An Introduction to *GenomeInfoDb*

Martin Morgan, Herve Pages, Marc Carlson, Sonali Arora

Modified: 17 January, 2014. Compiled: January 25, 2016

Contents

1 Introduction

The *GenomeInfoDb* provides an interface to access seqlevelsStyles (such as UCSC, NCBI, Ensembl) and their supported mappings for organisms. For instance, for Homo sapiens, seqlevelsStyle "UCSC" maps to "chr1", "chr2", ..., "chrX","chrY". The section below introduces these functions with examples.

2 Functionality for all existing organisms

2.1 genomeStyles

The `genomeStyles` lists out for each organism, the seqlevelsStyles and their mappings.

```
seqmap <- genomeStyles()
head(seqmap,n=2)

## $Arabidopsis_thaliana
##   circular  auto   sex NCBI TAIR9 Ensembl
## 1    FALSE  TRUE FALSE    1  Chr1       1
## 2    FALSE  TRUE FALSE    2  Chr2       2
## 3    FALSE  TRUE FALSE    3  Chr3       3
## 4    FALSE  TRUE FALSE    4  Chr4       4
## 5    FALSE  TRUE FALSE    5  Chr5       5
## 6     TRUE FALSE FALSE   MT  ChrM      Mt
## 7     TRUE FALSE  TRUE Pltd  ChrC      Pt
##
## $Caenorhabditis_elegans
##   circular  auto   sex NCBI   UCSC Ensembl
## 1    FALSE  TRUE FALSE    I   chrI       I
## 2    FALSE  TRUE FALSE   II  chrII      II
## 3    FALSE  TRUE FALSE  III chrIII     III
## 4    FALSE  TRUE FALSE   IV  chrIV      IV
## 5    FALSE  TRUE FALSE    V   chrV       V
## 6    FALSE FALSE  TRUE    X   chrX       X
## 7     TRUE  TRUE FALSE   MT   chrM   MtDNA
```

Oragnism's supported by GenomeInfoDb can be found by :

```
names(genomeStyles())

##  [1] "Arabidopsis_thaliana"     "Caenorhabditis_elegans"   "Canis_familiaris"
##  [4] "Cyanidioschyzon_merolae"  "Drosophila_melanogaster"  "Homo_sapiens"
##  [7] "Mus_musculus"             "Oryza_sativa"             "Populus_trichocarpa"
## [10] "Rattus_norvegicus"        "Saccharomyces_cerevisiae" "Zea_mays"
```

If one knows the organism one is interested in, then we can directly access the information for the given organism along. Each function accepts an argument called species which as "genus species", the default is "Homo sapiens". In the following example we list out only the first five entries returned by the code snippet.

```
head(genomeStyles("Homo_sapiens"),5)

##   circular auto   sex NCBI UCSC dbSNP Ensembl
## 1    FALSE TRUE FALSE    1 chr1   ch1       1
## 2    FALSE TRUE FALSE    2 chr2   ch2       2
## 3    FALSE TRUE FALSE    3 chr3   ch3       3
## 4    FALSE TRUE FALSE    4 chr4   ch4       4
## 5    FALSE TRUE FALSE    5 chr5   ch5       5
```

We can also check if a given style is supported by GenomeInfoDb for a given species. For example, if we want to know if "UCSC" mapping is supported for "Homo sapiens" we can ask :

```
"UCSC" %in% names(genomeStyles("Homo_sapiens"))

## [1] TRUE
```

2.2 extractSeqlevels

We can also extract the desired seqlevelsStyle from a given organism using the `extractSeqlevels`

```
extractSeqlevels(species="Arabidopsis_thaliana", style="NCBI")

## [1] "1"    "2"    "3"    "4"    "5"    "MT"   "Pltd"
```

2.3 extractSeqlevelsByGroup

We can also extract the desired seqlevelsStyle from a given organism based on a group (Group - 'auto' denotes autosomes, 'circular' denotes circular chromosomes and 'sex' denotes sex chromosomes; the default is all chromosomes are returned).

```
extractSeqlevelsByGroup(species="Arabidopsis_thaliana", style="NCBI",
                         group="auto")

## [1] "1" "2" "3" "4" "5"
```

2.4 seqlevelsStyle

We can find the seqname Style for a given character vector by using the `seqlevelsStyle`

```
seqlevelsStyle(paste0("chr",c(1:30)))

## [1] "UCSC"

seqlevelsStyle(c("2L","2R","X","Xhet"))

## [1] "NCBI"
```

2.5 seqlevelsInGroup

We can also subset a given character vector containing seqnames using the `seqlevelsInGroup`. We currently support 3 groups: 'auto' for autosomes, 'sex' for allosomes/sex chromosomes and circular for 'circular' chromosomes. The user can also provide the style and species they are working with. In the following examples, we extract the sex, auto and circular chromosomes for Homo sapiens :

```
newchr <- paste0("chr",c(1:22,"X","Y","M","1_gl000192_random","4_ctg9_hap1"))
seqlevelsInGroup(newchr, group="sex")

## [1] "chrX" "chrY"

seqlevelsInGroup(newchr, group="auto")

##  [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"  "chr7"  "chr8"  "chr9"  "chr10"
## [11] "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18" "chr19" "chr20"
## [21] "chr21" "chr22"

seqlevelsInGroup(newchr, group="circular")

## [1] "chrM"

seqlevelsInGroup(newchr, group="sex","Homo_sapiens","UCSC")

## [1] "chrX" "chrY"
```

if we have a vector containing seqnames and we want to verify the species and style for them , we can use:

```
seqnames <- c("chr1", "chr9", "chr2", "chr3", "chr10")
all(seqnames %in% extractSeqlevels("Homo_sapiens", "UCSC"))

## [1] TRUE
```

2.6 orderSeqlevels

The `orderSeqlevels` can return the order of a given character vector which contains seqnames.In the following example, we show how you can find the order for a given seqnames character vector.

```
seqnames <- c("chr1","chr9", "chr2", "chr3", "chr10")
orderSeqlevels(seqnames)

## [1] 1 3 4 2 5

seqnames[orderSeqlevels(seqnames)]

## [1] "chr1"  "chr2"  "chr3"  "chr9"  "chr10"
```

2.7 rankSeqlevels

The `rankSeqlevels` can return the rank of a given character vector which contains seqnames.In the following example, we show how you can find the rank for a given seqnames character vector.

```
seqnames <- c("chr1","chr9", "chr2", "chr3", "chr10")
rankSeqlevels(seqnames)

## [1] 1 4 2 3 5
```

2.8 mapSeqlevels

Returns a matrix with 1 column per supplied sequence name and 1 row per sequence renaming map compatible with the specified style. If `best.only` is `TRUE` (the default), only the "best" renaming maps (i.e. the rows with less NAs) are returned.

```
mapSeqlevels(c("chrII", "chrIII", "chrM"), "NCBI")

##  chrII chrIII   chrM
##   "II"  "III"   "MT"
```

We also have several seqlevel utility functions.Let us construct a basic GRanges and show how these functions can be used. .

```
gr <- GRanges(paste0("ch",1:35), IRanges(1:35, width=5))
gr

## GRanges object with 35 ranges and 0 metadata columns:
##        seqnames    ranges strand
##           <Rle> <IRanges>  <Rle>
##    [1]      ch1    [1, 5]      *
##    [2]      ch2    [2, 6]      *
##    [3]      ch3    [3, 7]      *
##    [4]      ch4    [4, 8]      *
##    [5]      ch5    [5, 9]      *
##    ...      ...       ...    ...
##   [31]     ch31  [31, 35]      *
##   [32]     ch32  [32, 36]      *
##   [33]     ch33  [33, 37]      *
##   [34]     ch34  [34, 38]      *
##   [35]     ch35  [35, 39]      *
##   -------
##   seqinfo: 35 sequences from an unspecified genome; no seqlengths
```

As you can see , we have "ch" instead of "chr" for chromosome names. We can use `renameSeqlevels` to change the "ch" to "chr"

2.9 renameSeqlevels

As the first argument - it takes the object whose seqlevels we need to change, and as the second argument it takes a named vector which has the changes.

```
newnames <- paste0("chr",1:35)
names(newnames) <- paste0("ch",1:35)
head(newnames)

##    ch1    ch2    ch3    ch4    ch5    ch6
## "chr1" "chr2" "chr3" "chr4" "chr5" "chr6"

gr <- renameSeqlevels(gr,newnames)
gr

## GRanges object with 35 ranges and 0 metadata columns:
##        seqnames    ranges strand
##           <Rle> <IRanges>  <Rle>
##    [1]     chr1    [1, 5]      *
##    [2]     chr2    [2, 6]      *
##    [3]     chr3    [3, 7]      *
##    [4]      chr4    [4, 8]      *
##    [5]      chr5    [5, 9]      *
##    ...       ...       ...    ...
##   [31]     chr31  [31, 35]      *
##   [32]     chr32  [32, 36]      *
##   [33]     chr33  [33, 37]      *
##   [34]     chr34  [34, 38]      *
##   [35]     chr35  [35, 39]      *
##   -------
##   seqinfo: 35 sequences from an unspecified genome; no seqlengths
```

Humans have just 22 primary chromosomes - but here we have some extra seqlevels which we want to remove - there are several ways we can achieve this:

2.10 dropSeqlevels

Here the second argument is the seqlevels that you want to drop.

```
dropSeqlevels(gr,paste0("chr",23:35))
```

```
## GRanges object with 22 ranges and 0 metadata columns:
##        seqnames    ranges strand
##           <Rle> <IRanges>  <Rle>
##    [1]      chr1    [1, 5]      *
##    [2]      chr2    [2, 6]      *
##    [3]      chr3    [3, 7]      *
##    [4]      chr4    [4, 8]      *
##    [5]      chr5    [5, 9]      *
##    ...       ...       ...    ...
##   [18]     chr18  [18, 22]      *
##   [19]     chr19  [19, 23]      *
##   [20]     chr20  [20, 24]      *
##   [21]     chr21  [21, 25]      *
##   [22]     chr22  [22, 26]      *
##   -------
##   seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

2.11 keepSeqlevels

Here the second argument is the seqlevels that you want to keep.

```
keepSeqlevels(gr, paste0("chr",1:22))
```

```
## GRanges object with 22 ranges and 0 metadata columns:
##        seqnames    ranges strand
##           <Rle> <IRanges>  <Rle>
##    [1]      chr1    [1, 5]      *
##    [2]      chr2    [2, 6]      *
##    [3]      chr3    [3, 7]      *
##    [4]      chr4    [4, 8]      *
##    [5]      chr5    [5, 9]      *
##    ...       ...       ...    ...
##   [18]     chr18  [18, 22]      *
##   [19]     chr19  [19, 23]      *
##   [20]    chr20  [20, 24]      *
##   [21]    chr21  [21, 25]      *
##   [22]    chr22  [22, 26]      *
##   -------
##   seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

2.12 keepStandardChromosomes

This function internally uses the pre-defined tables inside GenomeInfoDb to find the correct seqlevels according to the style of the object.

```
keepStandardChromosomes(gr)

## GRanges object with 35 ranges and 0 metadata columns:
##        seqnames    ranges strand
##           <Rle> <IRanges>  <Rle>
##    [1]     chr1    [1, 5]      *
##    [2]     chr2    [2, 6]      *
##    [3]     chr3    [3, 7]      *
##    [4]     chr4    [4, 8]      *
##    [5]     chr5    [5, 9]      *
##    ...      ...       ...    ...
##   [31]    chr31  [31, 35]      *
##   [32]    chr32  [32, 36]      *
##   [33]    chr33  [33, 37]      *
##   [34]    chr34  [34, 38]      *
##   [35]    chr35  [35, 39]      *
##   -------
##   seqinfo: 35 sequences from an unspecified genome; no seqlengths
```

One can also specify the optional species argument to bemore precise.

```
plantgr <- GRanges(c(1:5,"MT","Pltd"), IRanges(1:7,width=5))
keepStandardChromosomes(plantgr, species="Arabidopsis thaliana")

## GRanges object with 7 ranges and 0 metadata columns:
##       seqnames    ranges strand
##          <Rle> <IRanges>  <Rle>
##   [1]        1   [1,  5]      *
##   [2]        2   [2,  6]      *
##   [3]        3   [3,  7]      *
##   [4]        4   [4,  8]      *
##   [5]        5   [5,  9]      *
##   [6]       MT   [6, 10]      *
##   [7]     Pltd   [7, 11]      *
##   -------
##   seqinfo: 7 sequences from an unspecified genome; no seqlengths
```

3 Classes inside GenomeInfoDb package

3.1 Genome-Description class

We also provide a Genome Description class which can be used in the following way:

```
library(BSgenome.Celegans.UCSC.ce2)
class(Celegans)

## [1] "BSgenome"
## attr(,"package")
## [1] "BSgenome"

is(Celegans, "GenomeDescription")

## [1] TRUE

provider(Celegans)

## [1] "UCSC"

seqinfo(Celegans)

## Seqinfo object with 7 sequences (1 circular) from ce2 genome:
##   seqnames seqlengths isCircular genome
##   chrI       15080483      FALSE    ce2
##   chrII      15279308      FALSE    ce2
##   chrIII     13783313      FALSE    ce2
##   chrIV      17493791      FALSE    ce2
##   chrV       20922231      FALSE    ce2
##   chrX       17718849      FALSE    ce2
##   chrM          13794       TRUE    ce2

gendesc <- as(Celegans, "GenomeDescription")
class(gendesc)

## [1] "GenomeDescription"
## attr(,"package")
## [1] "GenomeInfoDb"

gendesc

## | organism: Caenorhabditis elegans (Worm)
## | provider: UCSC
## | provider version: ce2
## | release date: Mar. 2004
## | release name: WormBase v. WS120
## | ---
## | seqlengths:
## |      chrI     chrII    chrIII     chrIV      chrV      chrX      chrM
## |  15080483 15279308 13783313 17493791 20922231 17718849     13794

provider(gendesc)

## [1] "UCSC"

seqinfo(gendesc)

## Seqinfo object with 7 sequences (1 circular) from ce2 genome:
##   seqnames seqlengths isCircular genome
##   chrI       15080483      FALSE    ce2
##   chrII      15279308      FALSE    ce2
##   chrIII     13783313      FALSE    ce2
##   chrIV      17493791      FALSE    ce2
##   chrV       20922231      FALSE    ce2
##   chrX       17718849      FALSE    ce2
##   chrM          13794       TRUE    ce2
bsgenomeName(gendesc)
## [1] "BSgenome.Celegans.UCSC.ce2"
```

3.2 SeqInfo class

```
## Note that all the arguments (except 'genome') must have the
## same length. 'genome' can be of length 1, whatever the lengths
## of the other arguments are.
x <- Seqinfo(seqnames=c("chr1", "chr2", "chr3", "chrM"),
             seqlengths=c(100, 200, NA, 15),
             isCircular=c(NA, FALSE, FALSE, TRUE),
             genome="toy")
length(x)
## [1] 4
seqnames(x)
## [1] "chr1" "chr2" "chr3" "chrM"
names(x)
## [1] "chr1" "chr2" "chr3" "chrM"
seqlevels(x)
## [1] "chr1" "chr2" "chr3" "chrM"
seqlengths(x)
## chr1 chr2 chr3 chrM
##  100  200   NA   15
isCircular(x)
##  chr1  chr2  chr3  chrM
##    NA FALSE FALSE  TRUE
genome(x)
##  chr1  chr2  chr3  chrM
## "toy" "toy" "toy" "toy"
x[c("chrY", "chr3", "chr1")]  # subset by names
## Seqinfo object with 3 sequences from 2 genomes (NA, toy):
##   seqnames seqlengths isCircular genome
##   chrY             NA         NA   <NA>
##   chr3             NA      FALSE    toy
##   chr1            100         NA    toy
## Rename, drop, add and/or reorder the sequence levels:
xx <- x
seqlevels(xx) <- sub("chr", "ch", seqlevels(xx))  # rename
xx
```

```
## Seqinfo object with 4 sequences (1 circular) from toy genome:
##   seqnames seqlengths isCircular genome
##   ch1             100         NA    toy
##   ch2             200      FALSE    toy
##   ch3              NA      FALSE    toy
##   chM              15       TRUE    toy

seqlevels(xx) <- rev(seqlevels(xx))  # reorder
xx

## Seqinfo object with 4 sequences (1 circular) from toy genome:
##   seqnames seqlengths isCircular genome
##   chM              15       TRUE    toy
##   ch3              NA      FALSE    toy
##   ch2             200      FALSE    toy
##   ch1             100         NA    toy

seqlevels(xx) <- c("ch1", "ch2", "chY")  # drop/add/reorder
xx

## Seqinfo object with 3 sequences from 2 genomes (toy, NA):
##   seqnames seqlengths isCircular genome
##   ch1             100         NA    toy
##   ch2             200      FALSE    toy
##   chY              NA         NA   <NA>

seqlevels(xx) <- c(chY="Y", ch1="1", "22")  # rename/reorder/drop/add
xx

## Seqinfo object with 3 sequences from 2 genomes (NA, toy):
##   seqnames seqlengths isCircular genome
##   Y                NA         NA   <NA>
##   1               100         NA    toy
##   22               NA         NA   <NA>

y <- Seqinfo(seqnames=c("chr3", "chr4", "chrM"),
             seqlengths=c(300, NA, 15))
y

## Seqinfo object with 3 sequences from an unspecified genome:
##   seqnames seqlengths isCircular genome
##   chr3            300         NA   <NA>
##   chr4             NA         NA   <NA>
##   chrM             15         NA   <NA>

merge(x, y)  # rows for chr3 and chrM are merged

## Warning in .Seqinfo.mergexy(x, y):  Each of the 2 combined objects has sequence levels not in
the other:
##  - in 'x':  chr1, chr2
##  - in 'y':  chr4
##  Make sure to always combine/compare objects based on the same reference
##  genome (use suppressWarnings() to suppress this warning).

## Seqinfo object with 5 sequences (1 circular) from 2 genomes (toy, NA):
##   seqnames seqlengths isCircular genome
##   chr1            100         NA    toy
##   chr2            200      FALSE    toy
##   chr3            300      FALSE    toy
##   chrM             15       TRUE    toy
##   chr4             NA         NA   <NA>
suppressWarnings(merge(x, y))
## Seqinfo object with 5 sequences (1 circular) from 2 genomes (toy, NA):
##   seqnames seqlengths isCircular genome
##   chr1            100         NA    toy
##   chr2            200      FALSE    toy
##   chr3            300      FALSE    toy
##   chrM             15       TRUE    toy
##   chr4             NA         NA   <NA>
## Note that, strictly speaking, merging 2 Seqinfo objects is not
## a commutative operation, i.e., in general 'z1 <- merge(x, y)'
## is not identical to 'z2 <- merge(y, x)'. However 'z1' and 'z2'
## are guaranteed to contain the same information (i.e. the same
## rows, but typically not in the same order):
suppressWarnings(merge(y, x))
## Seqinfo object with 5 sequences (1 circular) from 2 genomes (toy, NA):
##   seqnames seqlengths isCircular genome
##   chr3            300      FALSE    toy
##   chr4             NA         NA   <NA>
##   chrM             15       TRUE    toy
##   chr1            100         NA    toy
##   chr2            200      FALSE    toy
## This contradicts what 'x' says about circularity of chr3 and chrM:
isCircular(y)[c("chr3", "chrM")] <- c(TRUE, FALSE)
y
## Seqinfo object with 3 sequences (1 circular) from an unspecified genome:
##   seqnames seqlengths isCircular genome
##   chr3            300       TRUE   <NA>
##   chr4             NA         NA   <NA>
##   chrM             15      FALSE   <NA>
if (interactive()) {
  merge(x, y)  # raises an error
}
```

4 Examples

4.1 converting seqlevel styles (eg:UCSC to NCBI)

A quick example using Drosophila Melanogaster. The txdb object contains seqlevels in UCSC style, we want to convert them to NCBI

```
txdb <- TxDb.Dmelanogaster.UCSC.dm3.ensGene
seqlevels(txdb)
##  [1] "chr2L"     "chr2R"     "chr3L"     "chr3R"     "chr4"      "chrX"      "chrU"
##  [8] "chrM"      "chr2LHet"  "chr2RHet"  "chr3LHet"  "chr3RHet"  "chrXHet"   "chrYHet"
## [15] "chrUextra"
```

```
genomeStyles("Drosophila melanogaster")

##    circular   sex  auto  NCBI      UCSC                  Ensembl
## 1     FALSE FALSE  TRUE    2L     chr2L                       2L
## 2     FALSE FALSE  TRUE    2R     chr2R                       2R
## 3     FALSE FALSE  TRUE    3L     chr3L                       3L
## 4     FALSE FALSE  TRUE    3R     chr3R                       3R
## 5     FALSE FALSE  TRUE     4      chr4                        4
## 6     FALSE  TRUE FALSE     X      chrX                        X
## 7      TRUE FALSE FALSE    MT      chrM dmel_mitochondrion_genome
## 8     FALSE FALSE FALSE 2LHet   chr2LHet                   2LHet
## 9     FALSE FALSE FALSE 2Rhet   chr2RHet                   2RHet
## 10    FALSE FALSE FALSE 3LHet   chr3LHet                   3LHet
## 11    FALSE FALSE FALSE 3RHet   chr3RHet                   3RHet
## 12    FALSE FALSE FALSE  Xhet    chrXHet                    XHet
## 13    FALSE FALSE FALSE  Yhet    chrYHet                    YHet
## 14    FALSE FALSE FALSE    Un      chrU                        U
## 15    FALSE FALSE FALSE  <NA> chrUextra                   Uextra

mapSeqlevels(seqlevels(txdb), "NCBI")

##     chr2L     chr2R     chr3L     chr3R      chr4      chrX      chrU      chrM  chr2LHet
##      "2L"      "2R"      "3L"      "3R"       "4"       "X"      "Un"      "MT"   "2LHet"
##  chr2RHet  chr3LHet  chr3RHet   chrXHet   chrYHet chrUextra
##   "2Rhet"   "3LHet"   "3RHet"    "Xhet"    "Yhet"        NA
```

4.2 converting styles and removing unwanted seqlevels

Suppose we read in a Bam file or a BED file and the resulting GRanges have a lot of seqlevels which are not required by your analysis or you want to rename the seqlevels from the current style to your own style (eg:USCS to NCBI), we can use the functionality provided by GenomeInfoDb to do that.

Let us say that we have extracted the seqlevels of the Seqinfo object(say GRanges from a BED file) in a variable called "sequence".

```
sequence <- seqlevels(x)

## sequence is in UCSC format and we want NCBI style
newStyle <- mapSeqlevels(sequence,"NCBI")
newStyle <- newStyle[complete.cases(newStyle)] # removing NA cases.

## rename the seqlevels
x <- renameSeqlevels(x,newStyle)

## keep only the seqlevels you want (say autosomes)
auto <- extractSeqlevelsByGroup(species="Homo sapiens", style="NCBI",
                                group="auto")
x <- keepSeqlevels(x,auto)
```

5 Session Information

Here is the output of `sessionInfo` on the system on which this document was compiled:

```
toLatex(sessionInfo())
```

- R version 3.2.3 (2015-12-10), `x86_64-apple-darwin13.4.0`
- Locale: `C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.32.3, BSgenome 1.38.0, BSgenome.Celegans.UCSC.ce2 1.4.0, Biobase 2.30.0, BiocGenerics 0.16.1, Biostrings 2.38.3, GenomeInfoDb 1.6.3, GenomicFeatures 1.22.9, GenomicRanges 1.22.3, IRanges 2.4.6, S4Vectors 0.8.9, TxDb.Dmelanogaster.UCSC.dm3.ensGene 3.2.2, XVector 0.10.0, rtracklayer 1.30.1
- Loaded via a namespace (and not attached): BiocParallel 1.4.3, BiocStyle 1.8.0, DBI 0.3.1, GenomicAlignments 1.6.3, RCurl 1.95-4.7, RSQLite 1.0.0, Rsamtools 1.22.0, SummarizedExperiment 1.0.2, XML 3.98-1.3, biomaRt 2.26.1, bitops 1.0-6, evaluate 0.8, formatR 1.2.1, futile.logger 1.4.1, futile.options 1.0.0, highr 0.5.1, knitr 1.12.3, lambda.r 1.1.7, magrittr 1.5, stringi 1.0-1, stringr 1.0.0, tools 3.2.3, zlibbioc 1.16.0