 - lists of genomic blocks
- PSL
 - RNA/DNA alignments
- .chain
 - pair-wise cross species alignments
- .maf
 - multiple genome alignments
- .wig
 - numerical data



195

Basic BED (in autoSQL format)

- *BED4 is the basic BED format and consists of:*

```
string chrom; "Reference sequence chromosome or scaffold"
uint chromStart; "Start position in chromosome"
uint chromEnd; "End position in chromosome"
string name; "Name of item"
```



196

BED6 format

string chrom; "Reference sequence chromosome or scaffold"
uint chromStart; "Start position in chromosome"
uint chromEnd; "End position in chromosome"
string name; "Name of item"
uint score; "Score from 0 - 1000"
Char[1] strand; "+" or "-"



197

BED12 format

- BED12 consists of the six fields of BED6 plus:*

uint thickStart; "Start of where display is thick (start codon)"
uint thickEnd; "End of where display is thick (stop codon)"
uint reserved; "used for RGB"
int blockCount; "Number of blocks"
int [blockCount] blockSizes; "Comma separated list of block sizes"
int [blockCount] chromStarts; "Start positions relative to chrom Start"



198

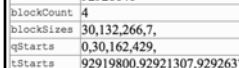
PSL in table format

Schema for Human mRNAs - Human mRNAs from GenBank

Database: hg17 Primary Table: all_mrna Row Count: 199,440

Description: Summary info about a putSpace alignment

field	example	SQL type	info	description
bin	1293	smallint(5) unsigned	range	Indexing field to speed chromosome range queries.
matches	432	int(10) unsigned	range	Number of bases that match that aren't repeats
mismatch	3	int(10) unsigned	range	Number of bases that don't match
repMatches	0	int(10) unsigned	range	Number of bases that match but are part of repeats
nCount	0	int(10) unsigned	range	Number of 'N' bases
qNumInsert	1	int(10) unsigned	range	Number of inserts in query
qBaseInsert	1	int(10) unsigned	range	Number of bases inserted in query
tNumInsert	2	int(10) unsigned	range	Number of inserts in target
tBaseInsert	6411	int(10) unsigned	range	Number of bases inserted in target
strand	+	char(2)	values	+ or - for strand. First character query, second target (optional)
qName	AL137623	varchar(255)	values	Query sequence name
qSize	624	int(10) unsigned	range	Query sequence size
qStart	0	int(10) unsigned	range	Alignment start position in query
qEnd	436	int(10) unsigned	range	Alignment end position in query
tName	chr9	varchar(255)	values	Target sequence name
tSize	138429268	int(10) unsigned	range	Target sequence size
tStart	92919800	int(10) unsigned	range	Alignment start position in target
tEnd	92926646	int(10) unsigned	range	Alignment end position in target
blockCount	4	int(10) unsigned	range	Number of blocks in alignment
blockSizes	30,132,266,7,	longblob		Size of each block
qStarts	0,30,162,429,	longblob		Start of each block in query.
tStarts	92919800,92921307,92926373,	longblob		Start of each block in target.



200

PSL format subtleties

- PSL format effectively represents an alignment as a set of "blocks" that are gapless in both sequences.
- However, PSL format does have some subtleties:
 - Much of the confusion in PSL stems from the fact that two coordinate systems are used:
 - Some fields (eg tStart and tEnd) are *always* measured in coordinates of the *positive* strand
 - Other fields (eg tStarts and qStarts) are measured in "strand" coordinates that start at the 3' end of the molecule if it is on the negative strand.



200

More PSL format subtleties

- Strand annotation is also a bit tricky:
 - For nucleotide (usually same-species) alignments, the PSL strand parameter is a single character ('+' or '-').
 - However, for *translated* alignments (e.g. xeno alignments), the PSL strand parameter is *two* characters (e.g. '+-') indicating whether the query and/or the target had to be reverse complemented in order to perform the protein alignment.
- Finally, insertions usually have different interpretations depending on whether they are in the "target" (chromosome) or the "query" (mRNA/EST)
 - "tBaseInsert" typically is the total length of the mRNA introns.
 - In contrast, "qBaseInsert" (if not = 0) represents insertions in the mRNA sequence relative to the genome (or artifacts).



201

MAF in table format

Schema for Conservation - Vertebrate Multiz Alignment & Conservation

field	example	SQL type
bin	585	smallint(5) unsigned
chrom	chr1	varchar(255)
chromStart	1024	int(10) unsigned
chromEnd	1259	int(10) unsigned
extFile	21208583	int(10) unsigned
offset	34	bigint(20)
score	0.970745	double

Note that maf tables really are *index* tables.

Actual mafs are stored in external files.

If you obtain the alignments via the Table Browser, you don't need to worry about this.

Later, we'll discuss how to download the actual maf files if you need them.



202

Actual maf alignment (partial)

```
##maf version=1
a score=3469.000000
s hg17.chr1      67108775 249 + 245522847 TTCCAAATCAAGGCTAGCTAT
s panTro1.chr1  65472086 249 + 229575298 TTCCAAATCAAGGCTGCTAT
s rheMac2.chr1  69709327 241 + 228252215 TTCCAAATCAAGGCTATCTA
s rn3.chr5      124059293 300 + 173106704 TTCCAAGTCAAGCTGTCACG
s mm7.chr4      102496518 305 + 155175443 TTACAAATCAAGCTGTCACG
s oryCun1.scaffold_210784 55797 281 + 212082- TACCAAGTCAAGCTTTTCC
s bosTau2.scaffold231 595046 351 + 47601633 AGGCTTTCTATTCTCTGCTCTTC
s canFam2.chr5  453905411 246430 TTGAATATTAAGGCTACCTATTATGTTT
s loxAfr1.scaffold_17369 51679 266659 TTCCAAATCATTACTAAGACTCTAGTTTC
s echTel1.scaffold_313458068 216 179910 TCCTAGACCAGCTATAATCTTTTATAA
```



203

Browser Table Descriptions

- Detailed descriptions of database tables at: <http://genome.ucsc.edu/goldenPath/gdbDescriptions.html>
- However, often you can obtain a sufficient table description from the Table Browser itself.



204

Table browser input form

Table browser input form

Explorer File Edit View Go Favorites Tools Window Help

Table Browser

Home Genomes Blat Tables Gene Sorter PCR FAQ Help

Table Browser

Use this program to get the data associated with a track in text format, to calculate intersections between tracks, and to retrieve DNA sequence covered by a track. See [Using the Table Browser](#) for a description of the controls in this form.

clade: genome: assembly:

group: track:

table: [describe table schema](#)

regions:

Schema for Known Genes - UCSB Known Genes (June, 05) Based on UniProt, RefSeq, and GenBank mRNA

Database: hg17 Primary Table: knownGene Row Count: 39,368

Description: Protein coding genes based on proteins from SWISS-PROT, TrEMBL, and TrEMBL-NEW and their corresponding mRNA

field	example	SQL type	info	description
name	BC073913	varchar (255)	values	Name of gene
chrom	chr1	varchar (255)	values	Reference sequence chromosome or scaffold
strand	-	char (1)	values + or -	strand
txStart	4268	int (10) unsigned	range	Transcription start position
txEnd	7438	int (10) unsigned	range	Transcription end position
cdsStart	6607	int (10) unsigned	range	Coding region start
cdsEnd	7173	int (10) unsigned	range	Coding region end
exonCount	6	int (10) unsigned	range	Number of exons
exonStarts	4268,4832,5658,6469,6720,7095	longblob		Exon start positions
exonEnds	4692,4901,5810,6628,6918,7438	longblob		Exon end positions
proteinID	Q6GMS0 HUMAN	varchar (40)	values	SWISS-PROT ID
alignID	G220323	varchar (8)	values	Unique identifier for each (known gene, alignment position) pair

sample Rows

name	chrom	strand	txStart	txEnd	cdsStart	cdsEnd	exonCount	exonStarts	exonEnds	proteinID	alignID
BC073913	chr1	+	4268	7438	6607	7173	6	4268,4832,5658,6469,6720,7095	4692,4901,5810,6628,6918,7438	Q6GMS0 HUMAN	G220323
AT360408	chr1	+	24416	29444	25080	25990	9	24416,25124,25183	25971,25974,25984,25994	Q9H087 HUMAN	Q12015
NM_001004484	chr1	+	59857	59857	59857	59857	1	59857	59857	Q9H087 HUMAN	Q12118
NM_001005221	chr1	+	60752	408460	407402	408460	1	60752	408460	Q9H087 HUMAN	Q12118
NM_001005221	chr1	+	600505	661367	600505	661367	1	600505	661367	Q9H087 HUMAN	Q12118
AC020727	chr1	+	76261	131007	767252	763232	3	76261,763065	763445,763575	Q9H087 HUMAN	Q12118
NM_024766	chr1	+	860146	962149	862142	962144	1	860146	962144	Q9H082 HUMAN	Q1
AC024207	chr1	+	96281	1396329	963773	966179	1	96281	966179	Q9H084 HUMAN	Q12092
AK020428	chr1	+	11288	122007	114832	1150476	10	11288,114832,114971,81666,817104,817652,818081,818175,818223,818264	114832,81662,81665,81668,81774,81801,818105,818160,818223	Q9H087 HUMAN	Q12092
AK020428	chr1	+	1147470	1151762	1151762	1151767	1	1147470,114666,91768,91762,91808,918771,919220	91808,91828,91831,91774,91801,91805,91806,918762	Q9H087 HUMAN	Q12092

206

Table Browser

Use this program to get the data associated with a track in text format, to calculate intersections between tracks, and to retrieve DNA sequence covered by a track. See [Using the Table Browser](#) for a description of the controls in this form.

clade: Other **genome:** S. cerevisiae **assembly:** Oct. 2003

group: Genes and Gene Prediction Tracks **track:** SGD Genes

table: sgdGene describe table schema

region: ☒ genome ☐ position chr13:746300-756100

identifiers (names/accessions):

filter:

intersection:

correlation:

output format: BED - browser extensible data

output file: (leave blank to keep output in browser)

file type returned: ☒ plain text ☐ gzip compressed

- You can restrict which table entries you retrieve by:
 - Filtering on values of specific fields in the table and/or in other tables to which it is linked.
 - Retrieving only the records in the intersection of two tables.
- You can also quickly calculate the amount of overlap or correlation between two tracks using the correlation output.

Filter on Fields from hg17.knownGene

name	<input type="text" value="does"/>	<input type="button" value="match"/>	<input type="text" value="*"/>	
chrom	<input type="text" value="does"/>	<input type="button" value="match"/>	<input type="text" value="*"/>	AND
strand	<input type="text" value="does"/>	<input type="button" value="match"/>	<input type="text" value="*"/>	AND
txStart	is <input type="text" value="ignored"/>	<input type="button" value="match"/>	<input type="text" value=""/>	AND
txEnd	is <input type="text" value="ignored"/>	<input type="button" value="match"/>	<input type="text" value=""/>	AND
cdsStart	is <input type="text" value="ignored"/>	<input type="button" value="match"/>	<input type="text" value=""/>	AND
cdsEnd	is <input type="text" value="ignored"/>	<input type="button" value="match"/>	<input type="text" value=""/>	AND
exonCount	is <input type="text" value="ignored"/>	<input type="button" value="match"/>	<input type="text" value=""/>	AND
exonStarts	<input type="text" value="does"/>	<input type="button" value="match"/>	<input type="text" value="*"/>	
exonEnds	<input type="text" value="does"/>	<input type="button" value="match"/>	<input type="text" value="*"/>	
proteinID	<input type="text" value="does"/>	<input type="button" value="match"/>	<input type="text" value="*"/>	AND
alignID	<input type="text" value="does"/>	<input type="button" value="match"/>	<input type="text" value="*"/>	AND

Table Intersect Menu

Intersect with Known Genes

Select a group, track and table to intersect with:
group: [Variation and Repeats] track: [SNPs]
table: [SNPs (snp)]

These combinations will maintain the gene/alignment structure (if any) of Known Genes:

☒ All Known Genes records that have any overlap with SNPs
☐ All Known Genes records that have no overlap with SNPs
☐ All Known Genes records that have at least 80 % overlap with SNPs
☐ All Known Genes records that have at most 80 % overlap with SNPs

These combinations will discard the gene/alignment structure (if any) of Known Genes and produce a simple list of position ranges.

☐ Base-pair-wise intersection (AND) of Known Genes and SNPs
☐ Base-pair-wise union (OR) of Known Genes and SNPs

Check the following boxes to complement one or both tables. To complement a table means to include a row in the intersection if it is *not* included in the table.
☐ Complement Known Genes before intersection/union
☐ Complement SNPs before intersection/union

[submit] [cancel]

I/MB 2006

Fortaleza, Brazil

August 4-9, 2006

210

Specifying table query output format

[illegible]

Table Browser Fasta Output

- Capable of extracting sequence data from multiple genomic region
- Can extract only intron or exons or
- UTRs or upstream / downstream regions

Gene fasta output options

Home Genomes Blat Tables Gene Sorter PCR FAQ Help

Known Genes Genomic Sequence

Sequence Retrieval Region Options:

- ☐ Promoter/Upstream by 1000 bases
- ☐ 5' UTR Exons
- ☐ CDS Exons
- ☐ 3' UTR Exons
- ☐ Introns
- ☐ Downstream by 1000 bases
- ☐ One FASTA record per gene.
- ☐ One FASTA record per region (exon, intron, etc.) with 0 extra bases upstream (5') and 0 extra downstream (3')
- ☐ Split UTR and CDS parts of an exon into separate FASTA records

Note: if a feature is close to the beginning or end of a chromosome and upstream/downstream bases are added, they may be truncated in order to avoid extending past the edge of the chromosome.

Sequence Formatting Options:

- ☐ Exons in upper case, everything else in lower case.
- ☐ CDS in upper case, UTR in lower case.
- ☐ All upper case.
- ☐ All lower case.
- ☐ Mask repeats: ☐ to lower case ☐ to N

get sequence cancel

IUMB 2006
Fortaleza, Brazil
August 6-10, 2006

213

Custom Tracks

- Custom tracks are essentially BED, PSL or GTF files with formatting lines so they can be displayed on the browser.
- A custom track file can contain multiple tracks, which may be in different formats.
- Custom tracks are useful for:
 - Display of regions of interest on the browser.
 - Sharing custom data with others.
 - Input of multiple, arbitrary regions for annotation by the Table Browser.
- Custom tracks can be made by the Table Browser, or you can make them easily yourself.



214

Selecting custom track output

Home Genomes Blat Tables Gene Sorter PCR FAQ Help

Table Browser

Use this program to get the data associated with a track in text format, to calculate intersections between tracks, and to retrieve DNA sequence covered by a track. See Using the Table Browser for a description of the controls in this form.

clade: Vertebrate genome: Human assembly: May 2004

group: Genes and Gene Prediction Tracks track: Known Genes

table: knownGene describe table schema

region: genome ENCODE position chr7:126665857-128301056 lookup

identifiers (names/accessions): paste list upload list

filter: create

intersection: create

correlation: create

output format: all fields from selected table selected fields from primary and related tables sequence CTR - genome transfer format only data leave blank to keep output in browser)

output file: file type return custom track compressed

get output summary/statistics

To reset all user cart settings (including custom tracks), click here.

IUMB 2006
Fortaleza, Brazil
August 6-10, 2006

215

Sending custom track to browser

Output knownGene as Custom Track

Custom track header:

name= tb_knownGene

description= table browser query on knownGene

visibility= pack

url=

Create one BED record per:

☐ Whole Gene

☐ Upstream by 200 bases

☒ Exons plus 0 bases at each end

☐ Introns plus 0 bases at each end

☐ 5' UTR Exons

☐ Coding Exons

☐ 3' UTR Exons

☐ Downstream by 200 bases

Note: if a feature is close to the beginning or end of a chromosome and upstream/downstream bases are added, they may be truncated in order to avoid extending past the edge of the chromosome.

get custom track in table browser get custom track in file

get custom track in genome browser cancel

Adding a custom track

The UCSC Genome Browser was created by the [Genome Bioinformatics Group of UC Santa Cruz](#).
Software Copyright (c) The Regents of the University of California. All rights reserved.

clade genome assembly position or search term image width
 Vertebrate Human May 2004 chr17:7,512,463-7,531,642 1000 submit

[Click here to reset the browser user interface settings to their defaults.](#)

[add your own custom tracks](#) [configure tracks and display](#) [clear position](#)

IMB 2006
Fortaleza, Brazil
August 6-10, 2006

217

Adding a custom track (II)

Add Your Own Custom Track

Display your own annotation tracks in the browser using the [procedure described here](#). Annotations may be uploaded from files or pasted into the text box below. You can also paste a URL or a list of URLs into the large text box that refer to files in one of the supported formats.

[Click here to view a collection of custom annotation tracks submitted by Genome Browser users.](#)

Annotation File:

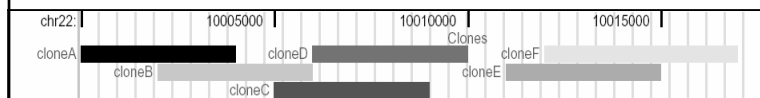
browser position chr1:148374643-148374642
 browser hide all
 browser track knownGene
 track type=usggle_0 name=cat250_gval description="average p-values"
 color=0,0,255 visibility=full
 chr1 148374674 148375188 0.0379727407986273
 chr1 148375824 148376426 0.0212499308118058
 chr1 148376485 148377081 0.435388573890358
 chr1 148378442 148379421 6
 chr1 148383005 148383262 0.532038552475468
 chr1 148384344 148384605 0.234713017233213
 chr1 148385382 148385602 1.70151808748632
 chr1 148386675 148387044 1.405914893905219
 chr1 148857332 148857713 0.841758359855708

IMB 2006
Fortaleza, Brazil
August 6-10, 2006

218

Custom track example

```
browser position chr22:100000000-100200000
browser hide all
track name=clones description="Clones" visibility=3
color=0,128,0 useScore=1
chr22 10000000 10004000 cloneA 960
chr22 10002000 10006000 cloneB 200
chr22 10005000 10009000 cloneC 700
chr22 10006000 10010000 cloneD 600
chr22 10011000 10015000 cloneE 300
chr22 10012000 10017000 cloneF 100
```



IMB 2006
Fortaleza, Brazil
August 6-10, 2006

219

Limitations of the table browser

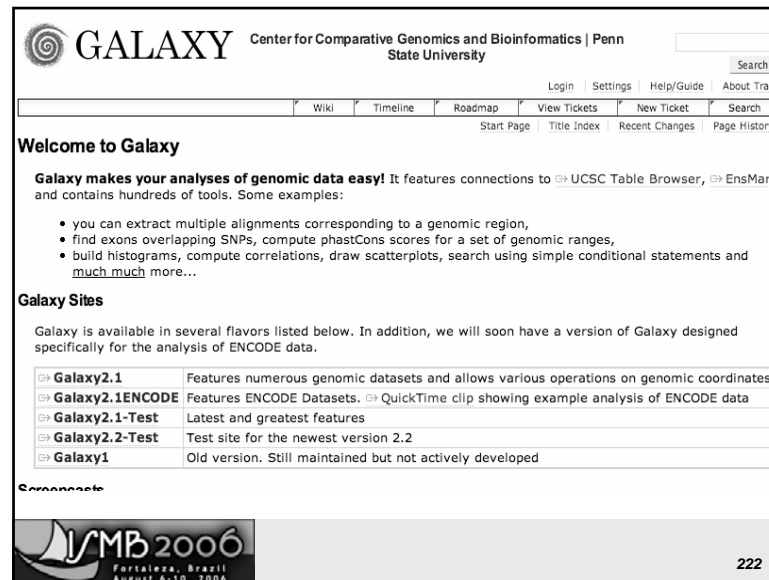
- Can be difficult to create more complex queries.
- With hundreds of tables, finding the one(s) you want can be confusing.
- Getting intersections or unions of genomic regions is often a multi-step process and can be tedious or error prone.
- May be slower than direct SQL query.
- Not designed for fully automated operation.

IMB 2006
Fortaleza, Brazil
August 6-10, 2006

220

The Galaxy Website

- Galaxy website: <http://g2.bx.psu.edu>
- Galaxy objective: Provide sequence and data manipulation tools (a la SRS or the UCSD Biology Workbench) that are capable of being applied to genomic data.
- The intent is to provide an easy interface to numerous analysis tools with varied output formats that can work on data from multiple browsers / databases.



The screenshot shows the Galaxy website homepage. At the top is the GALAXY logo and the text "Center for Comparative Genomics and Bioinformatics | Penn State University". Below this is a navigation bar with links: Login, Settings, Help/Guide, About Trac, and a Search box. A secondary navigation bar includes links: Wiki, Timeline, Roadmap, View Tickets, New Ticket, and another Search box. The main content area starts with "Welcome to Galaxy" and a paragraph stating "Galaxy makes your analyses of genomic data easy! It features connections to UCSC Table Browser, EnsMart, and contains hundreds of tools. Some examples:". This is followed by a bulleted list of features: extracting multiple alignments, finding exons overlapping SNPs, computing phastCons scores, building histograms, computing correlations, drawing scatterplots, and searching using simple conditional statements. Below this is a section titled "Galaxy Sites" with a paragraph stating "Galaxy is available in several flavors listed below. In addition, we will soon have a version of Galaxy designed specifically for the analysis of ENCODE data." This is followed by a table listing different Galaxy versions and their features.

Galaxy2.1	Features numerous genomic datasets and allows various operations on genomic coordinates
Galaxy2.1ENCODE	Features ENCODE Datasets. QuickTime clip showing example analysis of ENCODE data
Galaxy2.1-Test	Latest and greatest features
Galaxy2.2-Test	Test site for the newest version 2.2
Galaxy1	Old version. Still maintained but not actively developed

Below the table is a "Connect" section.

Galaxy - current status

- Galaxy is a new site, still a "work in progress".
- So far, supports UCSC Table Browser, EBI EnSmart and NHGRI EncodeDB.
- As yet, few sequence manipulation tools are available, e.g.
 - GC%
 - Ka/ Ks calculations
- Galaxy does already provide an effective query and result "history" system which makes the Table Browser interface more "user friendly".

Direct SQL queries of the underlying databases

- If you are familiar with SQL, direct queries can be:
 - much more flexible
 - and sometimes easier or faster than using the Table Browser or Galaxy

SQL code can access the UCSC databases via any of:

- Table browser interface
- The public UCSC browser database
- Your own mirror site

Table Browser SQL Interface

Filter on Fields from hg17.knownGene

name	does	match	*	
chrom	does	match	*	AND
strand	does	match	*	AND
txStart	is	ignored		AND
txEnd	is	ignored		AND
cdsStart	is	ignored		AND
cdsEnd	is	ignored		AND
exonCount	is	ignored		AND
exonStarts	does	match	*	
exonEnds	does	match	*	
proteinID	does	match	*	AND
alignID	does	match	*	AND

AND Free-form query:

submit cancel

The public UCSC genome database

- UCSC has recently made a mirror of its genomic databases available for (limited) direct SQL queries.
- Access information:
 - host=genome-mysql.cse.ucsc.edu
 - user=genome
 - No password

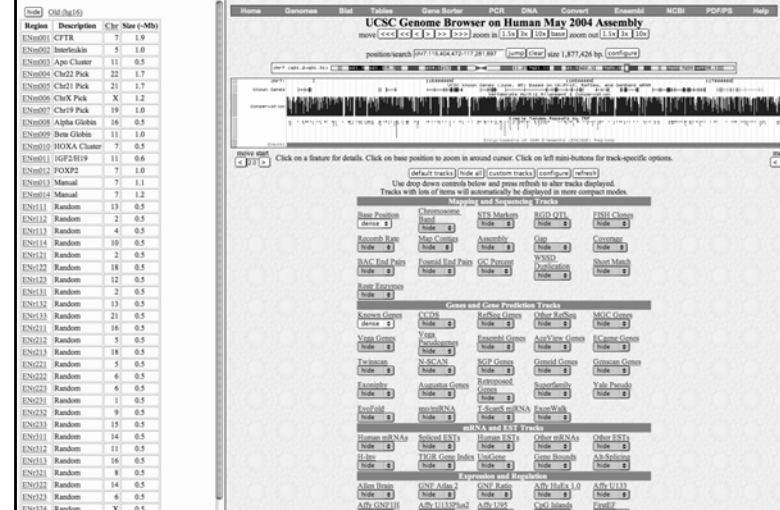
The public UCSC genome database (II)

- *More details on the public database can be found at:*
<http://genome.ucsc.edu/FAQ/FAQdownloads/download29#download29>
- *In particular the instructions note:*
 - “Avoid heavy queries that may impact the server performance.
 - “If you plan a query that may be excessive, contact UCSC first to avoid the possibility of blocked access.
 - “Bot access and excessive program-driven use are not permitted.”

Custom frames

- “Custom frames” are a useful tool for navigating among a set of regions of interest.
- A list of all the regions is shown on one side of the screen, with a standard genome browser image in the other.
- Writing code to convert a bed file to a custom frame is straightforward – or you can use bedToFrame from the kent code source tree (described later).

ENCODE custom frame



Batch query example

- Recall our example, where we have one hundred candidate polymorphisms and you want to know:
 - Are they in dbSNP?
 - Do they occur in any known ESTs?
 - Are the sites conserved in other vertebrates?
 - Are they near any “LINE” repeat sequences?

Batch query example in the Table Browser

To answer these questions with the Table Browser we could:

- Run BLAT to find the polymorphism locations in the genome.
- Convert BLAT psl output to custom track (manually or with a simple script).
- Intersect the custom track with tables in the Table Browser.

Batch query example continued

Specifically in the Table Browser we would need to:

- Intersect the custom track with SNP track
- Intersect the custom track with EST track, outputting the sequence and writing a program (or manually checking) to see if the mutated bases occur.
- Intersect the track with the multiz17way table to determine if the site is conserved.
- Extend the range of the custom track and intersect the modified track with the repeat-element table (rmsk), filtering on LINE elements.



233

Browser “gotchas”

- Browser “gotchas” are not bugs but rather situations where the system interface works in an unexpected manner (at least unexpected, to me!)
- Some gotchas are the result of intrinsic difficulties in describing genomic sequence data.
- Other gotchas are specific to the conventions used in the UCSC browser and database systems.



234

Subtleties inherent to genomic data representation

- A gap in an mRNA alignment to the genome may not be an intron, but rather an insertion in the genome sequence or a deletion in the mRNA.
- A sequence difference between an mRNA and the genome may not be a polymorphism or post-transcriptional editing, but rather a sequencing artifact (especially if the mRNA record is old).
- The number of blocks in an alignment between an mRNA and the genome may not be the same as the number of “blocks” (i.e. exons) in the the gene that is predicted to be represented by that mRNA.



235

Gotchas specific to the UCSC data representation

- UCSC uses a [1,n) numbering system.
 - That is, the region 1000-2000 includes base 1000 but not base 2000.
- Database tables use a nucleotide numbering system that is 0-based. But the browser display uses genomic numbering that is 1-based.
- Many tables can be accessed in Table Browser only by using “all tables”.



236

UCSC data representation gotchas (II)

- Multiple alignments (MAF records) are stored in separate files.
- Block start surprises:
 - In BED the block start is relative to the start location of the *bed* (chromStart)
 - in genePred, the exon start/end are *absolute* chromosome locations.
- Many tables have an extra “bin” field, used for fast indexing. This field must be stripped away (e.g. using the Unix “cut” utility) before the data can be input to C-code from the kent source tree.



237

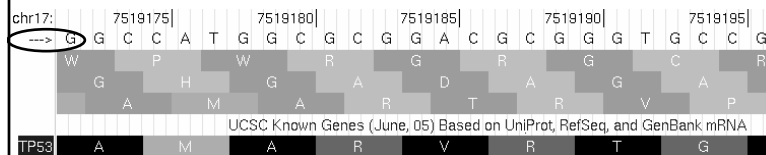
UCSC data representation gotchas (III)

- Translated BLAT has difficulty with intron-exon boundaries in xeno alignments, which are sometimes less accurate than those found by other approaches.
- As described earlier, interpreting PSL negative strand data can be tricky. In fact, even on the browser, negative strand genes may appear incorrect if you are not careful (example on next page).
- These “gotcha” examples are just those that have personally tripped me up. To find more, look at the UCSC FAQ pages: <http://www.genome.ucsc/FAQ/>



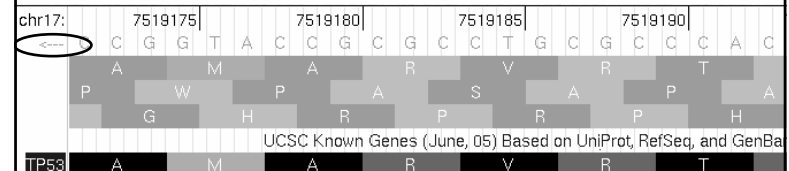
238

Apparent inconsistency with negative strand genes



239

Apparent inconsistency resolved



240

Custom tracks in Ensembl

- In Ensembl, custom tracks are implemented using either a variant of UCSC's custom track system or using the "Distributed Annotation System" (DAS).
- However using the UCSC syntax on Ensembl requires access to a local webserver.
- The DAS approach does not require a server (but does require an understanding of DAS syntax).



241

BioMart

- BioMart - the Ensembl "Table browser"
- Similar to the Table Browser and Galaxy tools.
- Previous version was called EnsMart.
- Fewer tables can be accessed with BioMart than with UCSC Table Browser. In particular, non-gene oriented queries may be difficult.
- However, the user interface is simpler.
- Tight interface with Bioconductor project for annotation of microarray genes.



242

DAS

- DAS stands for "Distributed Annotation System"
- DAS consists of:
 - a single "sequence server"
 - Multiple "annotation servers"
 - DAS client software to integrate results



243

DAS - pros and cons

- DAS is intended to be scalable in the face of hundreds of annotation tracks.
- Maintaining tracks is not responsibility of single group (as with UCSC's or NCBI's browsers).
- However, response time may be slow since data must be retrieved from multiple servers.
- Also. It may be difficult to keep track of what's available, so there is a DAS "registry" where user can find annotation availability:
<http://das.sanger.ac.uk/registry>



244

Part V – Automated Querying Procedures

Automated querying overview

- Introduction / motivation
- What you'll need to get started
- Choice of languages and databases
- Using the kent code tree to access the UCSC database
- Example

Introduction / motivation

- Using the table browser is still a partly interactive process.
- Consequently, when performing multiple, large scale queries this approach becomes time-consuming and error prone.
- More complex data analyses usually require developing data manipulation software, anyway,
- It is usually more efficient to develop this software using tools and routines already developed by the genome-browser teams.

An example

- Let's imagine that, instead of experimental data, we have a computer algorithm to *predict* candidate disease polymorphisms and we want to know:
 - Are they in dbSNP?
 - Do they occur in any known ESTs?
 - Are the sites conserved in other vertebrates?
 - Are they near any "LINE" repeat sequences?
- In addition, we want to test our algorithm with many different parameters and options.
- We will NOT want to interactively perform all the Table Browser table-intersections every time we modify a parameter.

More examples

Although the previous example is a bit contrived, it is not unlike more realistic ones, e.g.:

- Characterizing the introns of genes that host snoRNAs and determining whether the host genes of homologous snoRNAs are homologous themselves (Schattner et. al, RNA, 2006).
- Characterizing regions of extreme codon conservation among mammalian genes in terms of SNPs, conserved alternative splicing, exon splicing enhance motifs, etc. (Schattner and Diekhans, NAR, 2006).



249

What you'll need to get started

- General programming skills
- Database querying skills
- Access to the data files used by one of the browsers.



250

Design Choices

- Before developing data querying software, one needs to choose which language, databases and data-retrieval strategy to use.
- Criteria include:
 - Ease of data access and/or installation requirements.
 - Your experience with relevant language and database.
 - Capabilities & features of language, database and associated tools and software.



251

Main Supported Languages

Although, in principle, any of the databases can be accessed by any computer language the best supported interfaces (APIs) are:

- UCSC: Main API uses C
- Ensembl: Primary APIs use Perl and Java
- NCBI: API in C/C++ via the NCBI toolkit



252

Why I use the UCSC databases

- Library routines in the kent code base do most of the work for me.
- The databases are comprehensive.
- C is fast.



253

Data access strategies

- Downloading one or more database tables or files.
- Remote login to a public mirror.
- Mirroring all or part of an entire database.



254

No bots!

- In principle, one can also access a genome database using a “bot”, or web robot.
- A bot is a computer program which generates code to “look like” a user to an interactive website (such as a genome browser site.)
- However, interactive sites are typically not designed to handle programmed “hits”. Consequently, you will be denied site access if you use a bot.
- So, don’t do it. The methods described here are easier ways of making automated database queries than using bots, anyway.



255

Option 1: Downloading individual database tables or files

- Generally simplest approach for limited automated querying.
- You can download data with the Table Browser or directly from:
<http://hgdownload.cse.ucsc.edu/goldenPath/hg18/database/>
(substitute the name of the database you are interested in for “hg18”)
- You can download both the table data files and files with SQL code to load the data into a local MySQL database.



256

Advantages of downloading individual files and tables

- Simplest method.
- Limited disk space required.
- Can be quite fast (especially if you take advantage of the kent code routines).
- Don't need to set up SQL database.



257

Disadvantages of downloading individual files and tables

- Limited to accessing small number of tables and files.
- Need to repeat procedure if you need additional tables.
- Need to separately download sequence and genomic alignment files and understand how to access them (if you need this data).



258

Finding external UCSC database files

For build hg18:

- Genome sequences are located at:
<http://hgdownload.cse.ucsc.edu/goldenPath/hg18/chromosomes/>
- Multiple genome alignments are located at:
<http://hgdownload.cse.ucsc.edu/goldenPath/hg18/multiz17way/>
- mRNA, EST, refseq and other large sequence data are at:
<http://hgdownload.cse.ucsc.edu/goldenPath/hg18/bigZips/>

Offsets to individual sequences within these files are found in the database tables:

- multiz17way
- gbSeq



259

Loading Standard Formats

- If you do choose to build a small local MySQL database for your downloaded files, there are command line tools in `src/hg/makeDb` to load them:
 - `ldHgGene` (GTF), `hgLoadPsl`, `hgLoadBed`, `hgLoadChain`, `hgLoadMaf`, `hgGenericMicroarray`
- This directory also has source for 60 other database loaders for more specialized situations.
- Typical database loader is about 200 lines, using SQL and text parsing routines in `src/lib`.



260

Option 2: Remote login to the UCSC or Ensembl databases

- UCSC site:
genome-mysql.cse.ucsc.edu
- Ensembl site:
ensembl.db.ensembl.org



261

Advantages / disadvantages of remote login to a public mirror

- Does not require local disk space allocation.
- No installation or database maintenance.
- Kent source code routines not requiring auxiliary sequence files run without any modifications.
- The biggest disadvantage is that a public mirror is a shared resource that may be slow, can't be modified and shouldn't be overused by a single user.



262

Option 3: Setting up a mirror database

- A complete install of the UCSC genome database requires 1.2+ terabytes, but you do **not** need to do a complete install for automated database querying.
- You do not need Apache or any CGI or HTML files.
- You only need to download the species databases you actually want to use.
 - Human genome database requires ~ 200 - 250 GB
 - Most other species databases are significantly smaller.



263

Advantages of setting up a mirror database

- The database will not be a shared resource.
- You can run it as heavily as you like.
- Performance doesn't depend on usage by others.
- Accessing sequence and alignment datafiles is significantly easier.
- You can be modify or customize the database as desired.



264

Overview of mirror database installation

- Install mysql if not already present.
- Download sequence and annotation data for genomes of interest.
- Load annotation data into local mysql database.
- Download and compile kent source code.
- Detailed instructions at:
 - <http://genome.ucsc.edu/admin/mirror.html>
- Note that these procedures may not work on Windows machines (cygwin has worked).
- Unix, Linux, OS X are all OK



265

Using the kent code base

...or why I no longer miss Perl and Bioperl!

Whatever data access method you use, you will want to take advantage of the kent source code base...

- Code is clearly written, extensively tested and fast.
- Code is open source so you can learn from it and modify it for your own use.
- Code is free for academic, government and personal use (core routines are even free for commercial use).
- Even if you run Windows or never plan to set up a database mirror, taking advantage of the kent code libraries can save you hundreds of hours of time.



266

Using the kent code base (continued)

- Many important utilities are usable “right out of the box”.
- If you don’t immediately see what you want, using “grep”, “find” or “tags” on the source tree will often find it.
- Built-in library functions provide almost any sequence and data manipulation capability you might want.
- Plenty of code examples illustrate exactly how to use the library effectively.



267

Built-in utilities

There are over 100 utility programs available for tasks such as:

- Sorting, splitting, merging, counting and getting lengths of fasta sequences.
- Record parsing and data conversion utilities for handling genbank, fasta, nib, blast and other records.
- Programs for sequence alignment, motif searching, hidden Markov models, etc.
- Programs for automatically generating SQL or XML code from user specifications (AutoSQL and AutoXML).



268

Properties of the kent source code

- In addition to stand-alone programs, there are CGI-based programs to perform all the sequence and data manipulations performed by the browser.
- In general, if the browser performs some data manipulation, with a little detective work, you can find the code to insert in your program.
- Sometimes the appropriate program can be identified by simply looking after the “cgi-bin” in the web address as in genome.ucsc.edu/cgi-bin/hgTracks



269

Using code from browser routines

- To understand the functioning of browser cgi-programs, it can be helpful to know what arguments they are being passed.
- These CGI input arguments are stored in the “CART”.
- Current CART arguments can be examined by running:
 - <http://genome.cse.ucsc.edu/cgi-bin/cartDump>



270

cartDump screenshot

```
taJD hide
taJDsnp hide
targetScanS hide
textSize medium
tfbsConsSites hide
tigrGeneIndex hide
trackControlsOnMain 1
twinscan hide
uniGene_2 hide
vegaGene hide
vegaPseudoGene hide
wgRna hide
xenoEst hide
xenoMrna hide
xenoRefGene hide
```

alter variable named: new value
Put n/a in for the new value to clear a variable.



271

kent library functions

There are many extremely useful routines including:

- Memory allocation and error handling
- String and array manipulation
- Data structures for singly and doubly linked lists, balanced trees, directed graphs etc.
- Code for very fast hashing, indexing and data retrieval using “binkeeper” and related programs
- Powerful code “wrappers” for SQL, CGI and HTML code generation
- Sequence manipulation routines including:
 - Reverse complementation
 - Codon and amino acid lookup
 - Sequence translation



272

UCSC Source Important Dirs

src -

- inc - interface to general purpose routines.
- lib - implementation of general purpose routines. Freeware.
- hg - genome project specific code
 - inc - interface to shared genome code
 - lib - implementation of shared genome code
 - hgTracks - genome browser
 - hgTables - table browser
 - hgNear - for gene sorter
 - makeDb - database building
 - hgLoadBed - load bed files
 - makeHg18.doc - how to build latest human genome database
 - schema - contains all.joiner that describes table relationships
- jkOwnLib - BLAT and other stuff Jim Kent personally owns
- utils - stand alone utility programs



273

Databases and program objects

- The kent library and database code is largely object oriented.
- Browser tracks and tables often have associated C structures defined in a .h “include” file.
- Typically the .h file will describe the functions that can be performed on the structure.
- Associated “.c” files describe their implementation.
- Much of this software is automatically generated by the autoSql program.



274

Gene-prediction C- Structure (slightly simplified)

```
struct genePred    /* A gene prediction */
{
    struct genePred *next;    /* Next in singly linked list. */
    char *name;              /* Name of loci, transcript, mRNA, etc */
    char *chrom;             /* Chromosome name */
    char strand[2];          /* + or - for strand */
    unsigned txStart;        /* Transcription start position */
    unsigned txEnd;          /* Transcription end position */
    unsigned cdsStart;       /* Coding region start */
    unsigned cdsEnd;         /* Coding region end */
    unsigned exonCount;      /* Number of exons */
    unsigned *exonStarts;    /* Exon start positions */
    unsigned *exonEnds;      /* Exon end positions */
}
```



275

A simple C program illustrating automated database querying

- Program objective: to determine whether specific introns (e.g. those containing snoRNAs) have different median length than other introns of the same genes.
- Approach:
 - Read in list of snoRNA coordinates.
 - Extract genes “hosting” these snoRNAs.
 - Compute lengths of host introns and (for comparison) other introns of the host genes.

(For more realistic code examples of automated UCSC database querying see:
<http://nar.oxfordjournals.org/cgi/data/34/6/1700/DC1/1X>)



276

```

int main(int argc, char *argv[])
{
    char *db = argv[1]; char *geneTable = argv[2];
    char *bedFile = argv[3]; char *method = argv[4];
    if (argc != 5) usage();
    struct sqlConnection *conn = NULL;
    struct hash *gpHash = NULL;
    if (sameWord(method, "file"))
        gpHash = readGpToBinKeeper(db, geneTable);
    else
    {
        conn = sameWord(method, "public") ?
            getHgdbtestConn(db) : sqlConnect(db);
    }
    processBedFile(bedFile, conn, geneTable, gpHash);
    slFreeList(&overlapList); slFreeList(&otherList);
    sqlDisconnect(&conn); binKeeperGpHashFree(&gpHash);
    return 0;
}

```

```

struct slDouble *overlapList = NULL;
struct slDouble *otherList = NULL;

/*****
struct sqlConnection *getHgdbtestConn(char *db)
/* Read .hg.conf and return connection. */
{
    char *host = "genome-mysql.cse.ucsc.edu";
    char *user = "genome";
    char *password = NULL;
    hSetDbConnect(host, db, user, password);
    return sqlConnectRemote(host, user, password,
        db);
}

```

```

struct hash *readGpToBinKeeper(char *gpFileName)
{
    #define MAX_CHROM_SIZE 400000000
    struct binKeeper *bk; struct genePred *gp;
    struct lineFile *pf = lineFileOpen(gpFileName, TRUE);
    struct hash *hash = newHash(0);
    char *row[21]; int genePredLineCtMin = 10;
    while (lineFileNextRow(pf, row, genePredLineCtMin))
    {
        gp = genePredLoad(row);
        if (hashLookup(hash, gp->chrom) == NULL)
        {
            bk = binKeeperNew(0, MAX_CHROM_SIZE);
            hashAdd(hash, gp->chrom, bk);
        }
        bk = hashMustFindVal(hash, gp->chrom);
        binKeeperAdd(bk, gp->txStart, gp->txEnd, gp);
    }
    lineFileClose(pf);
    return hash;
}

```

```

void processBedFile(char *bedFile, struct
    sqlConnection *conn, char *geneTable, struct hash
    *gpHash)
/* Read file and process */
{
    struct bed *bedList=NULL, *bed=NULL;
    bedList = bedLoadAll(bedFile);
    for(bed = bedList; bed != NULL; bed = bed->next)
    {
        doOneBed(bed, conn, geneTable, gpHash);
    }
    printf("Median value of lengths of overlapping
        introns = %f\n", slDoubleMedian(overlapList));
    printf("Median value of lengths of other introns =
        %f\n", slDoubleMedian(otherList));
    bedFreeList(&bedList);
}

```

```

void doOneBed(struct bed *bed, struct sqlConnection *conn,
             char *geneTable, struct hash *gpHash)
{
    int bStart = bed->chromStart;
    int bEnd = bed->chromEnd;
    struct genePred *gp = NULL;
    if (gpHash == NULL)
        gp = genePredReaderLoadRangeQuery(conn, geneTable,
                                           bed->chrom, bStart, bEnd, NULL);
    else
        gp = bkToGenePreds(gpHash, bed->chrom, bStart, bEnd);
    slSort(&gp, genePredLongestCmp);
    if (gp == NULL)
    {
        errAbort("No gene found in %s overlapping %s:%d-%d\n",
                geneTable, bed->chrom, bStart, bEnd);
    }
    doOneGene(gp, bStart, bEnd);
    if (gpHash == NULL) genePredFreeList(&gp);
}

```



281

Sort comparison function

```

int genePredLongestCmp(const void *va, const void *vb)
/* Compare to sort based sizes of txEnd - txStart,
largest first. */
{
    const struct genePred *a = *((struct genePred **)va);
    const struct genePred *b = *((struct genePred **)vb);
    int lengthA = a->txEnd - a->txStart;
    int lengthB = b->txEnd - b->txStart;
    int dif = lengthB - lengthA;
    return dif;
}

```



282

```

void doOneGene(struct genePred *gp, int qStart, int qEnd)
/* get intron statistics for longest gene in range */
{
    int i, intronStart, intronEnd;
    for (i=0; i< gp->exonCount; ++i)
    {
        intronStart = gp->exonEnds[i];
        intronEnd = gp->exonStarts[i + 1];
        double intronLength = (double) (intronEnd -
                                         intronStart);
        struct slDouble *slIntronLength =
            slDoubleNew(intronLength);
        if (positiveRangeIntersection(qStart, qEnd,
                                       intronStart, intronEnd))
            slSafeAddHead(&overlapList, slIntronLength);
        else
            slSafeAddHead(&otherList, slIntronLength);
    }
}

```



283

Downsides of automated database querying

Although automated querying is very helpful when requiring multiple complex queries, it does have disadvantages:

- Installation is a (one time) pain and requires significant disk space.
- Conversely queries to the public database are restricted in scope.
- Writing and testing code takes time.
- Data sets stored in external files (e.g. maf file alignments) may be easier to obtain through the Table Browser.



284

Finding more information

- Articles in the literature: see handout
- Openhelix - excellent (introductory) on-line tutorials at: www.openhelix.com
- Tutorials / userguides / FAQs at the browser websites
 - www.ensembl.org/Docs
 - genome.cse.ucsc.edu/goldenPath/help/hgTracksHelp.html
 - www.ncbi.nlm.nih.gov/mapview/static/MapViewHelp.html



285

Some UCSC Resources

- Home page genome.ucsc.edu
- Pre-release genome-test.cse.ucsc.edu
- Mailing list genome-www@soe.ucsc.edu
- Download hgdownload.cse.ucsc.edu
- Src CVS genome.ucsc.edu/admin/cvs.html
- MySQL DB genome-mysql.cse.ucsc.edu
- List of progs:
 - <http://genome-test.cse.ucsc.edu/eng/useMessageIndex.html>



286

Acknowledgements

UCSC

Mark Diekhans
Hiram Clawson
Fan Hsu
Bob Kuhn
Daryl Thomas

EBI

Ewan Birney
Xose Fernandez

NCBI

Deanna Church
David Wheeler

*And the three browser teams for creating such
outstanding and useful tools in the first place!*



287