

MyVariant.info R Client

Adam Mark, Chunlei Wu

May 3, 2016

Contents

1	Overview	1
2	Variant Annotation Service	1
2.1	Obtaining HGVS IDs from a VCF file.	1
2.2	<code>getVariant</code>	2
2.3	<code>getVariants</code>	3
3	Variant Query Service	3
3.1	<code>queryVariant</code>	3
3.2	<code>queryVariants</code>	4
4	References	4

1 Overview

MyVariant.Info is a simple-to-use REST web service to query/retrieve genetic variant annotation from an aggregation of variant annotation resources. *myvariant* is an easy-to-use R wrapper to access MyVariant.Info services and explore variant annotations.

2 Variant Annotation Service

2.1 Obtaining HGVS IDs from a VCF file.

- Use `readVcf` from the `VariantAnnotation` package to read a Vcf file in. The Vcf object can then be passed to `formatHgvs` to retrieve HGVS IDs. HGVS IDs are based on the GRCh38/hg19 reference genome. Support for hg38 is coming soon.

```
> file.path <- system.file("extdata", "dbsnp_mini.vcf", package="myvariant")
> vcf <- readVcf(file.path, genome="hg19")
> rowRanges(vcf)
```

GRanges object with 240 ranges and 5 metadata columns:

seqnames	ranges	strand	paramRangeID	REF
<Rle>	<IRanges>	<Rle>	<factor>	<DNAStringSet>
rs376643643	1 [10019, 10020]	*	<NA>	TA
rs373328635	1 [10055, 10055]	*	<NA>	T
rs62651026	1 [10108, 10108]	*	<NA>	C
rs376007522	1 [10109, 10109]	*	<NA>	A
rs368469931	1 [10139, 10139]	*	<NA>	A
...
rs544020171	1 [17654, 17654]	*	<NA>	T
rs563880190	1 [17694, 17694]	*	<NA>	C
rs574335987	1 [17695, 17695]	*	<NA>	G
rs374995955	1 [17697, 17697]	*	<NA>	G
rs543363182	1 [17709, 17709]	*	<NA>	T

ALT	QUAL	FILTER
<DNAStringSetList>	<numeric>	<character>
rs376643643	T	<NA>
rs373328635	TA	<NA>
rs62651026	T	<NA>
rs376007522	T	<NA>
rs368469931	T	<NA>
...
rs544020171	C	<NA>
rs563880190	T	<NA>
rs574335987	A	<NA>
rs374995955	C	<NA>
rs543363182	G	<NA>

seqinfo: 1 sequence from hg19 genome; no seqlengths

- You can then use `formatHgvs` to extract HGVS IDs from the Vcf object.

```
> hgvs <- formatHgvs(vcf, variant_type="snp")
> head(hgvs)
```

```
[1] "chr1:g.10108C>T" "chr1:g.10109A>T" "chr1:g.10139A>T" "chr1:g.10150C>T"
[5] "chr1:g.10177A>C" "chr1:g.10180T>C"
```

2.2 getVariant

- Use `getVariant`, the wrapper for GET query of `"/v1/variant/<hgvsid>"` service, to return the variant object for the given HGVS id.

```
> variant <- getVariant("chr1:g.35367G>A")
> variant[[1]]$dbnsfp$genename
```

```
NULL
```

```
> variant[[1]]$cadd$phred
```

```
[1] 3.726
```

2.3 getVariants

- Use `getVariants`, the wrapper for POST query of `"/v1/variant"` service, to return the list of variant objects for the given character vector of HGVS ids.

```
> getVariants(c("chr1:g.35367G>A", "chr16:g.28883241A>G"),
+             fields="cadd.consequence")
```

```
DataFrame with 2 rows and 5 columns
```

	X_id	X_score	query	cadd._license	cadd.consequence
	<character>	<numeric>	<character>	<character>	<character>
1	chr1:g.35367G>A	1	chr1:g.35367G>A	http://goo.gl/bkpNhq	NONCODING_CHANGE
2	chr16:g.28883241A>G	1	chr16:g.28883241A>G	http://goo.gl/bkpNhq	NON_SYNONYMOUS

3 Variant Query Service

3.1 queryVariant

- `queryVariant` is a wrapper for GET query of `"/v1/query?q=<query>"` service, to return the query result. This function accepts wild card input terms and allows you to query for variants that contain a specific annotation. For example, the following query searches for the CADD phred score and consequence for all variants whose genename (dbNSFP) is MLL2.

```
> queryVariant(q="dbnsfp.genename:MLL2", fields=c("cadd.phred", "cadd.consequence"))
```

```
$hits
```

```
list()
```

```
$max_score
```

```
NULL
```

```
$took
```

```
[1] 5
```

```
$total
```

```
[1] 0
```

- You can also use `queryVariant` to retrieve all annotations that map to a specific rsID.

```
> queryVariant(q="rs58991260", fields="dbSNP.flags")$hits
      _id  _score  flags
1 chr1:g.218631822G>A 16.93929 ASP, G5, G5A, GNO, KGPhase1, KGPhase3, SLO
```

3.2 queryVariants

- `queryVariants` is a wrapper for POST query of `"/v1/query?q=<query>"` service, to return the query result. Query terms include any available field as long as scopes are defined. The following example reads the dbSNP rsIDs from a VCF and queries for all fields. The returned `DataFrame` can then be easily subsetted to include, for example, those that have not been documented in the Welllderly study.

```
> rsids <- paste("rs", info(vcf)$RS, sep="")
> res <- queryVariants(q=rsids, scopes="dbSNP.rsid", fields="all")
```

Finished

Pass `returnall=TRUE` to return lists of duplicate or missing query terms.

```
> subset(res, !is.na(wellderly.vartype))$query
 [1] "rs367896724" "rs145427775" "rs147093981" "rs56289060" "rs55998931" "rs199606420"
 [7] "rs58108140" "rs62635284" "rs62635286" "rs200579949" "rs531730856" "rs180734498"
[13] "rs527952245" "rs201696125" "rs546169444" "rs201055865" "rs62635298" "rs79585140"
[19] "rs199856693" "rs201855936" "rs201045431" "rs71252250" "rs200030104" "rs78601809"
[25] "rs62636497" "rs62636497" "rs201635489" "rs533630043" "rs2691315" "rs113442401"
[31] "rs572465511" "rs372319358" "rs200358166" "rs11489794" "rs113141985" "rs62636498"
[37] "rs148220436" "rs141130360" "rs150723783" "rs201459529" "rs199676946" "rs62636367"
[43] "rs62636368" "rs200205172" "rs199745162" "rs200658479" "rs201833382" "rs199740902"
[49] "rs200978805" "rs201535981" "rs192890528" "rs200046632" "rs374995955"
```

4 References

MyVariant.info help@myvariant.info