

# *RSNPper*: utilities for SNP data

VJ Carey [stvjc@channing.harvard.edu](mailto:stvjc@channing.harvard.edu)

April 25, 2007

## Contents

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

## 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

The core of this package is the XML-RPC service at CHIP accessible through the following URL stub:

```
> print(.SNPperBaseURL)
```

```
[1] "http://snpper.chip.org/bio/rpcserv/dummy?cmd="
```

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

```
> dput(useSNPper)
```

```
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
```

```
[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"))
```

SNPper Gene metadata:

There are 1 entries.

Basic information:

GENEID	NAME	CHROM	STRAND	PRODUCT	NSNPS
GENE 1440	CRP	chr1	-	C-reactive protein, pentraxin-related	101
TX.START	TX.END	CODSEQ.START	CODSEQ.END	LOCUSLINK	OMIM UNIGENE
GENE 156495525	156497437	156496388	156497348	1401 123260	Hs.76452
SWISSPROT	MRNAACC	PROTACC	REFSEQACC		
GENE P02741	NM_000567	NP_000558	NULL		

SNPper info:

SOURCE	VERSION	GENOME	DBSNP
[1,] "*RPCSERV-NAME*"	"\$Revision: 1.38 \$"	"hg17"	"123"

The `geneLayout` function provides information on exon locations.

```
> print(geneLayout("546"))
```

ID	NAME	CHROM	TRANSCRIPT.START
" "	"RLF"	"chr1"	"40296154"
CODINGSEQ.START	TRANSCRIPT.END	CODINGSEQ.END	exon1.start
"40296165"	"40375684"	"40375212"	"40296154"
exon1.end	exon2.start	exon2.end	exon3.start
"40296401"	"40323820"	"40323974"	"40325537"
exon3.end	exon4.start	exon4.end	exon5.start
"40325618"	"40330397"	"40330529"	"40337177"
exon5.end	exon6.start	exon6.end	exon7.start
"40337379"	"40357339"	"40357475"	"40366282"
exon7.end	exon8.start	exon8.end	
"40366423"	"40370557"	"40375684"	

attr(,"toolInfo")

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

Information on all the genes catalogued in a certain chromosomal region can be obtained using `itemsInRange`.

```
> print(itemsInRange("genes", "chr1", "156400000", "156500000"))
```

	NAME	CHROM
	"CRP"	"chr1"
	PRODUCT	NSNPS
	"C-reactive protein, pentraxin-related"	"101"

```
$CHR
[1] "chr1"

$START
[1] "156400000"

$END
[1] "156500000"

$COUNT
[1] "1"

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"))
```

SNPper SNP metadata:

	DBSNPID	CHROMOSOME	POSITION	ALLELES	VALIDATED
[1,]	"rs25"	"chr7"	"11357382"	"A/G"	"Y"

There are details on 4 populations  
and 1 connections to gene features

SNPper info:

	SOURCE	VERSION	GENOME	DBSNP
[1,]	"*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]]
      DBSNPID      TSCID CHROMOSOME  POSITION  ALLELES  ROLE
"rs2263016"      " "      "chr1" "156400511"  "A/G"    " "
      RELPOS      AMINO  AMINOPOS
      " "      " "      " "
```

```
[[2]]
      DBSNPID      TSCID CHROMOSOME  POSITION  ALLELES  ROLE
"rs2263017"      " "      "chr1" "156400521"  "A/C"    " "
      RELPOS      AMINO  AMINOPOS
      " "      " "      " "
```

```
[[3]]
      DBSNPID      TSCID CHROMOSOME  POSITION  ALLELES  ROLE
"rs7531018"      " "      "chr1" "156400582"  "C/T"    " "
      RELPOS      AMINO  AMINOPOS
      " "      " "      " "
```

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("countsnp", "chr1", "156400000", "156500000"))
```

```
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`.

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

```
[1] 164
```

```
> print(gs[1:3])
```

```
[[1]]
      DBSNPID      TSCID
"rs6679879"      " "
```

CHROMOSOME	POSITION
"chr1"	"40286528"
ALLELES	ROLE
"C/G"	"Promoter"
RELPOS	AMINO
"-9637"	" "
AMINOPOS	HUGO
" "	"RLF"
LOCUSLINK	NAME
"6018"	"rearranged L-myc fusion sequence"
MRNA	
"NM_012421"	

[[2]]

DBSNPID	TSCID
"rs7550355"	" "
CHROMOSOME	POSITION
"chr1"	"40287075"
ALLELES	ROLE
"C/T"	"Promoter"
RELPOS	AMINO
"-9090"	" "
AMINOPOS	HUGO
" "	"RLF"
LOCUSLINK	NAME
"6018"	"rearranged L-myc fusion sequence"
MRNA	
"NM_012421"	

[[3]]

DBSNPID	TSCID
"rs12096261"	" "
CHROMOSOME	POSITION
"chr1"	"40288041"
ALLELES	ROLE
"G/T"	"Promoter"
RELPOS	AMINO
"-8124"	" "
AMINOPOS	HUGO
" "	"RLF"
LOCUSLINK	NAME
"6018"	"rearranged L-myc fusion sequence"

MRNA  
"NM\_012421"

## 5 Application: SNP density on chr 1

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")
+ }
```

1234567891011121314151617181920

```
> 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