# GS01 0163
# Analysis of Microarray Data

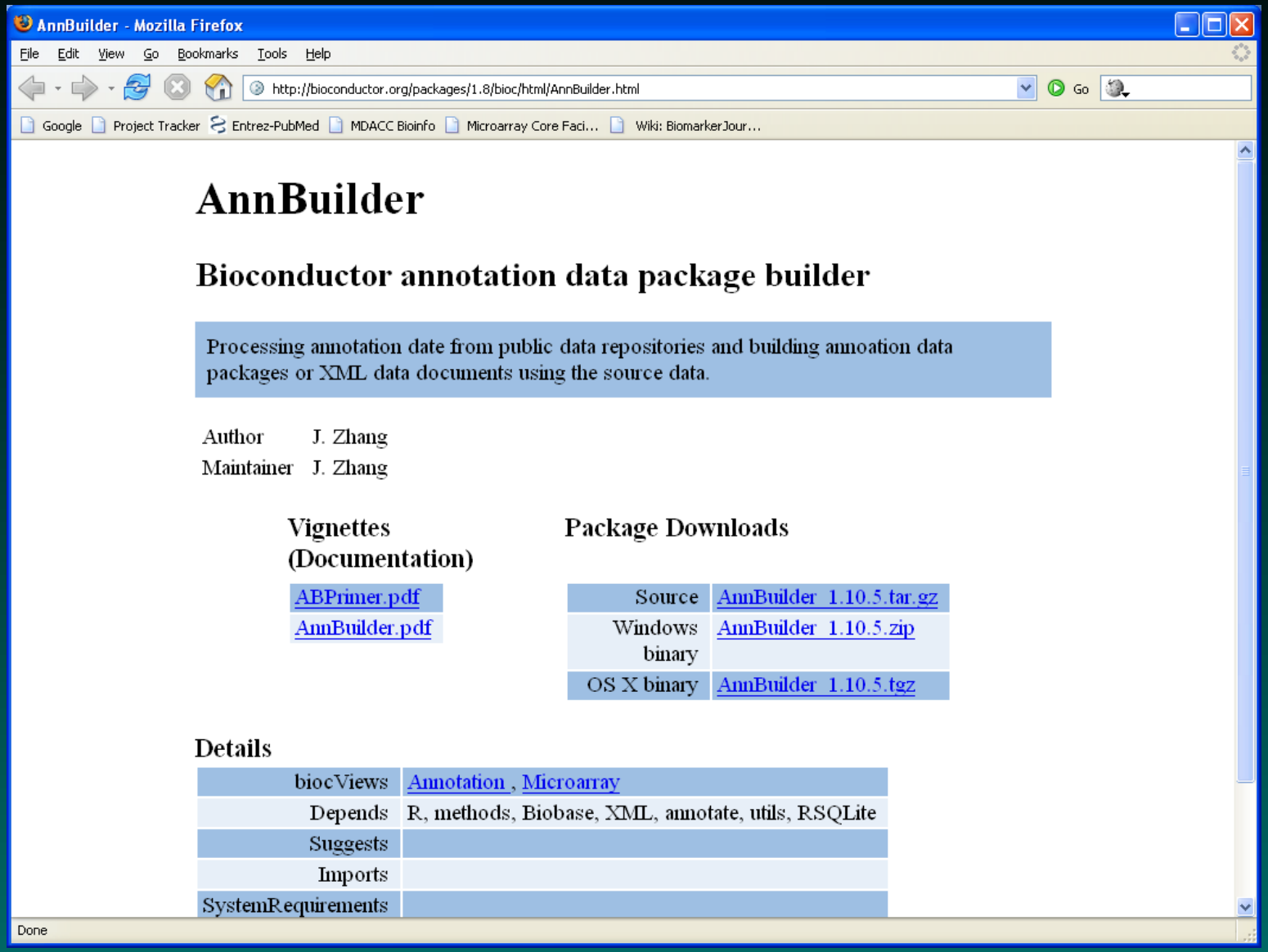
Keith Baggerly and Brad Broom
Department of Bioinformatics and Computational Biology
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
bmbroom@mdanderson.org

3 November 2009

# Lecture 20: Genome Browsing

- Annotation Environments in R
- AnnBuilder: Rolling Your Own Annotations
- The UCSC Genome Browser
- Chromosome Locations
- Building a Custom Track
- Viewing Your Custom Track
- Thoughts about TCGA

# Documentation for the AnnBuilder Package



# Annotation Environments in R

For most Affymetrix arrays, annotation packages are available directly (and automatically) from BioConductor whenever you need them. These packages were built using `AnnBuilder`.

You can load one of these packages as follows:

```
> require(hgu95av2.db)
```

To see what is in an annotation package, use its name as a function:

```
> hgu95av2()
Quality control information for hgu95av2:


This package has the following mappings:
```

```
hgu95av2ACCNUM has 12625 mapped keys (of 12625 key
hgu95av2ALIAS2PROBE has 37934 mapped keys (of 379
hgu95av2CHR has 11957 mapped keys (of 12625 keys)
hgu95av2CHRLENGTHS has 25 mapped keys (of 25 keys
hgu95av2CHRLOC has 11789 mapped keys (of 12625 key
hgu95av2CHRLOCEND has 11789 mapped keys (of 12625
hgu95av2ENSEMBL has 11639 mapped keys (of 12625 ke
hgu95av2ENSEMBL2PROBE has 9021 mapped keys (of 902
hgu95av2ENTREZID has 11960 mapped keys (of 12625 E
hgu95av2ENZYME has 1978 mapped keys (of 12625 keys
hgu95av2ENZYME2PROBE has 725 mapped keys (of 725 E
hgu95av2GENENAME has 11960 mapped keys (of 12625 E
hgu95av2GO has 11363 mapped keys (of 12625 keys)
hgu95av2GO2ALLPROBES has 9581 mapped keys (of 958
hgu95av2GO2PROBE has 6774 mapped keys (of 6774 key
```

```
hgu95av2MAP has 11919 mapped keys (of 12625 keys)
hgu95av2OMIM has 10350 mapped keys (of 12625 keys
hgu95av2PATH has 4585 mapped keys (of 12625 keys)
hgu95av2PATH2PROBE has 203 mapped keys (of 203 key
hgu95av2PFAM has 11878 mapped keys (of 12625 keys
hgu95av2PMID has 11898 mapped keys (of 12625 keys
hgu95av2PMID2PROBE has 206993 mapped keys (of 206
hgu95av2PROSITE has 11878 mapped keys (of 12625 ke
hgu95av2REFSEQ has 11883 mapped keys (of 12625 key
hgu95av2SYMBOL has 11960 mapped keys (of 12625 key
hgu95av2UNIGENE has 11905 mapped keys (of 12625 ke
hgu95av2UNIPROT has 11764 mapped keys (of 12625 ke
```

Additional Information about this package:

```
DB schema: HUMANCHIP_DB
DB schema version: 1.0
Organism: Homo sapiens
Date for NCBI data: 2009-Mar11
Date for GO data: 200903
Date for KEGG data: 2009-Mar10
Date for Golden Path data: 2008-Sep3
Date for IPI data: 2009-Mar03
Date for Ensembl data: 2009-Mar6
```

# Getting Annotations From Environments

Each of the items in the package is an *environment*, which computer scientists may recognize better if we tell them it is a hash table. The key into the probe-based hash table environments is the manufacturers identifier (i.e., an Affymetrix probeset id such as `1854_at`).

```
> get("1854_at", hgu95av2MAP)
[1] "20q13.1"
> get("1854_at", hgu95av2CHRLOC)
      20
41729122
> get("1854_at", hgu95av2ENTREZID)
[1] "4605"
```

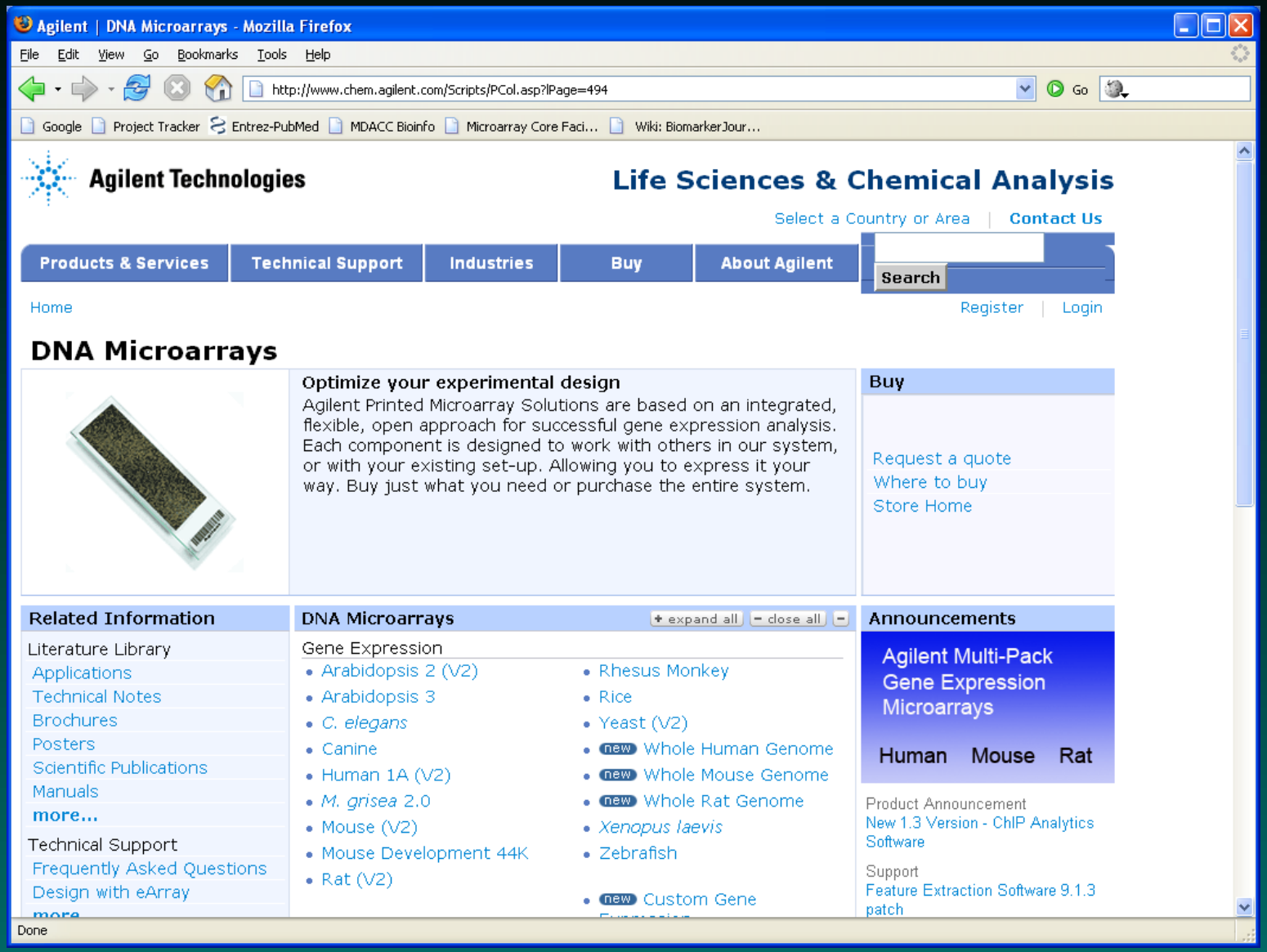
# More Getting Annotations From Environments

```
get("1854_at", hgu95av2REFSEQ)
[1] "NM_002466" "NP_002457"
> summary(hgu95av2REFSEQ)
REFSEQ map for chip hgu95av2 (object of class "An
|
| Lkeyname: probe_id (Ltablename: probes)
|    Lkeys: "1000_at", "1001_at", ... (total=1262
|
| Rkeyname: accession (Rtablename: refseq)
|    Rkeys: "NM_000015", "NM_000016", ... (total=
|
| direction: L --> R
get("NM_002466",revmap(hgu95av2REFSEQ))
[1] "1854_at"
```

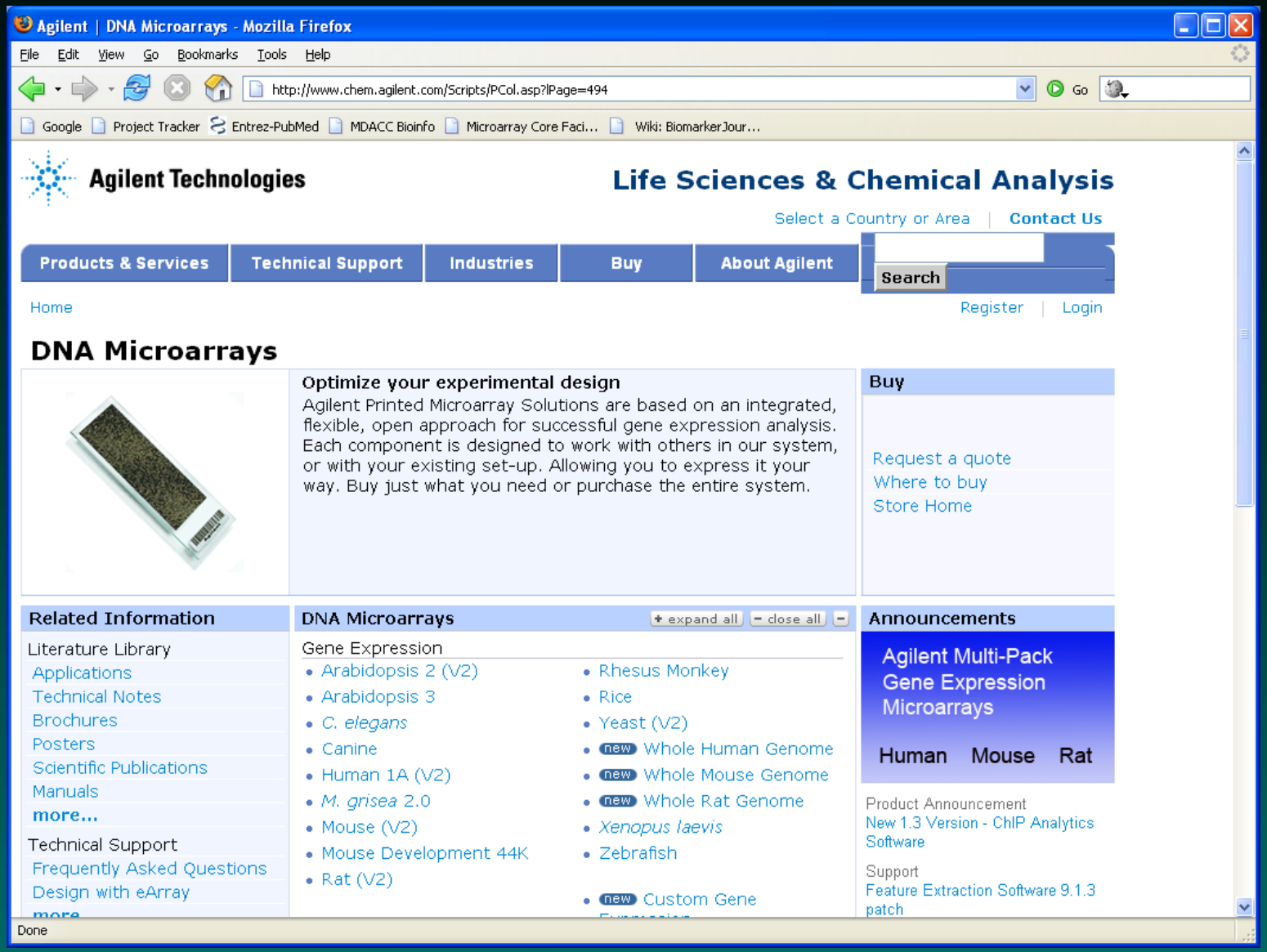
# AnnBuilder: Rolling Your Own Annotations

We recently had to analyze some data from an Agilent 44K two-color glass microarray. The corresponding annotation package was not available, so we had to build our own. Finding the manufacturers basic annotations was a nontrivial task. We started at the web site (http://www.agilent.com), then followed the link under "Products and Services" for "Life Sciences" and "Instruments and Systems" to get to the "DNA Microarrays" page.

# Follow the Link for "Human Genome, Whole"



# Try "Download Gene Lists (Specifications)"



# Reading the Feature Info

In any event, we finally obtained a pair of files that contained the mappings from spots to genomic material. We used the `read.table` command to get this file into R:

```
featureInfo <-
  read.table(file.path("GeneList",
     "014850_D_GeneList_20090416.txt"),
     header = TRUE, row.names = NULL,
     sep = "\t", quote = "",
     comment.char = "")
```

# Looking at the Feature Info

Here is part of the file:

```
> colnames(featureInfo)
[1] "ProbeID"     "TargetID"    "GeneSymbol"
[4] "GeneName"    "Accessions"  "Description"
> featureInfo[1:5,c("ProbeID","Accessions")]
       ProbeID
1 A_24_P919016                    gb|A18395|thc|NP238262|tl
2  A_32_P27041 gb|A18658|gb|AI819757|gb|AW299939|c
3 A_24_P693768                      gb|A18658|gb|BQ367072|c
4 A_24_P475014       gb|AA004321|gb|T84046|gb|T8372
5 A_24_P456043                             gb|AA004800|c
```

# What We Need

The critical information is given by the columns that contain the manufaturers identifier (`ProbeID`) and the GenBank or RefSeq accession numbers (`Accessions`). Ideally, we want one type of annotation.

```
allAnnotations <-
  as.character(featureInfo[,"Accessions"])
splitAnnotations <-
  strsplit(allAnnotations, "\\|")
firstAnnotation <-
   lapply(splitAnnotations,function(x){x[1]})
table(unlist(firstAnnotation))
  ens    gb   ref   thc
  909  7988 26631  2441
```

# What We Need

The function we are going to use to build annotations requires only these two columns to be present in a file. So we make them available for a few genes:

```
secondAnnotation <-
  unlist(lapply(splitAnnotations,function(x){x[2]
temp <-
  cbind(as.character(featureInfo[,"ProbeID"]),
    secondAnnotation)
write.table(temp[1:10,],"agilentGenesShort.tsv",
  sep="\t", quote=FALSE, col.names=NA)
```

# Setting Up the Annotation Package

```
> library(AnnBuilder)
> baseName <- "agilentGenes.tsv"
> baseType <- "gb"
> srcUrls <-
    getSrcUrl("all",
      organism = "Homo sapiens")
> myDir <- getwd()
```

# Building the Annotation Package

The next command takes a **very** long time, since it makes calls to databases all over the internet for every one of the $44,000$ probes on the array.

```
ABPkgBuilder(baseName = baseName,
  srcUrls = srcUrls, baseMapType = baseType,
  pkgName = "Agilent44K", pkgPath = myDir,
  organism = "Homo sapiens", version = "1.0",
  author = list(authors = "krc@mdacc.tmc.edu",
    maintainer = "krc@mdacc.tmc.edu"),
  fromWeb = TRUE)
```

# Producing the Final Package

This command produces the **source** for a package, which must still be compiled and zipped into a binary package that can be installed easily. This task is most easily accomplished on a UNIX based machine:

```
helios% R CMD build Agilent44K
helios% R CMD build --binary Agilent44K
```

You can then convert the resulting `.tar.gz` file to a `.zip` file, which is the preferred form for distributing a Windows package.

You can check out the results by getting the annotation package from our course web site.

# The Agilent 44K Annotations

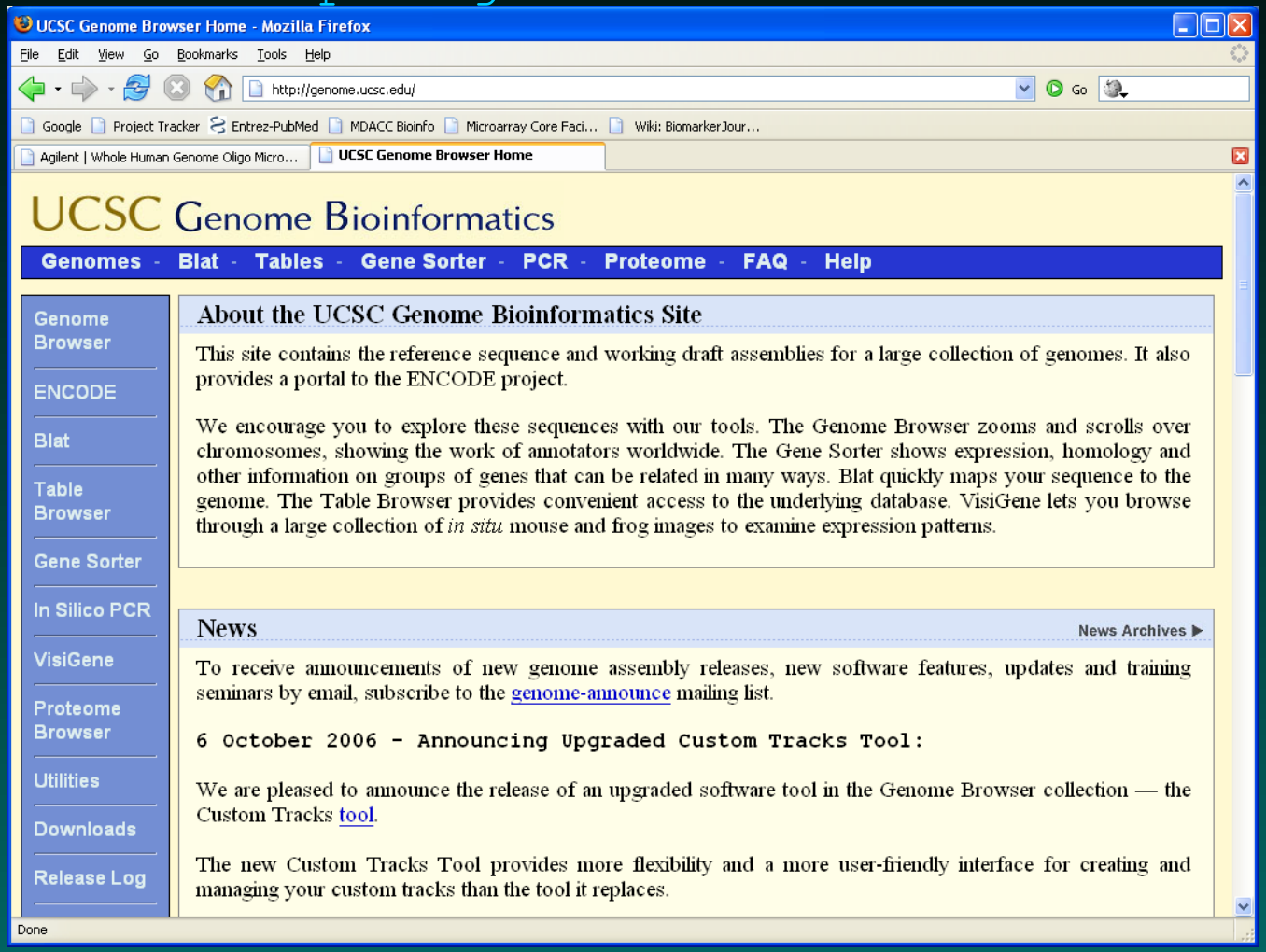
```
> library(Agilent44K)
> Agilent44K()
Quality control information for  Agilent44K
Date built: Created: Sun Sep 03 07:50:38 2006

Number of probes: 41001
Probe number missmatch: None
Probe missmatch: None
Mappings found for probe based rda files:
 Agilent44KACCNUM found 41001 of 41001
 Agilent44KCHR found 31185 of 41001
 Agilent44KCHRLOC found 28795 of 41001
 Agilent44KENZYME found 3056 of 41001
 Agilent44KGENENAME found 27824 of 41001
```

```
 Agilent44KGO found 23644 of 41001
 Agilent44KLOCUSID found 31224 of 41001
 Agilent44KMAP found 30939 of 41001
 Agilent44KOMIM found 17942 of 41001
 Agilent44KPATH found 6715 of 41001
 Agilent44KPMID found 30361 of 41001
 Agilent44KREFSEQ found 30057 of 41001
 Agilent44KSUMFUNC found 0 of 41001
 Agilent44KSYMBOL found 31217 of 41001
 Agilent44KUNIGENE found 31010 of 41001
Mappings found for non-probe based rda files:
   Agilent44KCHRLENGTHS found 25
 Agilent44KENZYME2PROBE found 794
 Agilent44KGO2ALLPROBES found 6883
 Agilent44KGO2PROBE found 5117
```

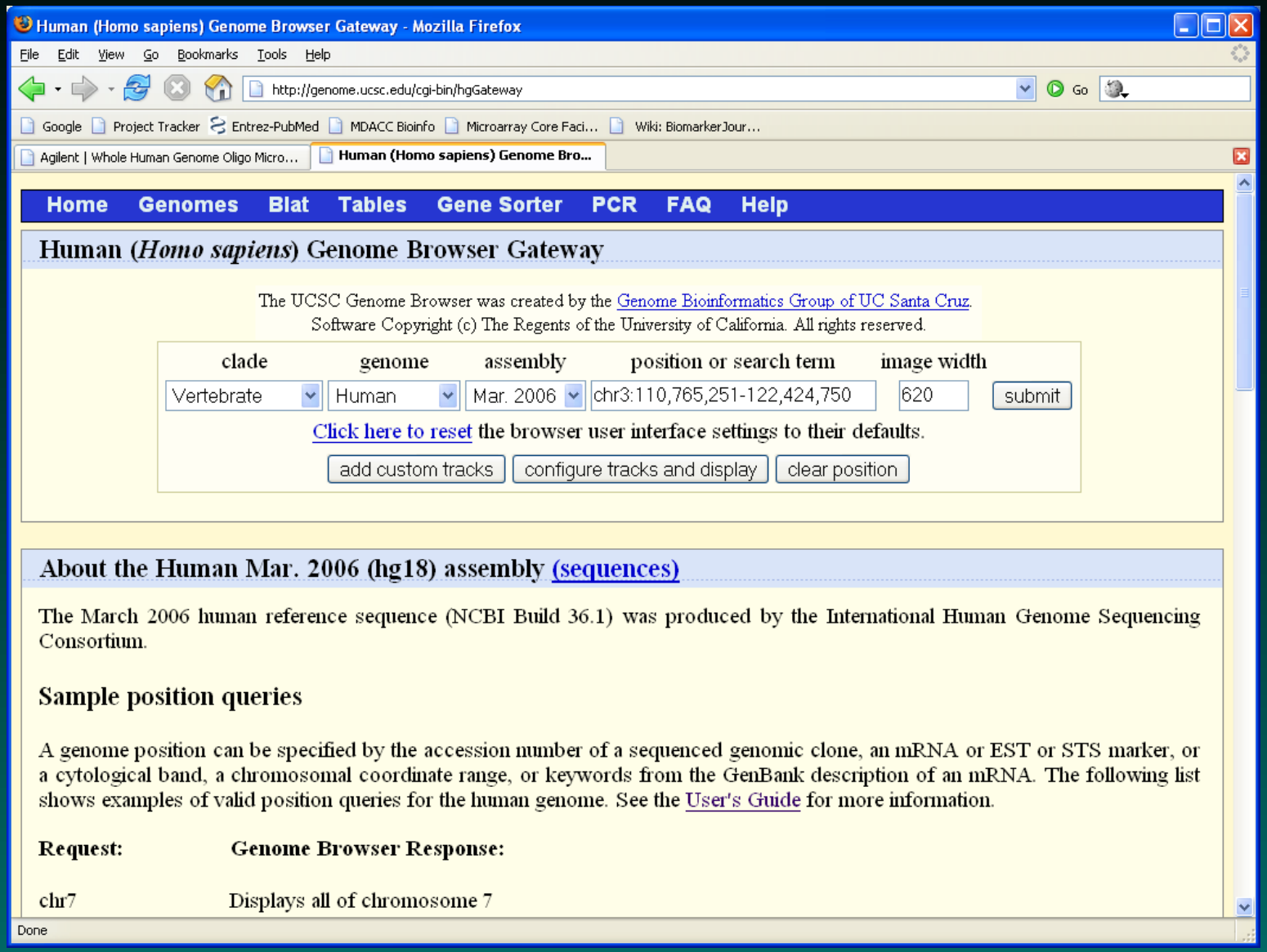
```
Agilent44KORGANISM found 1
Agilent44KPATH2PROBE found 183
Agilent44KPFAM found 21902
Agilent44KPMID2PROBE found 131104
Agilent44KPROSITE found 15055
```

# The UCSC Genome Browser

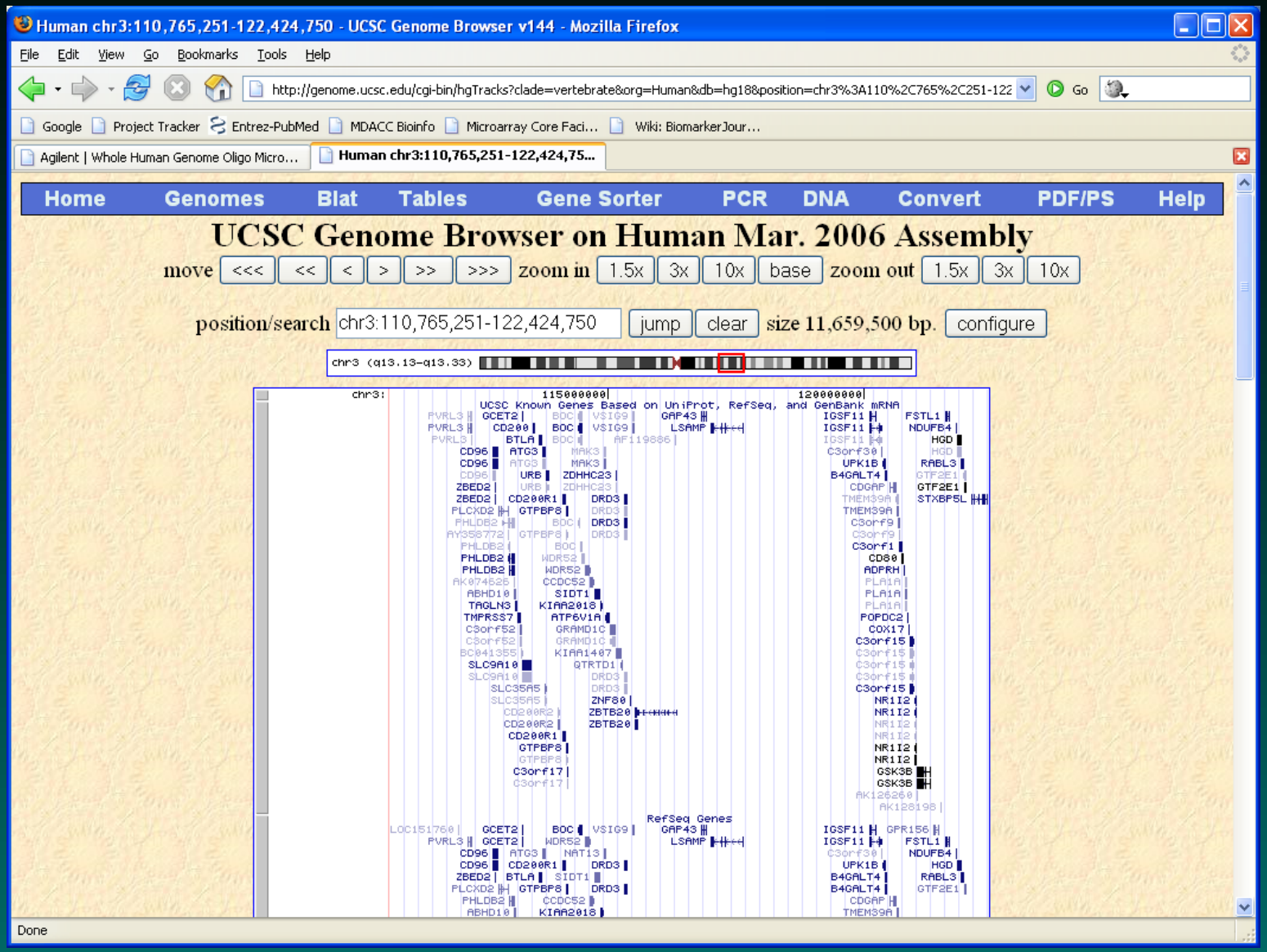
http://genome.ucsc.edu/



# Follow the Link to "Genome Browser"



# Press "Submit" to Start Browsing



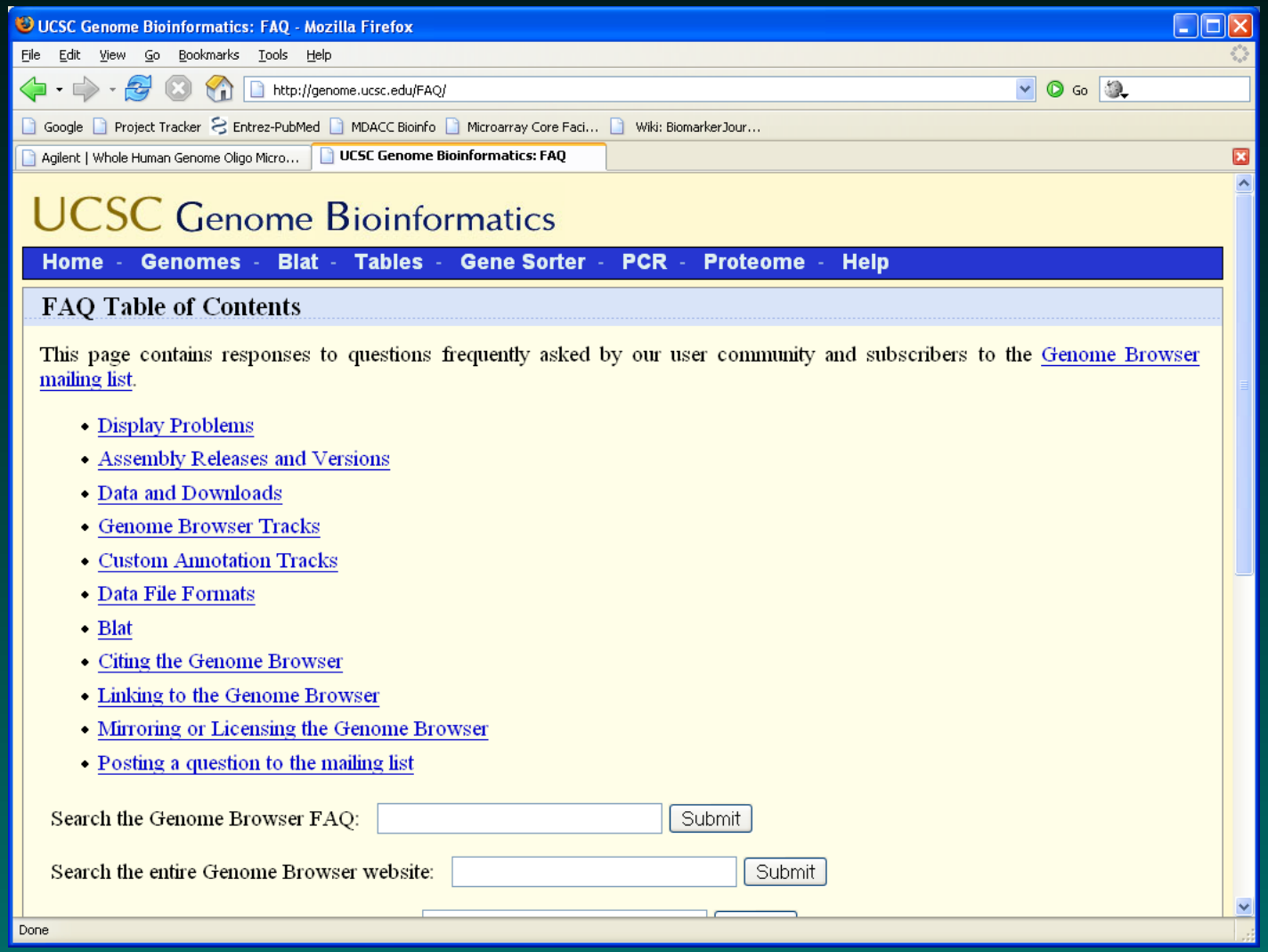
# About the Genome Browser

The genome browser lets you see a great deal of information laid out along the latest completed build of the human genome. The most obvious thing to look at are the known genes, which are typically displayed in such a way that you can see the individual introns and exons (provided you zoom in closely).

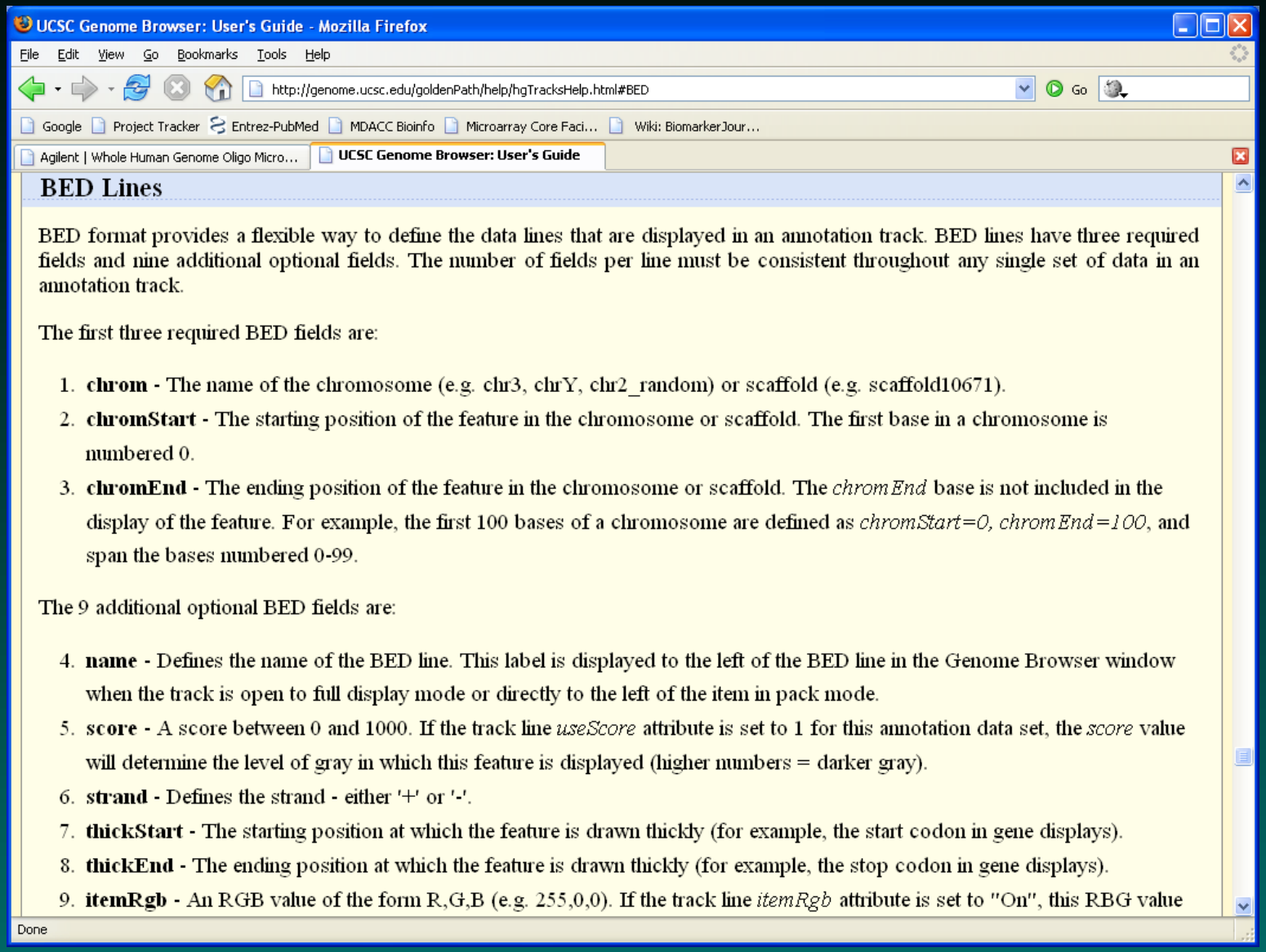
For our purposes (as people who analyze microarray data), an extremely interesting feature of the Genome Browser is that it lets you add your own “Custom Tracks”, which is their name for a set of annotations you can define.

# Custom Tracks

To learn about the genome (custom) tracks, go to the FAQ.



# BED Format

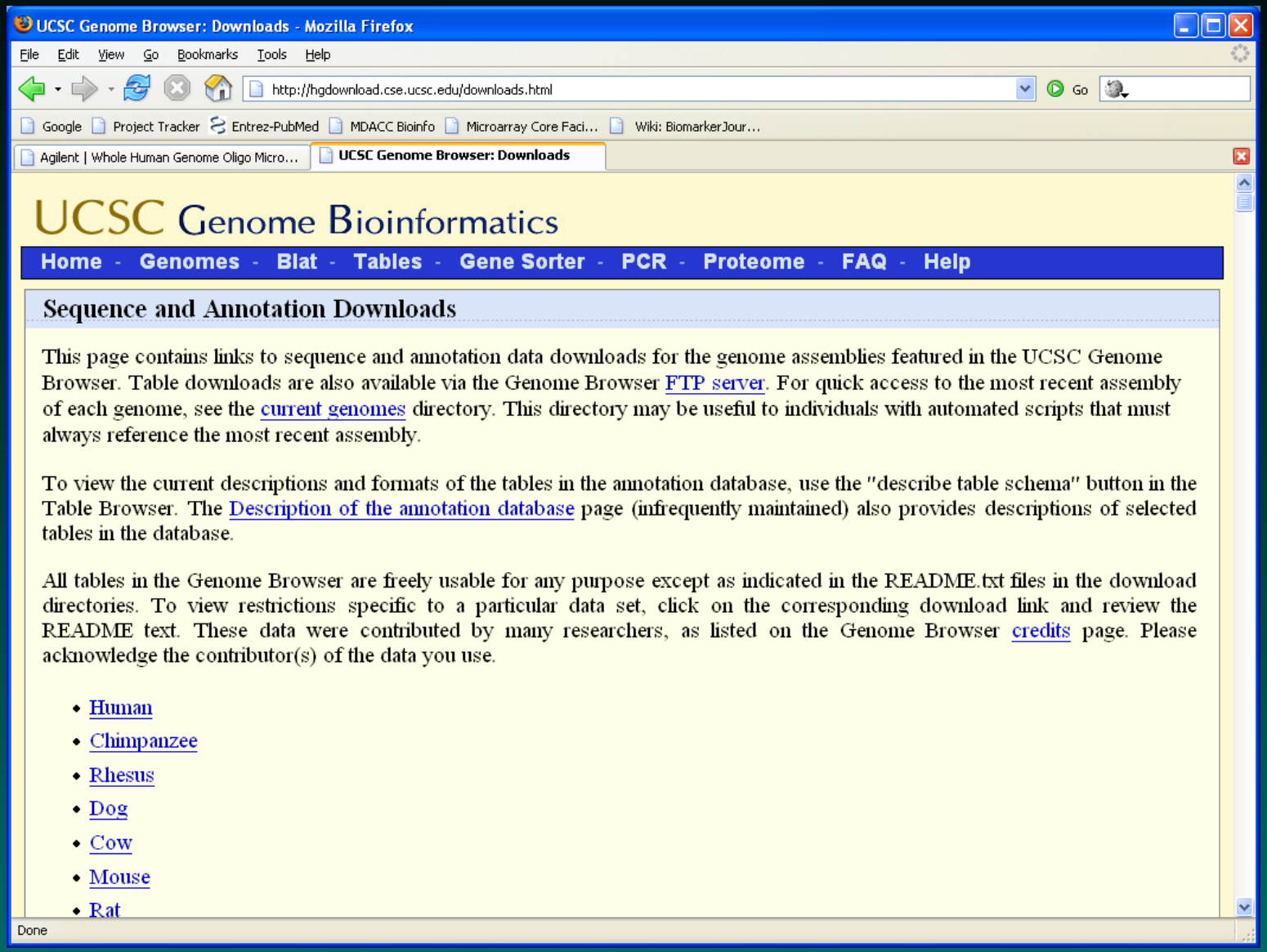


# Chromosome Locations

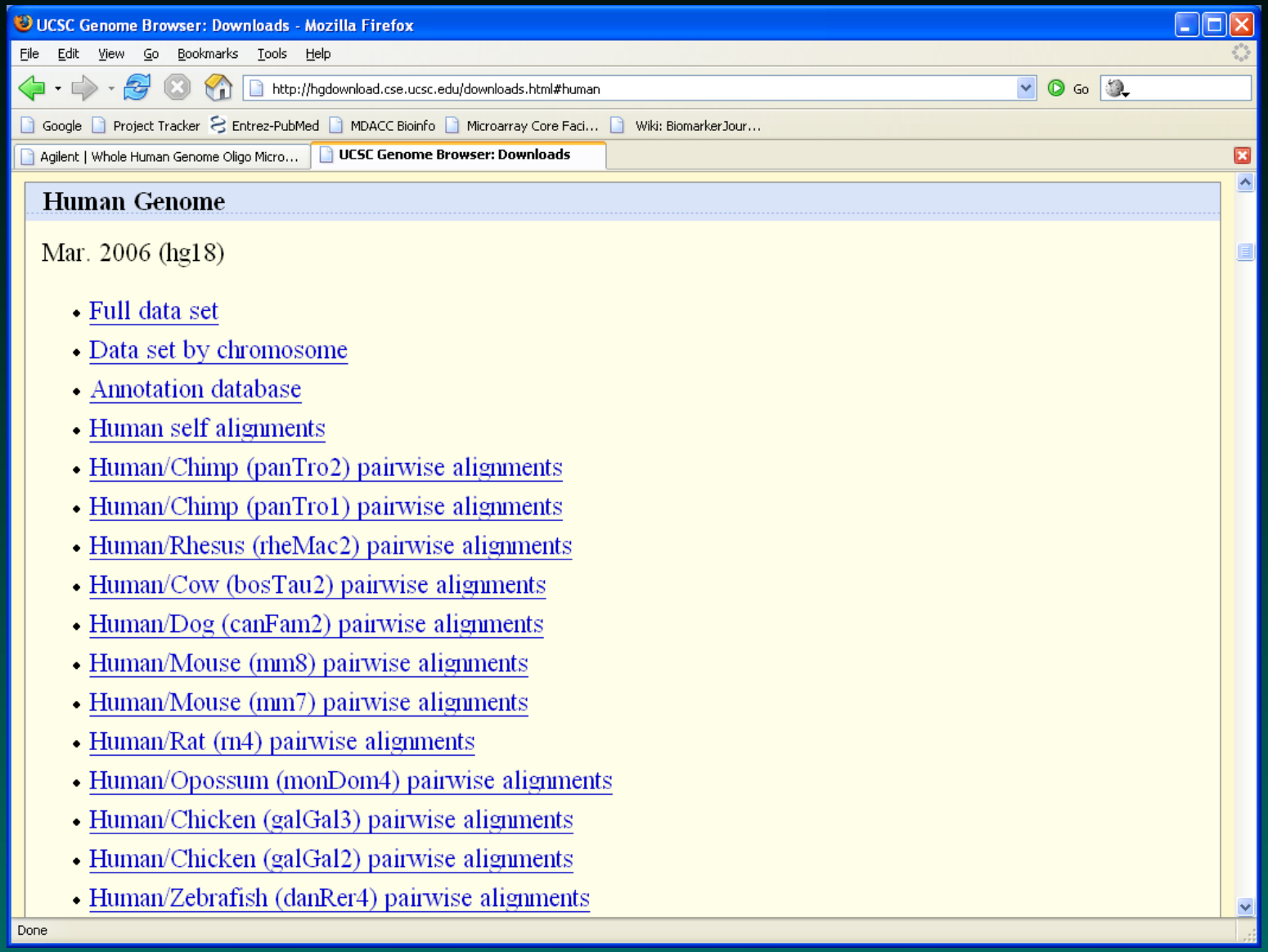
You can read more of the custom track documentation on your own; here, we are going to focus on how to build a custom track in R. The first thing we want to point out is that we need to know both the starting base location and the ending base location in order to build a custom track. Thus, the `CHRLOC` annotations that `AnnBuilder` constructs are not adequate.

Fortunately, we can get start and end points directly from the folks at the UCSC Genome Browser. Go back to the main page, then follow the link for “Downloads”.

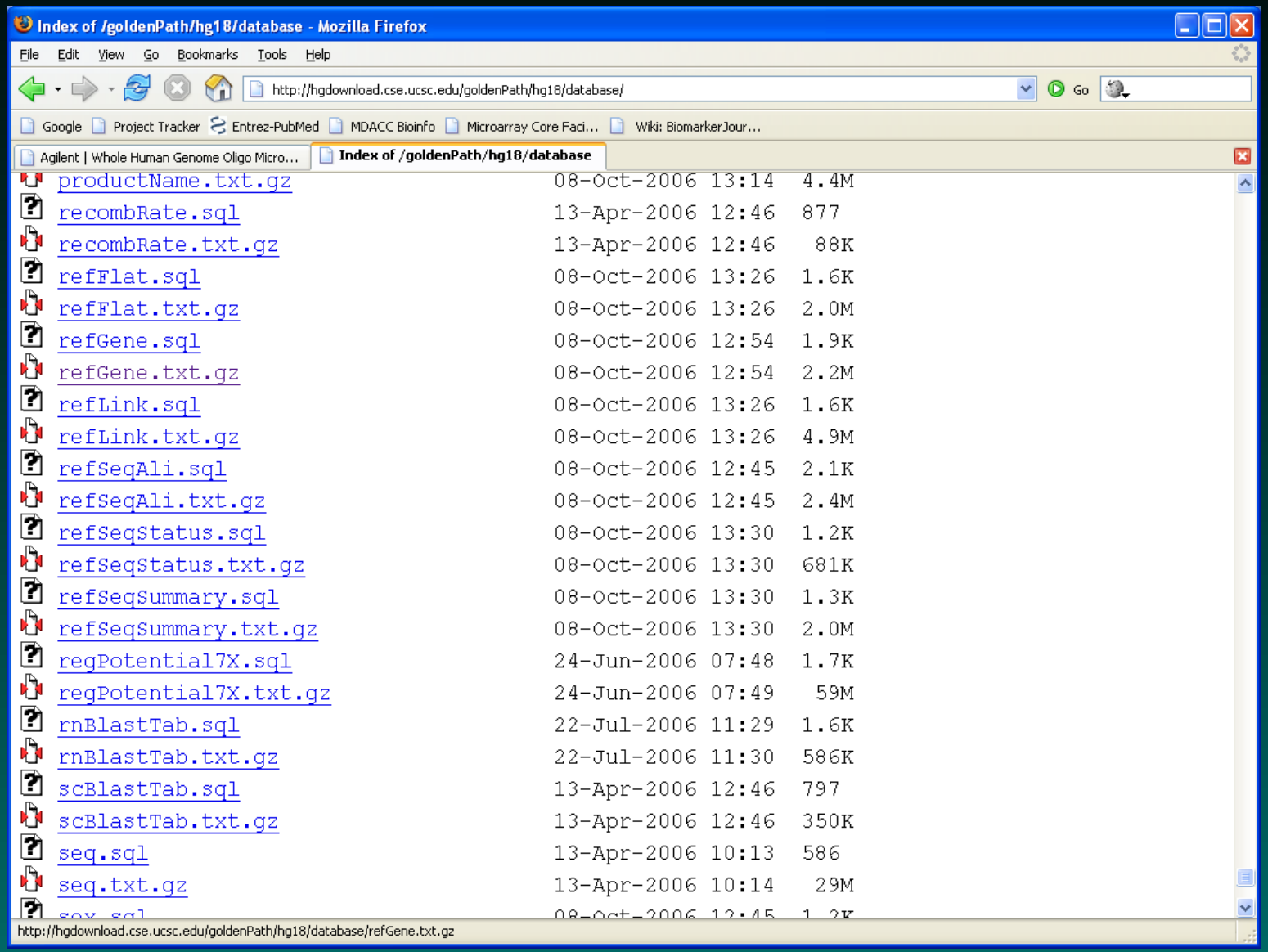
# UCSC Download Page



# Follow the link for "Human"



# In "Annotation Database", Scroll To "refGene"



# Using the RefGene locations in R

Load the file.

```
> refgene <- read.table("refGene.txt", header = F
      sep = "\t", comment.char = "", quote = "")
```

Add the column names, which are not included.

```
> colnames(refgene) <- c("bin", "name", "chrom",
      "strand", "txStart", "txEnd", "cdsStart",
      "cdsEnd", "exonCount", "exonStarts", "exonEn
      "id", "name2", "cdsStartStat", "cdsEndStat",
      "exonFrames")
```

We are going to ignore the intron and exon boundaries. We are also going to remove duplicate entries, which seem for some reason to exist; the search to identify these is long.

# More RefGene

```
> temprg <- refgene[, c(1:9, 13:15)]
> omit <- unlist(lapply(levels(temprg$name),
    function(x, n) {
     which(n == x)[1]
    }, as.character(temprg$name)))
> summary(omit)
> refgene <- temprg[omit, ]
> rownames(refgene) <-
    as.character(refgene[, "name"])
```

Finally, we save this as a binary object that we can load later.

```
> save(refgene, file = "refgene.rda")
```

# Linking the Agilent Array to RefGene locations

First, convert the environment in the AnnBuilder package for the Agilent 44K arrays to a list.

```
> temp2 <- as.list(Agilent44KREFSEQ)
```

Next, we produce a list that maps the annotations to the spots. This code works because the `Accessions` column of the `featureInfo` object contains RefSeq IDs (primarily), which are the names of the rows in the `temp2` object we just created.

```
> ag.annoList <- temp2[as.character(featureInfo[,
(filtering for RefSeq here)
```

# Alternative Splicing

```
> ag.annoList[1]
$A_23_P80353
[1] "NM_0010003689" "NP_0010003689" "NM_031488"
[4] "NP_113676"
```

Notice that some probes are associated with more than one RefSeq gene; this happens because different isoforms (produced by alternative splicing) of the same gene have different RefSeq identifiers. That is, the same piece of DNA can give rise to different mRNA molecules. So, we now search through and select just the first annotation for each spot.

# Grabbing the First

```
> agilent.lc <- unlist(lapply(ag.annoList, length
> agilentREFSEQ <- unlist(lapply(ag.annoList, func
    if (length(x) == 0) {
        return(NA)
    }
    if (length(x) == 1) {
        return(x)
    }
    idx <- 1
    while (idx <= length(x)) {
        if (x[[idx]] == "") {
            idx <- idx + 1
            next
        }
```

```
            return(x[[idx]])
        }
        return(NA)
  }))
> agilentREFSEQ[agilentREFSEQ == ""] <- NA
```

# Checking the Output

```
> length(agilentREFSEQ)
[1] 41675
> sum(!is.na(agilentREFSEQ))
[1] 30612
```

Finally, we use the updated RefSeqs (that we just constructed in the `agilentREFSEQ` object) as indices into the `refgene` chromosome locations above. This computation is also slow, since it uses a search in a list instead of in a hash.

# Checking More Output

```
> agilent2refgene <- refgene[agilentREFSEQ, ]
> agilent2refgene[1:3, ]
              bin          name chrom strand  txSta
NM_001003689 889 NM_001003689 chr22      + 399312
NM_005503     98    NM_005503 chr15      + 270011
NM_004672    795    NM_004672  chr1      - 275542
                txEnd cdsStart   cdsEnd exonCount
NM_001003689 39957220 39931312 39953547        18
NM_005503    27197806 27133379 27196628        14
NM_004672    27565924 27554468 27565675        29
             cdsStartStat cdsEndStat
NM_001003689         cmpl       cmpl
NM_005503            cmpl       cmpl
NM_004672            cmpl       cmpl
```

# Building a Custom Track

We analyzed the Agilent 44K microarray data using a linear model. The results are contained in an object called ourResults:

```
> summary(ourResults)
 UntreatedMeanLog       Beta                PValue
 Min.   : 4.870   Min.   :-3.15530   Min.   :2.02
 1st Qu.: 6.907   1st Qu.:-0.19572   1st Qu.:8.14
 Median : 8.058   Median :-0.05431   Median :2.74
 Mean   : 8.742   Mean   :-0.04300   Mean   :3.51
 3rd Qu.: 9.982   3rd Qu.: 0.10075   3rd Qu.:5.82
 Max.   :16.523   Max.   : 3.27672   Max.   :1.00
```

# Computing a Displayable Score

We are going to us the p-values to decide which genes to display, and we are going to use the coefficient (`Beta`) to compute a score that shows the amount of differential expression. The allowed scores for a custom track range from $0$ to $1000$. Since the true values of `Beta` range between $-3$ and $+3$ (more or less), we are going to multiply by $300$ to get a useful score.

```
score <- 300 * ourResults[, "Beta"]
score[score > 1000] <- 1000
score[score < -1000] <- -1000
score <- abs(score)
```

# A Track Data Frame

Now we build a data frame that includes the information we need for a custom track in the desired order:

```
> temp <- data.frame(agilent2refgene[, c("chrom",
      "txStart", "txEnd", "name2")], score = score,
      strand = agilent2refgene[, "strand"])
> temp[1:3, 1:5]
              chrom  txStart    txEnd   name2      s
NM_001003689 chr22 39931258 39957220 L3MBTL2 96.9
NM_005503    chr15 27001144 27197806   APBA2 74.4
NM_004672     chr1 27554256 27565924  MAP3K6  2.2
```

# Significant Overexpressed Genes

We built this data frame for all genes; now we are going to select the ones that are significant (p-value $< 0.02$) and are overexpressed in response to the treatment ($\beta > 0$). We further restrict to those genes that we are able to map.

```
> trackInfo <- temp[!is.na(temp[, "chrom"]) & our
      "PValue"] < 0.02 & ourResults[, "Beta"] >
      0, ]
```

We also have to create a header line that tells the browser to make use of the scores.

```
> trackheader <- paste("track name=upNormal",
      "description=\"Increased in Normal Cells\"",
      "useScore=1 color=0,60,120")
```

# Writing the Track Info to a File

We can now write the header line followed by the track data:

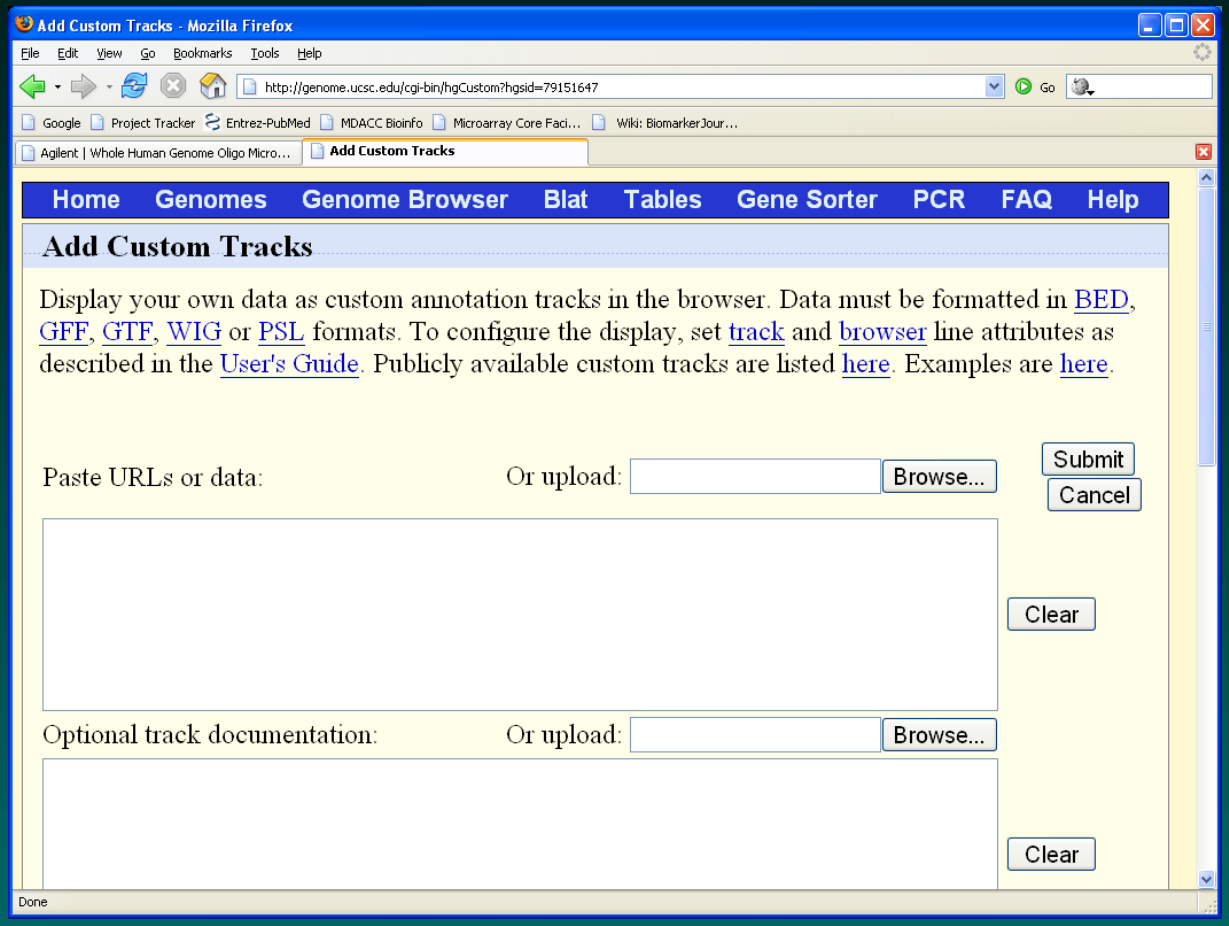
```
> write(trackheader, file = "upNormalRNA.tsv",
      append = FALSE)
> write.table(trackInfo, file = "upNormalRNA.tsv",
      append = TRUE, quote = FALSE, sep = "\t",
      row.names = FALSE, col.names = FALSE)
```

Finally, we do the same thing for the genes that are underexpressed.

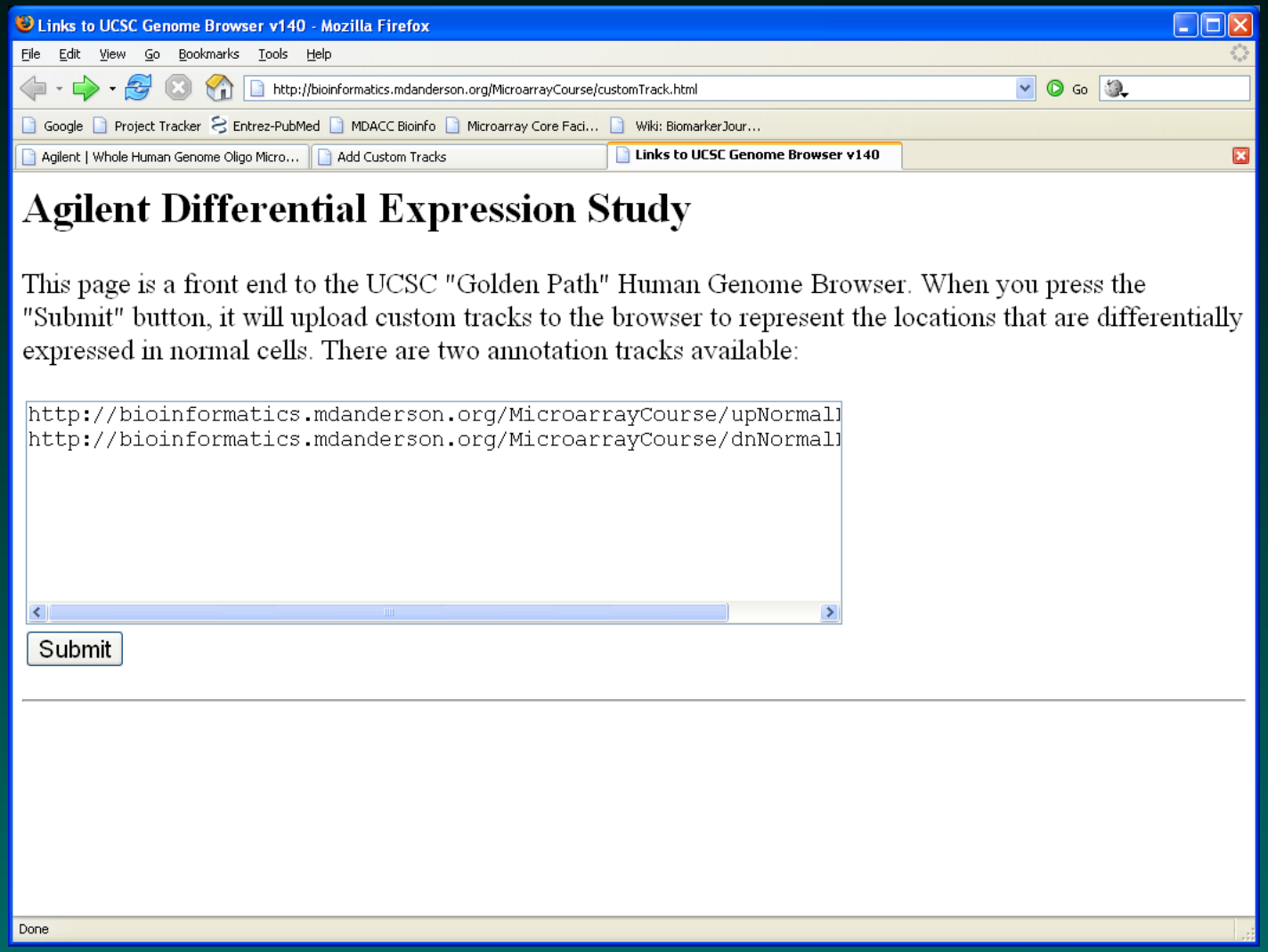
```
> trackInfo <- temp[!is.na(temp[, "chrom"]) & our
      "PValue"] < 0.02 & ourResults[, "Beta"] <
      0, ]
> trackheader <- paste("track name=downNormal",
      "description=\"Decreased in Normal Cells\"",
      "useScore=1 color=100,50,0")
> write(trackheader, file = "dnNormalRNA.tsv",
      append = FALSE)
> write.table(trackInfo, file = "dnNormalRNA.tsv",
      append = TRUE, quote = FALSE, sep = "\t",
      row.names = FALSE, col.names = FALSE)
```

# Viewing Your Custom Track

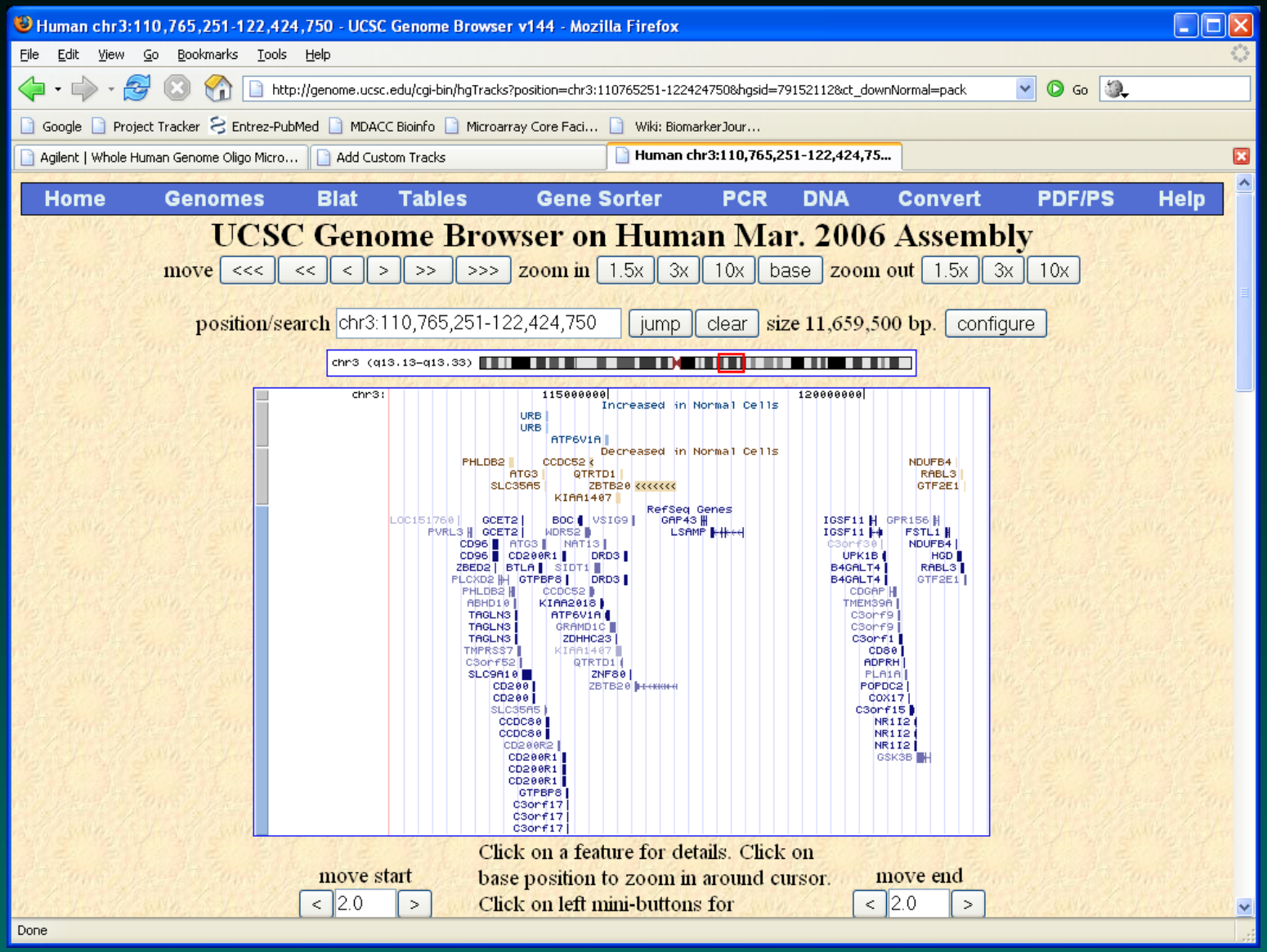
Now we can return to the genome browser and look at our custom tracks.



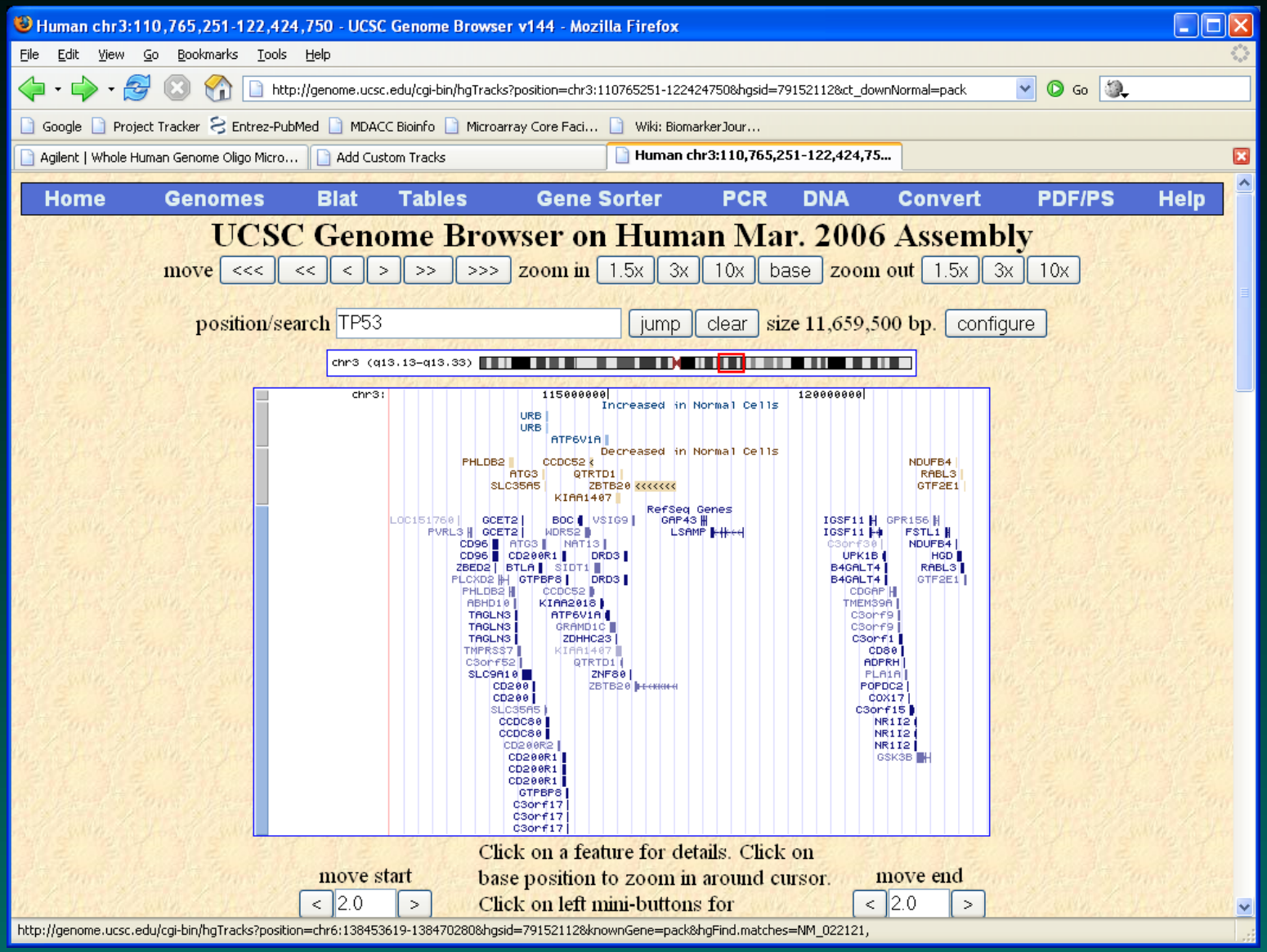
# http://bioinformatics.mdanderson.org/MicroarrayCourse/customTrack.html



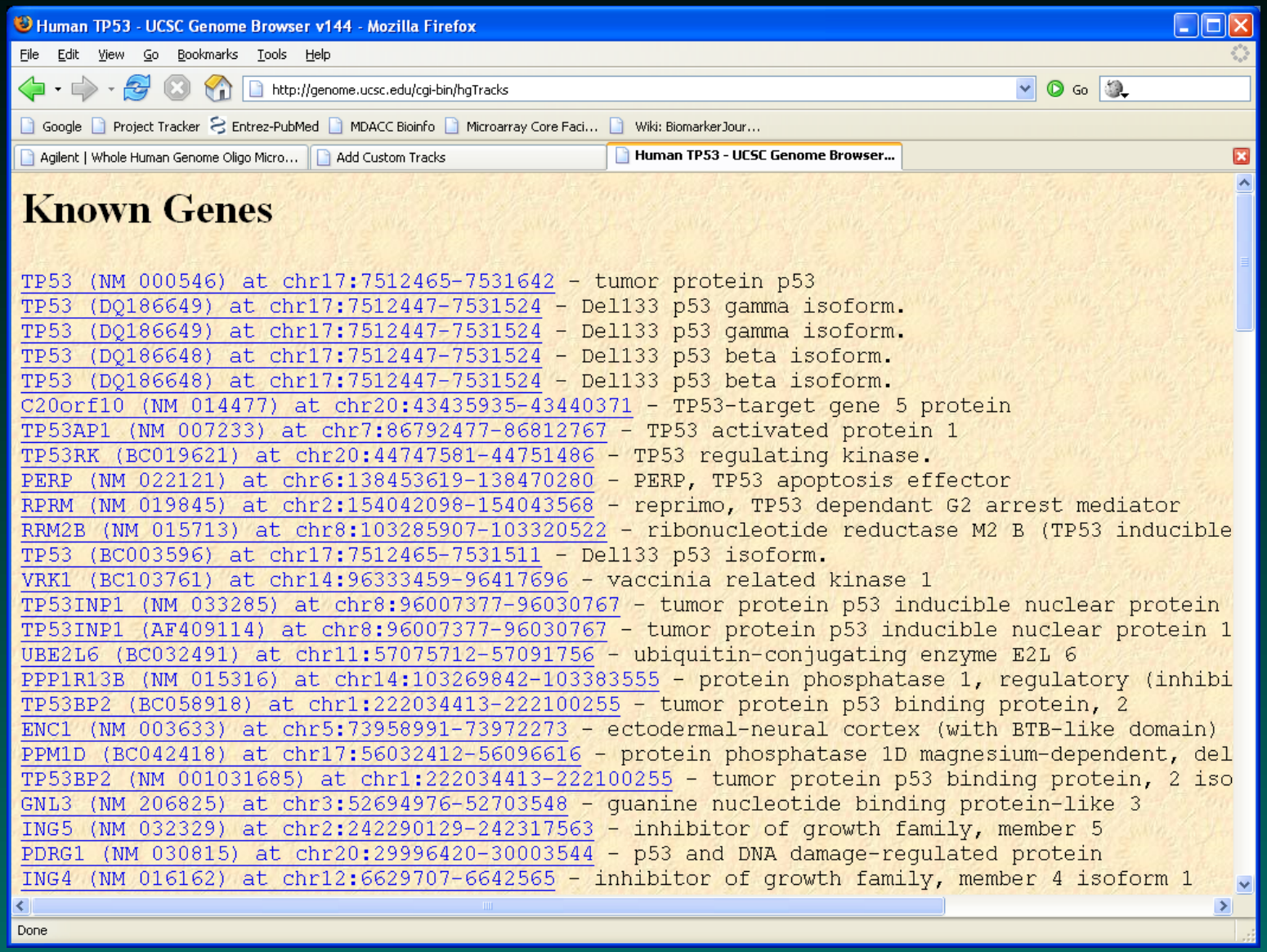
# Displaying Our Tracks



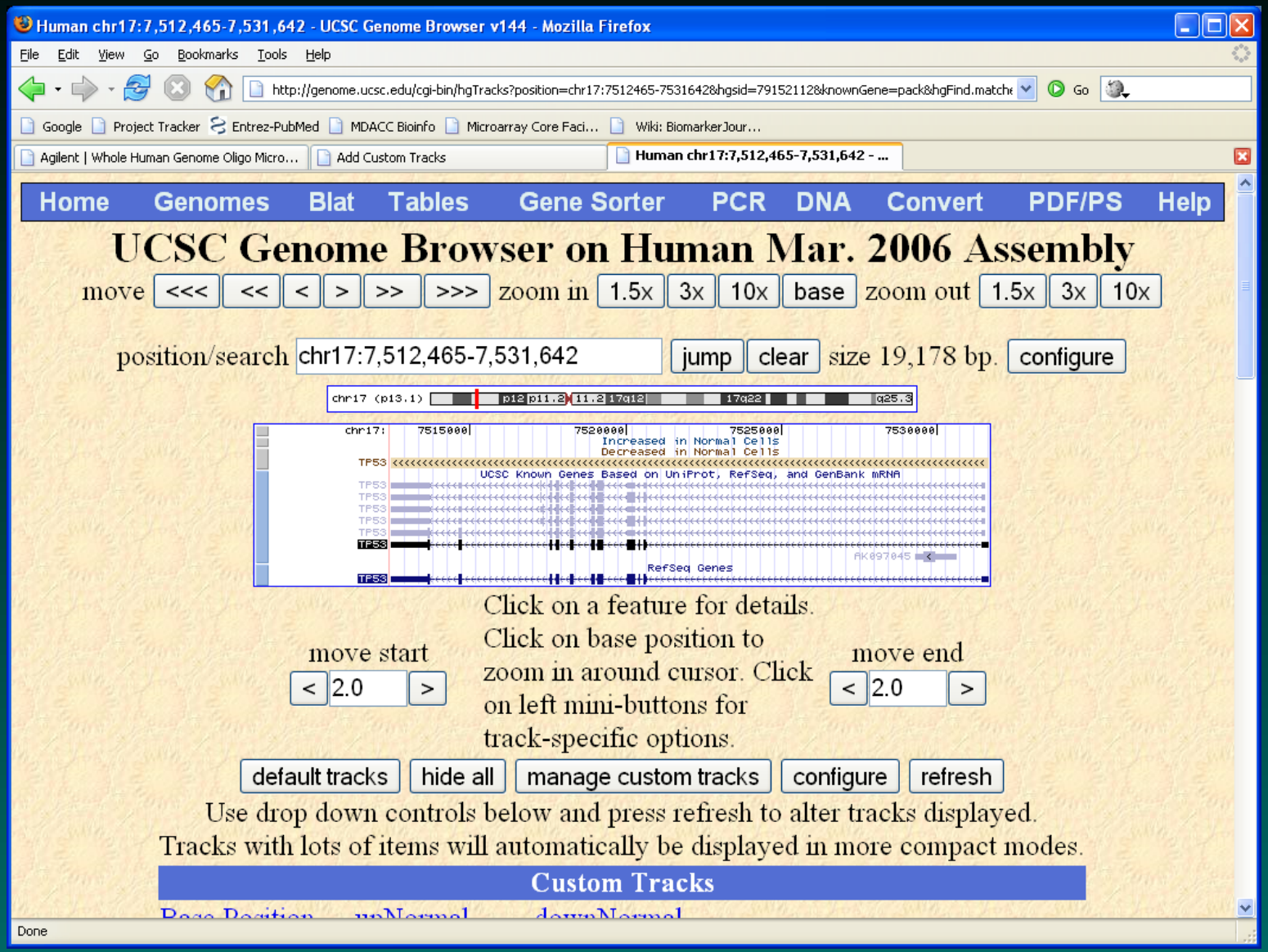
# Searching for a Gene



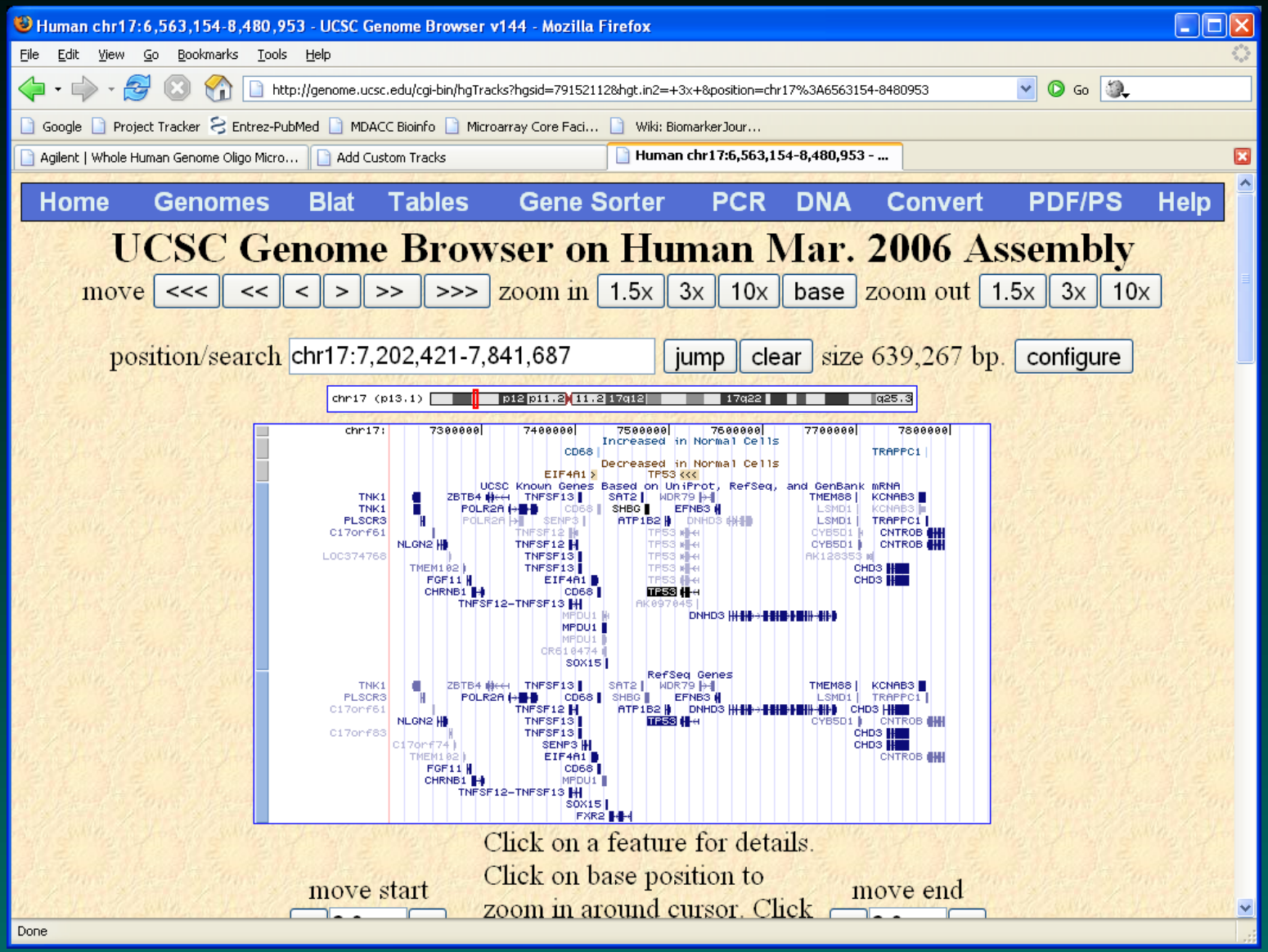
# Searching for a Gene



# Searching for a Gene



# Searching for a Gene



# Comments on TCGA

What's there?

from the *Broad*:
ht_hg_u133a

from *Harvard*:
hg-cgh-244a
hg-cgh-415k-g4124a
illumina-mrna-dge

from *Johns Hopkins*:
humanmethylation27

# Comments on TCGA (2)

from *Memorial Sloan-Kettering*:
hg_cgh_244a
cgh_1x1m_g4447a

from *U North Carolina*:
agilent4502a-07-2
agilent4502a-07-3
h_mirna_8x15K