Package 'ICAMS'

January 28, 2019

Type Package

Title In-depth Characterization and Analysis of Mutational Signatures

Version 0.0.0.9001

Author Steve Rozen, Nanhai Jiang, Arnoud Boot

Maintainer Steve Rozen <steverozen@gmail.com>

Description This package has functions to read in VCF files from Strelka and GATK, create SNS, DNS, ID catalogs and do different types of plotting.

License GPL-3

Encoding UTF-8

LazyData true

Imports Biostrings (>= 2.50.2),
BSgenome (>= 1.50.0),
BSgenome.Hsapiens.1000genomes.hs37d5 (>= 0.99.1),
data.table,
dplyr,
GenomicRanges (>= 1.34.0),
graphics,
grDevices,
methods,
RColorBrewer,
RCurl,
stringr,
utils

Depends R (>= 3.5),

RoxygenNote 6.1.1

Suggests knitr,
rmarkdown,
testthat

VignetteBuilder knitr

Collate 'ICAMS.R'
'INDELS_related_functions.R'
'utility_functions.R'
'VCF_to_catalog_functions.R'
'plot_SNS_catalog.R'
'plot_DNS_catalog.R'
'plot_INDELS_catalog.R'

'read_write_catalog.R'
'test_functions.R'

R topics documented:

`CatalogToPdf` *Catalog to Pdf Functions*

Description

Plot the mutation catalog of different samples to a PDF file

Usage

```
Cat96ToPdf(catalog, name, id = colnames(catalog), type = "density",
  grid = FALSE, upper = TRUE, xlabels = TRUE, abundance = NULL)

Cat192ToPdf(catalog, name, id = colnames(catalog), type = "counts",
  cex = 0.8, abundance = NULL)

Cat192StrandToPdf(catalog, name, id = colnames(catalog),
  type = "counts", cex = 1, abundance = NULL)

Cat1536ToPdf(catalog, name, id = colnames(catalog), abundance)

CatDNS78ToPdf(catalog, name, id = colnames(catalog), type = "density",
  abundance = NULL)

CatDNS144ToPdf(catalog, name, id = colnames(catalog), type = "counts",
  cex = 1, abundance = NULL)

CatIDToPdf(catalog, name, id = colnames(catalog), type = "counts")
```

Arguments

catalog	A matrix whose rownames indicate the mutation types while its columns contain the counts of each mutation type from different samples.
name	The name of the PDF file to be produced.
id	A vector containing the ID information of different samples.
type	A vector of values indicating the type of plot for each sample. If type = "counts", the graph will plot the occurrences of the mutation types in the sample. If type = "signature", the graph will plot mutation signatures of the sample. If type = "density", the graph will plot the rates of mutations per million nucleotides for each mutation type. (Please take note there is no "density" type for CatIDtoPdf function and the option of type = "density" is not implemented for function Cat192ToPdf, Cat192StrandToPdf and CatDNS144ToPdf at the current stage.)
grid	A logical value indicating whether to draw the grid lines in the graph.
upper	A logical value indicating whether to draw horizontal lines and names of major mutation class on top of graph.
xlabels	A logical value indicating whether to draw x axis labels.
abundance	A matrix containing nucleotide abundance information, to be used only when type = "density".
cex	A numerical value giving the amount by which mutation class labels, y axis labels, sample name and legend(if there exists) should be magnified relative to the default.

Details

`Cat96ToPdf` Plot the SNS 96 mutation catalog of different samples to a PDF file.

`Cat192ToPdf` Plot the SNS 192 mutation catalog of different samples to a PDF file.

`Cat192StrandToPdf` Plot the transcription strand bias graph of 6 SNS mutation types ("C>A", "C>G", "C>T", "T>A", "T>C", "T>G") of different samples to a PDF file.

`Cat1536ToPdf` Plot the 1536 mutation catalog of >= 1 samples to a PDF file. The mutation types are in six-letters like CATTAT, first 2-letters CA refers to (-2, -1) position, third letter T refers to the base which has mutation, next second 2-letters TA refers to (+1, +2) position, last letter T refers to the base after mutation.

`CatDNS78ToPdf` Plot the DNS 78 mutation catalog of different samples to a PDF file.

`CatDNS144ToPdf` Plot the transcription strand bias graph of 10 major DNS mutation types ("AC>NN", "AT>NN", "CC>NN", "CG>NN", "CT>NN", "GC>NN", "TA>NN", "TC>NN", "TG>NN", "TT>NN") of different samples to a PDF file.

`CatIDToPdf` Plot the insertion and deletion catalog of different samples to a PDF file. (Please take note that the deletions Repeat Size ranges from 0 to 5+ in the catalog, but for plotting and end user documentation it ranges from 1 to 6+.)

Value

invisible(TRUE)

`CollapseCatalog` *Collapse Catalog Functions*

Description

Collapse a catalog matrix to a canonical one

Usage

```
Collapse192To96(catalog)

Collapse1536To96(catalog)

Collapse144To78(catalog)
```

Arguments

`catalog`	A catalog matrix to be collapsed whose row names indicate the mutation types while its columns show the occurrences of each mutation type of different samples.

Details

`Collapse192To96` Collapse a SNS 192 catalog matrix to a SNS 96 catalog matrix.

`Collapse1536To96` Collapse a SNS 1536 catalog matrix to a SNS 96 catalog matrix.

`Collapse144To78` Collapse a DNS 144 catalog matrix to a DNS 78 catalog matrix.

Value

A canonical catalog matrix whose row names indicate the mutation types while its columns show the occurrences of each mutation type of different samples.

`CreateDinucAbundance` *Create Dinucleotide abundance file*

Description

Create Dinucleotide abundance file

Usage

```
CreateDinucAbundance(path)
```

Arguments

`path`	Path to the file with the nucleotide abundance information with 4 base pairs.

Value

A matrix whose row names indicate 10 different types of 2 base pairs combinations while its column contains the occurrences of each type.

`CreateOneColIDCatalog` *Create an indel (ID) mutation catalog for *one* sample from a Variant Call Format (VCF) file*

Description

Create an indel (ID) mutation catalog for *one* sample from a Variant Call Format (VCF) file

Usage

```
CreateOneColIDCatalog(ID.vcf, SBS.vcf)
```

Arguments

`ID.vcf`	An in-memory VCF as a data.frame annotated by the AddSequence and AddTranscript functions. It must only contain indels and must *not* contain SBS (single base substituions), DBS, or triplet base substituions etc. * Sequence must already have been added to ID.vcf One design decision for variant callers is the representation of "complex indels", e.g. mutations e.g. CAT > GC. Some callers represent this as C>G, A>C, and T>_. Others might represent it as CAT > CG. Multiple issues can arise. In PCAWG, overlapping indel/SBS calls from different callers were included in the indel VCFs.
`SBS.vcf`	An in-memory VCF as a data frame. Because we have to work with some PCAWG data, we will look for neigboring indels and indels adjoining SBS. That means this functions takes an SBS VCF and an ID VCF from the same sample.

Value

A list with two elemsents: ID.cat: A 1-column matrix containing the mutation catalog information. problems: Locations of neighboring indels or indels neighboring SBS. In the future we might handle these depending on what we find in the indel calls from different variant callers. TODO(steve) Is problems implemented?

`CreatePentanucAbundance` *Create pentanucleotide abundance file*

Description

Create pentanucleotide abundance file

Usage

```
CreatePentanucAbundance(path)
```

Arguments

`path`	Path to the file with the nucleotide abundance information with 5 base pairs.

Value

A matrix whose row names indicate 512 different types of 5 base pairs combinations while its column contains the occurrences of each type.

`CreateTetranucAbundance` *Create Tetranucleotide abundance file*

Description

Create Tetranucleotide abundance file

Usage

```
CreateTetranucAbundance(path)
```

Arguments

`path`	Path to the file with the nucleotide abundance information with 4 base pairs.

Value

A matrix whose row names indicate 136 different types of 4 base pairs combinations while its column contains the occurrences of each type.

CreateTrinucAbundance *Create Trinucleotide abundance file*

Description

Create Trinucleotide abundance file

Usage

```
CreateTrinucAbundance(path)
```

Arguments

path	Path to the file with the nucleotide abundance information with 3 base pairs.

Value

A matrix whose row names indicate 32 different types of 3 base pairs combinations while its column contains the occurrences of each type.

FindDelMH *Return the length of microhomology at a deletion*

Description

Return the length of microhomology at a deletion

Usage

```
FindDelMH(context, deleted.seq, pos, trace = 0)
```

Arguments

context	The deleted sequence plus ample surrounding sequence on each side (at least as long as del.sequence).
deleted.seq	The deleted sequence in context. #'
pos	The position of del.sequence in context.
trace	If > 0, cat various messages.

Details

This function is primarily for internal use, but we export it so that the somewhat complicated logic behind it will be documented for users.

Example:

GGCTAGTT aligned to GGCTAGAACTAGTT with a deletion represented as:

```
GGCTAGAACTAGTT
GG------CTAGTT GGCTAGTT GG[CTAGAA]CTAGTT
                           ----   ----
```

Presumed repair mechanism leading to this:

```
 ....
GGCTAGAACTAGTT
CCGATCTTGATCAA

=>

  ....
GGCTAG      TT
CC      GATCAA
        ....

=>

GGCTAGTT
CCGATCAA
```

The same deletion can be represented in several different ways.

```
GGC------TAGTT GGCTAGTT GGC[TAGAAC]TAGTT
                         * ---  * ---

GGCT------AGTT GGCTAGTT GGCT[AGAACT]AGTT
                         ** --  ** --

GGCTA------GTT GGCTAGTT GGCTA[GAACTA]GTT
                         *** -  *** -

GGCTAG------TT GGCTAGTT GGCTAG[AACTAG]TT
                         ****   ****
```

A deletion in a *repeat* can also be represented in several different ways. A deletion in a repeat is abstractly equivalent to microhomology that spans the entire deleted sequence. For example;

```
GACTAGCTAGTT
GACTA----GTT GACTAGTT GACTA[GCTA]GTT
                      *** -*** -
```

is really a repeat

```
TODO(steve): add check in code
GACTAG----TT GACTAGTT GACTAG[CTAG]TT
                      **** ----
```

```
GACT----AGTT GACTAGTT GACT[AGCT]AGTT
                       ** --** --
```

But the function only flags this with a -1 return; it does not figure out the repeat extent.

In the implementation, the function finds:

1. The maxium match of undeleted sequence on left that is identical to the right end of the deleted sequence, and
2. The maxium match of undeleted sequence on the right this is identical to the left end of the deleted sequence.

The microhomology sequence is the concatenation of items (1) and (2).

Value

The length of the maxium microhomology of `del.sequence` in `context`.

`GetMutectVAF` *Extract the VAFs (variant allele frequencies) from a VCF created by MuTect*

Description

Extract the VAFs (variant allele frequencies) from a VCF created by MuTect

Usage

```
GetMutectVAF(mutect.vcf)
```

Arguments

`mutect.vcf`	said VCF as a data.frame

Value

A vector of VAFs, one for each row of mutect.vcf

`GetStrelkaVAF` *Extract the VAFs (variant allele frequencies) from a VCF created by Strelka version 1*

Description

Extract the VAFs (variant allele frequencies) from a VCF created by Strelka version 1

Usage

```
GetStrelkaVAF(strelka.vcf)
```

Arguments

`strelka.vcf`	said VCF as a data.frame

Value

A vector of VAFs, one for each row of strelka.vcf

`ICAMS` *ICAMS: In-depth Characterization and Analysis of Mutational Signatures*

Description

This package has functions to read in VCF files from Strelka and GATK, create SNS, DNS, ID catalogs and do different types of plotting.

Reading catalogs

Functions for reading a catalog in PCAWG7 format from path: `ReadCatalog`

Writing catalogs

Functions for writting a mutation catalog to a file on disk: `WriteCatalog`

Collapsing catalogs

Functions for collapsing a mutation catalog to a canonical one: `CollapseCatalog`

Plotting catalogs

Functions for plotting the mutation catalog of one sample: `PlotCatalog`

Functions for plotting mutation catalog of different samples to a PDF file: `CatalogToPdf`

MakeVCFDNSdf *MakeVCFDNSdf TODO(steve) add average VAF*

Description

Take DNS ranges and the original VCF and generate a VCF with dinucleotide REF and ALT alleles. The output VCF has minimal columns: just CHROM, POS, ID, REF, ALT.

Usage

```
MakeVCFDNSdf(DNS.range.df, SNS.vcf.dt)
```

Arguments

DNS.range.df	Data frame with columns CHROM, LOW, HIGH
SNS.vcf.dt	TODO

Value

TODO

NewTestMakeCatalogFromStrelkaSNSVCFs *This function is to make catalogs from the sample VCF files to compare with the expected catalog information.*

Description

This function is to make catalogs from the sample VCF files to compare with the expected catalog information.

Usage

```
NewTestMakeCatalogFromStrelkaSNSVCFs()
```

NewTestStrelkaDNSCatalog *This function is to test whether the predefined functions are working correctly to produce the desired DNS catalogs from Strelka VCF.*

Description

This function is to test whether the predefined functions are working correctly to produce the desired DNS catalogs from Strelka VCF.

Usage

```
NewTestStrelkaDNSCatalog()
```

NewTestStrelkaSNSCatalog *This function is to test whether the predefined functions are working correctly to produce the desired SNS catalogs from Strelka VCF.*

Description

This function is to test whether the predefined functions are working correctly to produce the desired SNS catalogs from Strelka VCF.

Usage

```
NewTestStrelkaSNSCatalog()
```

NewVCFsToDNSCatalogs *Create DNS catalogs from VCFs*

Description

Create a list of 3 catalogs (one each for DNS78, DNS144 and QUAD136) out of the contents of the VCFs in list.of.vcfs

Usage

```
NewVCFsToDNSCatalogs(list.of.vcfs, genome, trans.ranges)
```

Arguments

list.of.vcfs	List of in-memory VCFs. The list names will be the sample ids in the output catalog.
genome	Name of a particular reference genome (without quotations marks).
trans.ranges	A data frame containing transcript ranges.

Value

A list of 3 catalogs, one each for DNS78, DNS144, QUAD136: catDNS78 catDNS144 catQUAD136

PlotCatalog *Plot Catalog Functions*

Description

Plot the mutation catalog of one sample

Usage

```
PlotCat96(catalog, id, type = "density", cex = 0.8, grid = TRUE,
  upper = TRUE, xlabels = TRUE, abundance = NULL)

PlotCat192(catalog, id, type = "counts", cex = 0.8, abundance = NULL)

PlotCat192Strand(catalog, id, type = "counts", cex = 1,
  abundance = NULL)

PlotCat1536(catalog, id, abundance)

PlotCatDNS78(catalog, id, type = "density", abundance = NULL)

PlotCatDNS144(catalog, id, type = "counts", cex = 1,
  abundance = NULL)

PlotQUAD136(catalog, id = colnames(catalog), type = "density",
  abundance = .abundance.4bp)

PlotCatID(catalog, id, type = "counts")
```

Arguments

catalog	A matrix whose rownames indicate the mutation types while its columns contain the counts of each mutation type.
id	The ID information of the sample which has mutations.
type	A value indicating the type of graph. If type = "counts", the graph will plot the occurrences of the mutation types in the sample. If type = "signature", the graph will plot mutation signatures of the sample. If type = "density", the graph will plot the rates of mutations per million nucleotides for each mutation type. (Please take note there is no "density" type for PlotCatID function and the option of type = "density" is not implemented for function PlotCat192, PlotCat192Strand and PlotCatDNS144 at the current stage.)
cex	A numerical value giving the amount by which mutation class labels, mutation counts(if there exists), y axis and its labels, x axis labels and its annotations(if there exists) sample name and legend(if there exists) should be magnified relative to the default.
grid	A logical value indicating whether to draw the grid lines in the graph.
upper	A logical value indicating whether to draw horizontal lines and names of major mutation class on top of graph.
xlabels	A logical value indicating whether to draw x axis labels.

abundance A matrix containing nucleotide abundance information and strand information(if there exists), to be used only when type = "density".

Details

`PlotCat96` Plot the SNS 96 mutation catalog of one sample.

`PlotCat192` Plot the SNS 192 mutation catalog of one sample.

`PlotCat192Strand` Plot the transcription strand bias graph of 6 SNS mutation types ("C>A", "C>G", "C>T", "T>A", "T>C", "T>G") in one sample.

`PlotCat1536` Plot the pentanucleotide sequence contexts for one sample, normalized by pentanucleotide occurrence in the genome. The mutation types are in six-letters like CATTAT, first 2-letters CA refers to (-2, -1) position, third letter T refers to the base which has mutation, next second 2-letters TA refers to (+1, +2) position, last letter T refers to the base after mutation.

`PlotCatDNS78` Plot the DNS 78 mutation catalog of one sample.

`PlotCatDNS144` Plot the transcription strand bias graph of 10 major DNS mutation types ("AC>NN", "AT>NN", "CC>NN", "CG>NN", "CT>NN", "GC>NN", "TA>NN", "TC>NN", "TG>NN", "TT>NN") in one sample.

`PlotQUAD136` Plot the tetranucleotide sequence contexts of 10 major DNS mutation types ("AC>NN", "AT>NN", "CC>NN", "CG>NN", "CT>NN", "GC>NN", "TA>NN", "TC>NN", "TG>NN", "TT>NN") for one sample, normalized by tetranucleotide occurrence in the genome.

`PlotCatID` Plot the insertion and deletion catalog of one sample. (Please take note that the deletions Repeat Size ranges from 0 to 5+ in the catalog, but for plotting and end user documentation it ranges from 1 to 6+.)

Value

invisible(TRUE)

`ReadCatalog` *Read Catalog Functions*

Description

Read a catalog in PCAWG7 format from path

Usage

```
ReadCat96(path, strict = TRUE)

ReadCat192(path, strict = TRUE)

ReadCat1536(path, strict = TRUE)

ReadCatDNS78(path, strict = TRUE)

ReadCatDNS144(path, strict = TRUE)

ReadCatQUAD136(path, strict = TRUE)

ReadCatID(path, strict = TRUE)
```

Arguments

path	Path to a catalog on disk in the "PCAWG7" format.
strict	If TRUE, do additional checks on the input, and stop if the checks fail.

Details

ReadCat96 Read a 96 SNS catalog from path

ReadCat192 Read a 192 SNS catalog from path

ReadCat1536 Read a 1536 SNS catalog from path

ReadCatDNS78 Read a 78 DNS catalog from path

ReadCatDNS144 Read a 144 DNS catalog from path

ReadCatQUAD136 Read a 136 QUAD catalog from path

ReadCatID Read a ID (insertion/deletion) catalog from path Please take note that the deletions Repeat Size ranges from 0 to 5+ in the catalog, but for plotting and end user documentation it ranges from 1 to 6+.

See also WriteCatalog

Value

A catalog in canonical in-memory format.

ReadListOfMutectVCFs *Read a list of Mutect VCF files from path*

Description

Read a list of Mutect VCF files from path

Usage

```
ReadListOfMutectVCFs(vector.of.file.paths)
```

Arguments

vector.of.file.paths	A vector containing the paths of the VCF files.

Value

A list of vcfs from vector.of.file.paths.

`ReadListOfStrelkaVCFs` *Read a list of Strelka VCF files from path*

Description

Read a list of Strelka VCF files from path

Usage

```
ReadListOfStrelkaVCFs(vector.of.file.paths)
```

Arguments

`vector.of.file.paths`	A vector containing the paths of the VCF files.

Value

A list of vcfs from vector.of.file.paths.

`ReadTranscriptRanges` *Read transcript ranges and strands from a gff3 format file. Use this one for the new, cut down gff3 file (2018 11 24)*

Description

Read transcript ranges and strands from a gff3 format file. Use this one for the new, cut down gff3 file (2018 11 24)

Usage

```
ReadTranscriptRanges(path)
```

Arguments

`path`	Path to the file with the transcript information with 1-based start end positions of genomic ranges.

Value

A data.table keyed by chrom, chromStart, and chromEnd.

revc *Reverse complement every string in string.vec*

Description

Reverse complement every string in string.vec

Usage

```
revc(string.vec)
```

Arguments

string.vec	a vector of type character.

Value

A vector of type characters with the reverse complement of every string in string.vec.

SplitListOfStrelkaVCFs *Split a list of in-memory Strelka VCF into SNS, DNS, and variants involving > 2 consecutive bases*

Description

SNSs are single nucleotide substitutions, eg C>T, A<G,.... DNSs are double nucleotide substitutions, eg CC>TT, AT>GG, ... Variants involving > 2 consecutive bases are rare, so this function just records them. These would be variants such ATG>CCT, AGAT > TCTA, ...

Usage

```
SplitListOfStrelkaVCFs(list.of.vcfs)
```

Arguments

list.of.vcfs	A list of in-memory data frame containing Strelka VCF file contents.

Value

A list of 3 in-memory objects with the elements:

`SplitMutectVCFs` *Split each Mutect VCF into SBS, DBS, and ID VCFs (plus two left-over data.frames)*

Description

Split each Mutect VCF into SBS, DBS, and ID VCFs (plus two left-over data.frames)

Usage

```
SplitMutectVCFs(list.of.vcfs)
```

Arguments

`list.of.vcfs`	List of VCFs as in-memory data.frames

Value

A list with 5 elements, as folows:

`StrelkaVCFFilesToCatalog` *Create SNS and DNS catalogs from Strelka VCF files*

Description

Create 3 SNS catalogs (96, 192, 1536) and 3 DNS catalogs (78, 136, 144) from the Strelka VCFs specified by vector.of.file.paths

Usage

```
StrelkaVCFFilesToCatalog(vector.of.file.paths, genome, trans.ranges)
```

Arguments

`vector.of.file.paths`	A vector containing the paths of the Strelka VCF files.
`genome`	Name of a particular reference genome (without quotations marks).
`trans.ranges`	A data.table which contains transcript range and strand information.

Details

This function calls `StrelkaVCFsToSNSCatalogs` and `StrelkaVCFsToDNSCatalogs`

Value

A list of 3 SNS catalogs (one each for 96, 192, and 1536) and 3 DNS catalogs (one each for 78, 136, and 144)

`TestMakeCatalogFromStrelkaSNSVCFs` *This function is to make catalogs from the sample VCF files to compare with the expected catalog information.*

Description

This function is to make catalogs from the sample VCF files to compare with the expected catalog information.

Usage

```
TestMakeCatalogFromStrelkaSNSVCFs()
```

`TestMutectVCFToCatalog` *test* `SplitMutectVCFs`.

Description

test `SplitMutectVCFs`.

Usage

```
TestMutectVCFToCatalog()
```

Details

Stop if the catalogs created do not match the expected values.

`TestStrelkaDNSCatalog` *This function is to test whether the predefined functions are working correctly to produce the desired DNS catalogs from Strelka VCF.*

Description

This function is to test whether the predefined functions are working correctly to produce the desired DNS catalogs from Strelka VCF.

Usage

```
TestStrelkaDNSCatalog()
```

`TestStrelkaSNSCatalog` *This function is to test whether the predefined functions are working correctly to produce the desired SNS catalogs from Strelka VCF.*

Description

This function is to test whether the predefined functions are working correctly to produce the desired SNS catalogs from Strelka VCF.

Usage

```
TestStrelkaSNSCatalog()
```

`VCFsToIDCatalogs` *Create ID (indel) catalog from VCFs*

Description

Create ID (indel) catalog from VCFs

Usage

```
VCFsToIDCatalogs(list.of.vcfs, genome)
```

Arguments

`list.of.vcfs`	List of in-memory VCFs. The list names will be the sample ids in the output catalog.
`genome`	Name of a particular reference genome (without quotations marks).

Value

An ID (indel) catalog

`WriteCatalog` *Write Catalog Functions*

Description

Write a mutation catalog to a file on disk

Usage

```
WriteCat96(ct, path, strict = TRUE)

WriteCat192(ct, path, strict = TRUE)

WriteCat1536(ct, path, strict = TRUE)

WriteCatDNS78(ct, path, strict = TRUE)

WriteCatDNS144(ct, path, strict = TRUE)

WriteCatQUAD136(ct, path, strict = TRUE)

WriteCatID(ct, path, strict = TRUE)
```

Arguments

ct	A matrix of mutation catalog.
path	The path of the file to be written on disk.
strict	If TRUE, do additional checks on the input, and stop if the checks fail.

Details

WriteCat96 Write a SNS 96 mutation catalog to a file on disk

WriteCat192 Write a SNS 192 mutation catalog to a file on disk

WriteCat1536 Write a SNS 1536 mutation catalog to a file on disk

WriteCatDNS78 Write a DNS 78 mutation catalog to a file on disk

WriteCatDNS144 Write a DNS 144 mutation catalog to a file on disk

WriteCatQUAD136 Write a 136 QUAD catalog from path

WriteCatID Write a ID (insertion/deletion) catalog to a file on disk Please take note that the deletions Repeat Size ranges from 0 to 5+ in the catalog, but for plotting and end user documentation it ranges from 1 to 6+.

See also ReadCatalog

Index