RSNPper: utilities for SNP data

VJ Carey stvjc@channing.harvard.edu

April 25, 2006

Contents
1 Introduction 1
2 How it works 1
3 Overview of the functions 2
4 Demonstrations 3
  4.1 Obtaining information on genes 3
  4.2 Obtaining information on SNPs 4
5 Application: SNP density on chr 1 7

1 Introduction

This document describes RSNPper version 1.0, added to Bioconductor in October of 2003. This first version focuses on SNP metadata, with functions that retrieve SNP-related data from the Boston Children’s Hospital Informatics Program SNPper web service.

Earlier non-released versions of this package included considerable code for working with prettybase format and for conducting other tasks in SNP discovery projects. That material has been moved to inst/OLD and may be re-introduced later. Users seeking legacy support should contact the author.

2 How it works

Loading required package: XML

The core of this package is the XML-RPC service at CHIP accessible through the following URL stub:
> print(.SNPperBaseURL)


The `useSNPper` function allows you to work directly with the XML-RPC server by packing up appropriate command and argument strings.

> dput(useSNPper)

```r
function (cmd, parmstring) {
  targ <- url(paste(.SNPperBaseURL, cmd, parmstring, sep = ""))
  open(targ)
  on.exit(close(targ))
  readLines(targ)
}
```

> print(useSNPper("geneinfo", ",&name=CRP")[1:7])

```
[1] " <SNPPER-RPC SOURCE="*RPCSERV-NAME*" VERSION="$Revision: 1.38 $" GENOME="hg17" DBSNP="123">"
[2] " <GENEINFO>"
[3] "   <GENE ID="1440">"
[4] "     <GENEID>1440</GENEID>"
[5] "     <NAME>CRP</NAME>"
[6] "   <CHROM>chr1</CHROM>"
[7] "   <STRAND>-</STRAND>"
```

The main functions of `RSNPper` attend to simplifying specification of parameters and parsing and packaging the XML results.

**Note on auditability.** All functions return textual information coupled with auditing information as a 'toolInfo' attribute, detailing the SNPper supplied information on the human genome sequence build, the dbSNP version, and the SNPper version from which the results are obtained. At present, there is one exception: when `itemsInRange` is invoked with `item='countsnps`, no toolInfo data is obtained. This will be corrected once the `countsnps` command at SNPper returns valid XML element tags.

### 3 Overview of the functions

The current set of functions intended for investigative use is:

- **geneInfo** – general information about location and nomenclature
- **geneLayout** – information about exon locations
• geneSNPs – all SNPs associated with a given gene
• SNPinfo – detailed information on a SNP
• itemsInRange – supports chromosome scanning for genes, SNPs, or counts of SNPs

An omission: for SNP information, I have not collected information on submitter.

4 Demonstrations

4.1 Obtaining information on genes

The geneInfo function will collect some basic information on a gene. The gene may be specified by HUGO name, mRNA accession number, or SNPper id.

> print(geneInfo("CRP"))

<table>
<thead>
<tr>
<th>snpper.ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;1440&quot;</td>
<td>&quot;CRP&quot;</td>
</tr>
<tr>
<td>CHROM</td>
<td>STRAND</td>
</tr>
<tr>
<td>&quot;chr1&quot;</td>
<td>&quot;-&quot;</td>
</tr>
</tbody>
</table>

PRODUCT LOCUSLINK
"C-reactive protein, pentraxin-related" "1401"

OMIM UNIGENE
"123260" "Hs.76452"

SWISSPROT NSNPS
"P02741" "101"

REFSEQACC MRNAACC
"" ""NM_000567"

TRANSCRIPT.START CODINGSEQ.START
"156495525" "156496388"

TRANSCRIPT.END CODINGSEQ.END
"156497437" "156497348"

attr(,"toolInfo")

SOURCE VERSION GENOME DBSNP
"*RPCSERV-NAME*" "$Revision: 1.38 $" "hg17" "123"

The geneLayout function provides information on exon locations.

> print(geneLayout("546"))

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>CHROM</th>
<th>TRANSCRIPT.START</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;&quot;</td>
<td>&quot;RLF&quot;</td>
<td>&quot;chr1&quot;</td>
<td>&quot;40296154&quot;</td>
</tr>
</tbody>
</table>

CODINGSEQ.START TRANSCRIPT.END CODINGSEQ.END exon1.start
"156497437" "156497348" | "40296154" |
Information on all the genes catalogued in a certain chromosomal region can be obtained using `itemsInRange`.

```r
define(itemsInRange("genes", "chr1", "156400000", "156500000"))

[[1]]
  NAME       CHROM
  "CRP"      "chr1"
  PRODUCT    NSNPS
  "C-reactive protein, pentraxin-related" "101"
```

```r
$CHR
[1] "chr1"
```

```r
$START
[1] "156400000"
```

```r
$END
[1] "156500000"
```

```r
$COUNT
[1] "1"
```

```r
attr(.\"toolInfo\"")
SOURCE VERSION GENOME DBSNP
"*RPCSERV-NAME*" "$Revision: 1.38 $" "hg17" "123"
```


4.2 Obtaining information on SNPs

Suppose you want information on the SNP with dbSNP id rs25.
> print(SNPinfo("25"))

<table>
<thead>
<tr>
<th>DBSNPID</th>
<th>TSCID</th>
<th>CHROMOSOME</th>
<th>POSITION</th>
<th>ALLELES</th>
<th>ROLE</th>
<th>RELPOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;rs25&quot;</td>
<td>&quot; &quot;</td>
<td>&quot;chr7&quot;</td>
<td>&quot;11357382&quot;</td>
<td>&quot;A/G&quot;</td>
<td>&quot; &quot;</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AMINO AMINOPOS
" " "

attr("toolInfo")

SOURCE VERSION GENOME DBSNP
"*RPCSERV-NAME*" "$Revision: 1.38 $" "hg17" "123"

Suppose instead you want information on all the SNPs cataloged in a certain chromosomal region.

> ird <- itemsInRange("snps", "chr1", "156400000", "156500000")
> print(length(ird))

[1] 314

> print(ird[1:3])

[[1]]
<table>
<thead>
<tr>
<th>DBSNPID</th>
<th>TSCID</th>
<th>CHROMOSOME</th>
<th>POSITION</th>
<th>ALLELES</th>
<th>ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;rs2263016&quot;</td>
<td>&quot; &quot;</td>
<td>&quot;chr1&quot;</td>
<td>&quot;156400511&quot;</td>
<td>&quot;A/G&quot;</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>RELPOS</td>
<td>AMINO</td>
<td>AMINOPOS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot; &quot;</td>
<td>&quot; &quot;</td>
<td>&quot; &quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[[2]]
<table>
<thead>
<tr>
<th>DBSNPID</th>
<th>TSCID</th>
<th>CHROMOSOME</th>
<th>POSITION</th>
<th>ALLELES</th>
<th>ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;rs2263017&quot;</td>
<td>&quot; &quot;</td>
<td>&quot;chr1&quot;</td>
<td>&quot;156400521&quot;</td>
<td>&quot;A/C&quot;</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>RELPOS</td>
<td>AMINO</td>
<td>AMINOPOS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot; &quot;</td>
<td>&quot; &quot;</td>
<td>&quot; &quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[[3]]
<table>
<thead>
<tr>
<th>DBSNPID</th>
<th>TSCID</th>
<th>CHROMOSOME</th>
<th>POSITION</th>
<th>ALLELES</th>
<th>ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;rs7531018&quot;</td>
<td>&quot; &quot;</td>
<td>&quot;chr1&quot;</td>
<td>&quot;156400582&quot;</td>
<td>&quot;C/T&quot;</td>
<td>&quot; &quot;</td>
</tr>
<tr>
<td>RELPOS</td>
<td>AMINO</td>
<td>AMINOPOS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot; &quot;</td>
<td>&quot; &quot;</td>
<td>&quot; &quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that the start and end locations are supplied as strings. This is to avoid coercion to textual scientific notation.

Additional detail on the count of SNPs can be obtained more briefly:

> print(itemsInRange("countsnps", "chr1", "156400000", "156500000"))

5
total exonic nonsyn
310  7  0

To see all the SNPs associated with a given gene, use the `geneSNPs` function. This requires knowledge of the SNPper gene id, which can be obtained using `geneInfo`.

```r
> gs <- geneSNPs("546")
> print(length(gs))

[1] 164

> print(gs[1:3])

[[1]]

<table>
<thead>
<tr>
<th>DBSNPID</th>
<th>TSCID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;rs6679879&quot;</td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHROMOSOME</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;chr1&quot;</td>
<td>&quot;40286528&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALLELES</th>
<th>ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;C/G&quot;</td>
<td>&quot;Promoter&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RELPOS</th>
<th>AMINO</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;-9637&quot;</td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AMINOPOS</th>
<th>HUGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot; &quot;</td>
<td>&quot;RLF&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOCUSLINK</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;6018&quot;</td>
<td>&quot;rearranged L-myc fusion sequence&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MRNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;NM_012421&quot;</td>
</tr>
</tbody>
</table>

[[2]]

<table>
<thead>
<tr>
<th>DBSNPID</th>
<th>TSCID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;rs7550355&quot;</td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHROMOSOME</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;chr1&quot;</td>
<td>&quot;40287075&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALLELES</th>
<th>ROLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;C/T&quot;</td>
<td>&quot;Promoter&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RELPOS</th>
<th>AMINO</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;-9090&quot;</td>
<td>&quot; &quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AMINOPOS</th>
<th>HUGO</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot; &quot;</td>
<td>&quot;RLF&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOCUSLINK</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;6018&quot;</td>
<td>&quot;rearranged L-myc fusion sequence&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MRNA</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;NM_012421&quot;</td>
</tr>
</tbody>
</table>
Human chromosome 1 is approximately 300Mb, and 142,629 SNPs have been recorded as of dbSNP build 106, according to NCBI SNP/maplists/maplist-newmap.html on 13 Sep 03. Let’s see if these facilities can recover this sort of information. Counting the number of SNPs on a long chromosomal region seems to take a long time for SNPper, so we will break up the task.

> print(itemsInRange("countsnps", "chr1", "1", "100000"))

```
  total exonic nonsyn
    80   1   0
> system("sleep 2")
> print(itemsInRange("countsnps", "chr1", "100001", "200000"))

  total exonic nonsyn
    2   0   0
> system("sleep 2")
> print(itemsInRange("countsnps", "chr1", "200001", "300000"))

  total exonic nonsyn
    4   0   0
```
> system("sleep 2")

These runs complete in a reasonable amount of time. Here we will just look at the first 2Mb in intervals of .1Mb.

> starts <- as.character(as.integer(seq(1, 2000001, 1e+05)))
> ends <- as.character(as.integer(as.integer(starts) + 99999))
> out <- matrix(NA, nr = 20, nc = 3)
> for (i in 1:20) {
+    cat(i)
+    out[i, ] <- itemsInRange("countsnps", "chr1", starts[i],
+                            ends[i])
+    system("sleep 2")
+ }

> print(out)

```
[,1] [,2] [,3]
[1,]  80   1   0
[2,]   2   0   0
[3,]   4   0   0
[4,]   0   0   0
[5,]   4   0   0
[6,]  28   0   0
[7,] 126   0   0
[8,] 377  23   2
[9,] 405   7   1
[10,] 356  36  12
[11,] 370  33   5
[12,] 361  31   9
[13,] 334  51  15
[14,] 190  23   9
[15,] 251  30   7
[16,] 392  33   7
[17,] 154  12   5
[18,] 190  14   2
[19,] 209  10   1
[20,] 292   2   1
```