

AnnotationHub

AnnotationHub包提供了一个访问Bioconductor AnnotationHub网站资源的客户端，该网站存放来自标准机构的基因组资源文件，例如UCSC, Ensembl等的vcf,bed,wig文件资源。

```
library(AnnotationHub)
```

```
ah <- AnnotationHub()
```

使用以下函数查询AnnotationHub的信息(x为AnnotationHub objects)

hubCache(x):返回本地AnnotationHub cache的系统存放位置

removeCache(x): 删除本地AnnotationHub数据以及相关资源

hubUrl(x): 返回在线hub的url

names(x): 返回hub记录的名称，AnnotationHub unique identifiers, AH12345

fileName(x): 返回本地cache的hub记录的文件路径

mcols(x): 返回记录的metadata列信息

dbfile(x): 返回SQLite 数据的全路径

使用以下函数访问AnnotationHub object

query(x, pattern, ignore.case=T, pattern.op='&'): 返回匹配pattern的AnnotationHub 对象的子集

使用 '[' 下载records, 首次使用时，对应的文件和其他hub资源回从网上下载到local cache。随后就可以快速使用这些local cache文件。假如想再次下载，而不是用cache文件，则需要加上参数'force=TRUE'。

```
res <- ah[["AH67442"]]
```

subset(x, subset): 返回匹配subset表达式的子集

display(x): 网页展示hub记录

keys: 返回包含在AnnotationDb对象中数据的关键字，搭配select使用

columns: 显示AnnotationDb对象所能返回的数据种类

keytypes: 显示可以供select， keys和keytype能选择的关键字类型

loadDb: 装载离线下载的annotation package

```
head(keys(orgdb, keytype="SYMBOL"))
```

select(x, keys, columns, keytype, ...): 根据keys， columns， keytype等参数以数据框的格式返回AnnotationDb对象所含内容，而mapIds返回向量

```
trial.sample <- sample(keys(orgdb, keytype="ENTREZID"))
```

```
select(orgdb,keys=trial.sample,column=( "SYMBOL" ),keytype="ENTREZID")
```

```
mapIds(orgdb,keys=trial.sample,column=( "SYMBOL" ),keytype="ENTREZID")
```

```
ah_kpc <- query(ah, c("klebsiella"))
```

```
ah_kpc[1]
```

```
ah_kpc$sourceurl
```

```
ah_kpc$description
```

```
ah_kpc$title
```

mapIds(x, keys, columns, keytype,...), 同**select**用法, 常用于返回字符向量

- 针对无法下载的ah, 例如klebsiella_res <- ah[["AH67442"]], 可选择百度网盘下载后, 根据错误提示, 将文件直接cp到提示文件及名称

```
cp org.Klebsiella_pneumoniae.eg.sqlite /Users/carlos/.AnnotationHub/74188
```

```
> klebsiella_res <- ah[["AH67442"]]
downloading 0 resources
loading from cache
  ‘/Users/carlos//.AnnotationHub/74188’
Loading required package: AnnotationDbi
Loading required package: stats4
Loading required package: Biobase
Welcome to Bioconductor
```

Vignettes contain introductory material; view with

```
> head(select(klebsiella_res,keys=keys(klebsiella_res),keytype="ENTREZID",columns=c("ENTREZID","GENE
NAME","GID","GO","ONTOLOGY","SYMBOL")))
+ )
'select()' returned 1:many mapping between keys and columns
  ENTREZID      GENENAME      GID      GO ONTOLOGY SYMBOL
1  1238784 klebicin B lysis 1238784 GO:0019867      CC      kbl
2  1238784 klebicin B lysis 1238784 GO:0019835      BP      kbl
3  1238784 klebicin B lysis 1238784 GO:0009405      BP      kbl
4  1238785          kbi 1238785 GO:0015643      MF      kbi
5  1238785          kbi 1238785 GO:0030153      BP      kbi
6  1238786 klebicin B      1238786 GO:0004519      MF      kba
```

GenomicFeatures

GenomicFeatures包使用TxDb对象存储转录组数据, 包括UTRs,CDSS,外显子信息等。所有TxDb对象都以SQLite database备份, 包含基因组的位置, pre-processed mRNA转录本, 外显子, 蛋白编码序列之间关系以及它们相关基因识别符。

loadDb(): 直接从合适的.sqlite数据文件读取对象

```
txdb <- loadDb(samplefile.sqlite)
```

或者

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

```
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
```

seqlevels(txdb): 查看染色体水平

TxDb对象inherit from AnnotationDb对象(Just as the ChipDb and OrgDb objects), 可以使用columns, keytypes, keys,和select函数操作TxDb对象。

transcripts(txdb): 返回所包含转录本位置区间信息

exons(txdb): 返回对应外显子区间信息

transcriptsBy(txdb, by="gene"): 查看根据genes分组的转录本信息

exonsBy(txdb, by="tx"): 查看根据转录本分组的基因信息

intronsByTranscript(txdb), fiveUTRsByTranscript(txdb), threeUTRsByTranscript(txdb)同上功能

提取序列信息

```
tx_seq <- extractTranscriptSeqs(Hsapiens, TxDb.Hsapiens.UCSC.hg19.knownGene,  
use.names=T)
```

转录序列成蛋白序列

```
translate(extractTranscriptSeqs(Hsapiens,txdb))
```

创建新的TxDb对象

GenomicFeatures包可使用从UCSC Genome Bioinformatics或BioMart下载的数据创建TxDb对象

```
supportedUCSCTables(genom="hg19")
```

查看UCSC基因组hg19所包含的tables信息

```
txdb_from_hg19 <- makeTxDbFromUCSC(genome="hg19",tablename="knownGene")
```

使用makeTxDbFromBiomart来从BioMart获得指定mart和data set的数据

```
txdb_from_biomart <- makeTxDbFromBiomart()
```

```
makeTxDbFromBiomart(biomart="ENSEMBL_MART_ENSEMBL",dataset="hsapiens_gene_ensembl", ...)
```

使用makeTxDbFromEnsembl,

```
makeTxDbFromEnsembl(organism="Homo sapiens",release=NA, ...)
```

同样可以使用makeTxDbFromGFF, makeTxDbFromGRanges来创建TxDb对象。

保存和载入TxDb对象,由于TxDb对象以SQLite数据备份, 因此保存也为该格式

一旦创建了TxDb对象, 可以保存本地, 避免反复创建

```
saveDb(txdb, file="fileName.sqlite")
```

```
txdb <- loadDb("fileName.sqlite")
```

BSgenome & VariantAnnotation

BSgenome包存储基因组信息及snp信息

`available.genomes()`: 查看当前基因组信息

选择加载hg19

```
library(BSgenome.Hsapiens.UCSC.hg19)
```

定位genes内以及周围的变异

```
locateVariants(query, subject, region, ...)
```

query: 包含变异的IntegerRanges, Granges 或 VCF对象

subject: 用于注释的TxDb或'GRangesList'对象

region: 8种变异类型中的一个'CodingVariants', 'IntronVariants', 'FiveUTRVariants', 'ThreeUTRVariants', 'IntergenicVariants', 'SpliceSiteVariants', 'PromoterVariants', 'AllVariants'.

```
library(VariantAnnotation)
```

```
fl <- system.file("extdata", "chr22.vcf.gz", package="VariantAnnotation")
```

```
vcf <- readVcf(fl, "hg19")
```

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

```
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
```

```
seqlevels(vcf) <- "chr22"
```

```
rd <- rowRanges(vcf)
```

```
loc <- locateVariants(rd, txdb, CodingVariants())
```

预测氨基酸密码子改变

针对非同义变异, predictCoding计算氨基酸密码子的改变, 只有来自subject中coding区域的query序列范围的改变才会预测, 同时参考序列来自于BSgenome或有seqSource指定的fasta文件

```
IntergenicVariants(upstream = 1e+06L, downstream = 1e+06L, idType=c("gene", "tx"))
```

```
predictCoding(query, subject, seqSource, varAllele, ..., ignore.strand=FALSE)
```

query: GRanges或vcf文件, 如果是GRanges文件, 就需要指定varAllele参数, 如果是vcf文件, 那么alternate alleles就从alt(vcf)中获得

subject: 用于注释的TxDb文件

seqSource: 一个 BSgenom instance或FaFile用于序列的提取

varAllele: 包含变异等位基因的DNAStringSet

```
library(BSgenome.Hsapiens.UCSC.hg19)
```

```
conding <- predictCoding(vcf, txdb, seqSource=Hsapiens)
```

或

```
rd <- rowRanges(vcf)

altallele <- alte(vcf)

eltROWS <- elementNROWS(altallele)

rd_exp <- rep(rd, eltROWS)

predictCoding(rd_exp , txdb, Hsapiens)
```

使用filterVcf函数过滤变异，略！！

biomaRt

```
library(biomaRt)
```

第一步检查BioMart网络服务

```
> listMarts()
      biomart      version
1 ENSEMBL_MART_ENSEMBL      Ensembl Genes 96
2  ENSEMBL_MART_MOUSE      Mouse strains 96
3    ENSEMBL_MART_SNP      Ensembl Variation 96
4 ENSEMBL_MART_FUNCGEN      Ensembl Regulation 96
```

useMart()函数指定需要连接的BioMart数据库，名称由listMarts()返回

```
ensembl=useMart("ensembl")
```

BioMart数据库能包含多个数据集，对于Ensembl，每个物种都是一个不同的数据集，使用listDatabases()展示可用的数据集

```
datasets <- listDatasets(ensembl)
```

```
> head(datasets)
      dataset      description
1 abrachyrhynchus_gene_ensembl Pink-footed goose genes (ASM259213v1)
2  acalliptera_gene_ensembl      Eastern happy genes (fAstCal1.2)
3  acarolinensis_gene_ensembl      Anole lizard genes (AnoCar2.0)
4  acitrinellus_gene_ensembl      Midas cichlid genes (Midas_v5)
5   ahaastii_gene_ensembl      Great spotted kiwi genes (aptHaa1)
6  amelanoleuca_gene_ensembl      Panda genes (ailMel1)
      version
1 ASM259213v1
2 fAstCal1.2
3 AnoCar2.0
4 Midas_v5
5 aptHaa1
6 ailMel1
```

使用useDataset选择数据集，更新Mart对象

```
ensembl = useDataset("hsapiens_gene_ensembl", mart=ensembl)
```

或者直接使用useMart选择更新

```
ensembl = useMart("ensembl", dataset="hsapines_gene_ensembl")
```

构建biomaRt查询

getBM()函数拥有三个参数：filters, attributes和values, 是biomaRt的主要函数

filters定义了query的限制范围：

```
filters <- listFilters(ensembl)
```

```
> filters[1:5,]  
      name      description  
1 chromosome_name Chromosome/scaffold name  
2      start      Start  
3      end      End  
4  band_start  Band Start  
5  band_end    Band End  
>
```

attributes定义了query想要查询的范围：

```
attributes <- listAttributes(ensembl)
```

```
> attributes[1:5,]  
      name      description      page  
1  ensembl_gene_id      Gene stable ID feature_page  
2  ensembl_gene_id_version      Gene stable ID version feature_page  
3  ensembl_transcript_id      Transcript stable ID feature_page  
4  ensembl_transcript_id_version      Transcript stable ID version feature_page  
5  ensembl_peptide_id      Protein stable ID feature_page
```

getBM()共含有4个主要参数：

- attributes, 想要获得的属性向量, 等同于query的输出
- filters, 需要过滤掉的值向量的向量, 对应values信息, 且要一一对应
- values, filters的值的向量, 可以为多重filters, 为list
- mart, Mart类别对象, 使用useMart()函数构建

另外一些函数如getGenes()和getSequence()函数同样可以实现getBM()功能, 对应指定的属性和过滤条件。

```
getBM(attributes, filters = "", values = "", mart, curl = NULL, checkFilters =  
TRUE, verbose = FALSE, uniqueRows = TRUE, bmHeader = FALSE, quote = "\\")
```

例如, 在affymetrix中查询对应EntrezGene id

```
affyids <- c("202763_at", "209310_s_at", "207500_at")
```

```
getBM(attributes=c("affy_hg_u133_plus_2", "entrezgene"),  
filters="affy_hg_u133_plus_2", values=affyids, mart=ensembl)
```

```
> getBM(attributes=c("affy_hg_u133_plus_2", "entrezgene"), filters="affy_hg_u133_plus_2", values=affyids,  
mart=ensembl)  
  affy_hg_u133_plus_2  entrezgene  
1      202763_at      836  
2      209310_s_at      837  
3      207500_at      838
```


针对listDatasets(), listAttributes(), listFilters()返回的长的结果内容, 可以使用searchDatasets(), searchAttributes(), searchFilters()对应搜索匹配指定条目的结果, pattern为正则表达式:

```
searchDatasets(mart=ensembl, pattern="hsapiens")
```

```
> searchDatasets(ensembl, pattern="hsapiens")
      dataset      description      version
73 hsapiens_gene_ensembl Human genes (GRCh38.p12) GRCh38.p12
```

```
searchAttributes(mart=ensembl, pattern="hgnc")
```

```
> searchAttributes(mart=ensembl, pattern='hgnc')
      name      description      page
59      hgnc_id      HGNC ID feature_page
60      hgnc_symbol      HGNC symbol feature_page
61 hgnc_trans_name HGNC transcript name ID feature_page
```

使用正则表达实现具体搜索

```
searchFilters(mart=ensembl, pattern="ensembl.*id")
```

```
> searchFilters(mart=ensembl, pattern="ensembl.*id")
      name
54      ensembl_gene_id
55      ensembl_gene_id_version
56      ensembl_transcript_id
57 ensembl_transcript_id_version
58      ensembl_peptide_id
59      ensembl_peptide_id_version
60      ensembl_exon_id

      description
54      Gene stable ID(s) [e.g. ENSG000000000003]
55      Gene stable ID(s) with version [e.g. ENSG000000000003.14]
56      Transcript stable ID(s) [e.g. ENST000000000233]
57 Transcript stable ID(s) with version [e.g. ENST000000000233.10]
58      Protein stable ID(s) [e.g. ENSP000000000233]
59      Protein stable ID(s) with version [e.g. ENSP000000000233.5]
60      Exon ID(s) [e.g. ENSE00000327880]
```

然后根据提示, 具体搜索:

```
> getBM(attributes=c("entrezgene"), filters="ensembl_transcript_id_version",
+ values=c("ENST00000577249.1"), mart=ensembl)
      entrezgene
1      990
> getBM(attributes=c("ensembl_transcript_id_version", "entrezgene"), filters="ensembl_transcript_id_version", values=c("ENST00000577249.1"), mart=ensembl)
      ensembl_transcript_id_version entrezgene
1      ENST00000577249.1      990
```

示例

1. 使用HUGO和对应基因的染色体位置注释affymatrix identifiers

```
affyids <- c("202763_at", "209310_s_at", "207500_at")
```

```
getBM(attributes=c("affy_hg_u133_plus_2", "hgnc_symbol", "chromosome_name",
"start_position", "end_position", "band"), filters="affy_hg_u133_plus_2"),
values=affyids, mart=ensembl)
```

```
> getBM(attributes=c("affy_hg_u133_plus_2", "hgnc_symbol", "chromosome_name", "start_position", "end_pos
ition", "band"), filters="affy_hg_u133_plus_2", values=affyids, mart=ensembl)
  affy_hg_u133_plus_2 hgnc_symbol chromosome_name start_position end_position
1          202763_at      CASP3              4       184627696    184649509
2          209310_s_at      CASP4             11       104942866    104969366
3          207500_at      CASP5             11       104994235    105023168
  band
1 q35.1
2 q22.3
3 q22.3
```

2. 使用GO注释EntrezGene ids

```
entrez <- c("673", "837")
```

```
goids <- getBM(attributes=c("entrezgene", "go_id"), filters="entrezgene",
values=entrez, mart=ensembl)
```

```
> head(goids)
  entrezgene      go_id
1         673 GO:0005524
2         673 GO:0007165
3         673 GO:0006468
4         673 GO:0035556
5         673 GO:0004672
6         673 GO:0046872
```

3. 查询位于17/20/Y染色体上属于指定GO条目的HUGO gene名称

```
go <- ("GO:0051330", "GO:0000080", "GO:0000114", "GO:0000082")
```

```
chrom <- c(17,20,"Y")
```

```
getBM(attributes="hgnc_symbol", filters=c("go", "chromosome_name"),
values=list(go, chrom), mart=ensembl)
```

```
> getBM(attributes = "hgnc_symbol", filters=c("go", "chromosome_name"), values=list(go, chrom), mart=ense
mbl)
  hgnc_symbol
1      RPS6KB1
2      RPA1
3      CDK3
4      CDC6
5      MCM8
6      CRLF3
```

或者，去除染色体的限制，增加attributes内容输出,filter和values一一对应！

```
> getBM(attributes =c("hgnc_symbol", "chromosome_name"), filters=c("go"), values=go, mart=ensembl)
  hgnc_symbol      chromosome_name
1      POLE3              9
2      POLA2             11
3      MNAT1             14
4      RPS6KB1            17
5      RPA1              17
6      USP29             19
```


4. 使用INTERPRO蛋白结构域注释refseq ids

```
refseqids <- c("NM_005359", "NM_000546")
```

```
ipro <- getBM(attributes=c("refseq_mrna", "interpro",
```

```
"interpro_description"), filters="refseq_mrna", values=refseqids, mart=ensembl)
```

```
> ipro
  refseq_mrna interpro
1 NM_000546 IPR002117
2 NM_000546 IPR008967
3 NM_000546 IPR010991
4 NM_000546 IPR011615
5 NM_000546 IPR012346
6 NM_000546 IPR013872
7 NM_000546 IPR036674
8 NM_005359 IPR001132
9 NM_005359 IPR003619
10 NM_005359 IPR008984
11 NM_005359 IPR013019
12 NM_005359 IPR013790
13 NM_005359 IPR017855
14 NM_005359 IPR036578

                                interpro_description
1                                p53 tumour suppressor family
2 p53-like transcription factor, DNA-binding
3                                p53, tetramerisation domain
```

5. 查询染色体固定区间的gene信息

```
getBM(attributes = c('affy_hg_u133_plus_2', 'ensembl_gene_id')
,
  filters = c('chromosome_name', 'start', 'end'),
  values = list(16, 1100000, 1250000),
  mart = ensembl)
```

filters信息和values信息一一对应

6. 查询'MAP kinase activity' 相关GO term的所有HUGO基因名称

```
getBM(attributes = c('entrezgene', 'hgnc_symbol'),
  filters = 'go',
  values = 'GO:0004707',
  mart = ensembl)
```

7. 针对指定EntrezGene ids查询启动子上游100bp序列

可直接使用getSequence()函数获取序列信息

```
getSequence(chromosome, start, end, id, type, seqType, upstream, downstream, mart,
verbose = FALSE)
```

```
entrez=c("673","7157","837")
getSequence(id = entrez,
            type="entrezgene",
            seqType="coding_gene_flank",
            upstream=100,
            mart=ensembl)
```

8. 查询位于3号染色体指定位置的所有5' utr序列

```
> utr5 <- getSequence(chromosome=3,start=185514033,end=185535839,
+ type="entrezgene",
+ seqType="5utr",
+ mart=ensembl)
> utr5

           5utr                                     S
1                                     S
sequence unavailable
2 ACCACACCTCTGAGTCGTCTGAGCTCACTGTGAGCAAAATCCCACAGTGAAACTCTTAAGCCTCTGCGAAGTAAATCATTCTTGTGAATGTGACACA
CGATCTCTCCAGTTTCCAT
3                                     ATTCTTGTGAATGTGACACA
CGATCTCTCCAGTTTCCAT
4                                     TGAGCAAAATCCCACAGTGAAACTCTTAAGCCTCTGCGAAGTAAATCATTCTTGTGAATGTGACACA
CGATCTCTCCAGTTTCCAT
   entrezgene
1      200870
```

9. 查询指定entrezgene的蛋白序列

```
protein = getSequence(id=c(100, 5728),
                      type="entrezgene",
                      seqType="peptide",
                      mart=ensembl)
```

```
protein
```

```
##
peptide
## 1
Sequence unavailable
## 2
MAQTPAFDKPKVELHVHLDGSIKPETILYYGRRRGIALPANTAEGLLNVIGMDKPLTLPDF
LAKFDYYMPAIARL*
## 3
MAQTPAFDKPKVELHVHLDGSTKPETILYYGRRRGIALPANTAEGLLNVTGMDKPLTLPDF
```

10. 查询指定位置snp信息

首先使用不同的BioMart数据库

```
snpmart <- useMart(biomart="ENSEMBL_MART_SNP", dataset="hsapiens_snp")
```

```
> getBM(attributes=c("refsnp_id","allele","chrom_start","chrom_strand"),
+ filters=c("chr_name","start","end"),
+ values=list(8,148350,148612),
+ mart=snpmart)
```


	refsnp_id	allele	chrom_start	chrom_strand
1	rs1450830176	G/C	148350	1
2	rs1360310185	C/T	148352	1
3	rs1434776028	A/T	148353	1
4	rs1413161474	C/T	148356	1
5	rs1410590268	A/G	148365	1
6	rs1193735780	T/A/C	148368	1

AnnotationForge

构建organism package可直接使用函数makeOrgPackageFromNCBI()或makeOrgPackage()。若package可使用NCBI Taxonomy ID通过NCBI构建，该命令根据tax_id从NCBI搜索相应gene records，因此选择正确的NCBI taxonomy ID很重要。例如，构建zebrafinch organism package:

```
library(AnnotationForge)
```

```
makeOrgPackageFromNCBI(version = "0.1",
                        author = "Some One <so@someplace.org>",
                        maintainer = "Some One <so@someplace.org>",
                        outputDir = ".",
                        tax_id = "59729",
                        genus = "Taeniopygia",
                        species = "guttata")
```



若不能从NCBI处获得所有构建信息，可从其他资源使用makeOrgPackage()函数构建，该函数不依赖NCBI构建。该函数使用data.frame的field name构建为相应的数据库信息，同时可通过columns()和keytypes()查看所构建数据库。对所添加数据类型没有限制，但是data.frame的第一列对应gene ID，命名为"GID"。

当data.frame包含GO information时，使用goTable方法指明。若使用该选项，makeOrgPackage()将1) remove IDs that are too new, 2) create a second table to also represent the GOALL, EVIDENCEALL, ONTOLOGYALL fields for the select method。此种情况(使用goTable)，需遵守严格规则，该data.frame仅需包含3列，对应为gene id, GO id 和 evidence codes，对应名称为"GID", "GO"和"EVIDENCE":

```

> head(fGO)
      GID      GO EVIDENCE
1 100190152 GO:0008152      IEA
2 100190152 GO:0016310      IEA
3 100190152 GO:0006222      IEA
4 100190152 GO:0015937      IEA
5 100190152 GO:0000166      IEA
6 100190152 GO:0005524      IEA
> head(fSym)
      GID SYMBOL      GENENAME
1 751582  SNCA synuclein, alpha (non A4 component of amyloid precursor)
2 751583  NCALD      neurocalcin delta
3 751584  BDNF      brain-derived neurotrophic factor
4 751585  CREB1      cAMP responsive element binding protein 1
5 751586  MTNR1A      melatonin receptor 1A
6 751588  MTNR1B      melatonin receptor 1B
> head(fChr)
      GID CHROMOSOME
1 751582          4
2 751583          2
3 751584          5
4 751585          7
5 751586          4
6 751588          1

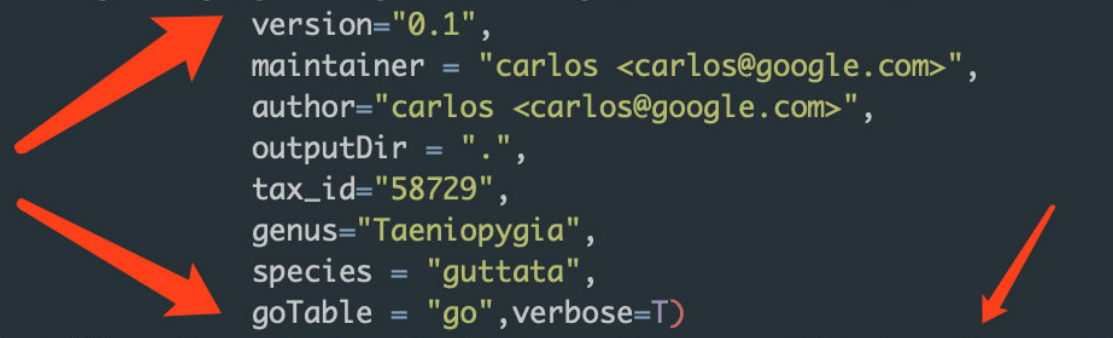
```

以上三个data.frame先分别删除没有信息列

```

1 makeOrgPackage(go=fGO, gene_info=fSym, chromosome=fChr,
2               version="0.1",
3               maintainer = "carlos <carlos@google.com>",
4               author="carlos <carlos@google.com>",
5               outputDir = ".",
6               tax_id="58729",
7               genus="Taeniopygia",
8               species = "guttata",
9               goTable = "go", verbose=T)
10 install.packages("./org.Tguttata.eg.db", repos=NULL, type="source")

```



本地安装package需指明type为"source"

其中go=fGO/gene_info=fSym/chromosome=fChr 满足：1) 数据框，2) 第一列名为"GID"，3) 不含 rownames信息；4) 各data.frame的列名称将成为columns()和keytypes()信息，供select()函数使用。