Building efficient NGS analysis pipelines with ViennaNGS

Michael T. Wolfinger
Jörg Fallmann
Florian Eggenhofer
Fabian Amman

30th TBI Winter Seminar
Bled, Slovenia

19 Febuary 2015

universität wien

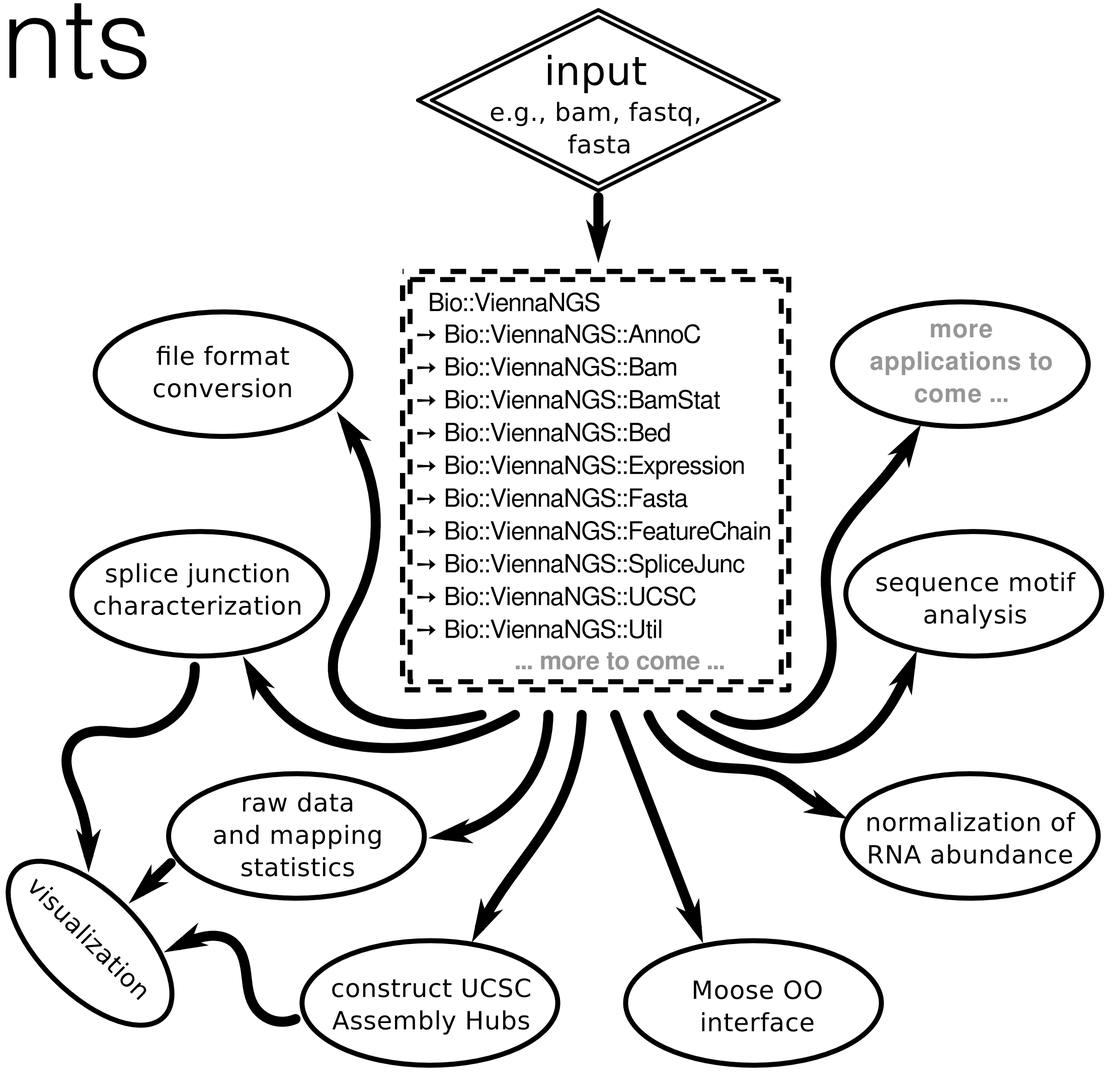
ViennaNGS Executive Summary

`ViennaNGS` is a Perl distribution for building efficient next-generation sequencing (NGS) data analysis pipelines, integrating high-level routines and wrapper functions for common NGS processing tasks.

- Project started in Summer 2014
- Not an established pipeline per se, it provides tools and functionality for the development of custom NGS pipelines in Perl
- Provides modular and reusable code for NGS processing

ViennaNGS Components

`ViennaNGS` implements thematically related functionality in different Perl modules and classes under the **`Bio`** namespace, partly building on *BioPerl* and the *Moose* object framework.



ViennaNGS Module Overview 1/3

`Bio::ViennaNGS::AnnoC`
Lightweight interface for conversion of sequence annotation data

`Bio::ViennaNGS::Bam`
High-level manipulation of BAM files

`Bio::ViennaNGS::BamStat`
Moose based class for collecting mapping statistics

`Bio::ViennaNGS::BamStatSummary`
Interface for processing `BamStatSummary` objects on multiple BAM files

`Bio::ViennaNGS::Util`
Wrapper routines for common third party NGS utils and auxiliary functions

ViennaNGS Module Overview 2/3

`Bio::ViennaNGS::Expression`
Compute normalized expression based on read counts

`Bio::ViennaNGS::Fasta`
Moose wrapper for **`Bio::DB::Fasta`**

`Bio::ViennaNGS::Bed`
Convenience class for handling genomic interval data in BED format

`Bio::ViennaNGS::SpliceJunc`
Identification and characterization of splice junctions

`Bio::ViennaNGS::UCSC`
Automatic generation of UCSC Assembly and Track Hubs

ViennaNGS Module Overview 3/3

`Bio::ViennaNGS::MinimalFeature`
Base class for handling genomic interval data

`Bio::ViennaNGS::Feature`
Interface for simple genomic intervals representing BED6 entries

`Bio::ViennaNGS::ExtFeature`
Extends BED6 elements

`Bio::ViennaNGS::FeatureChain`
Bundles individual **`Feature`** objects

`Bio::ViennaNGS::FeatureLine`
Abstract representation of transcripts, pools **`FeatureChain`** objects

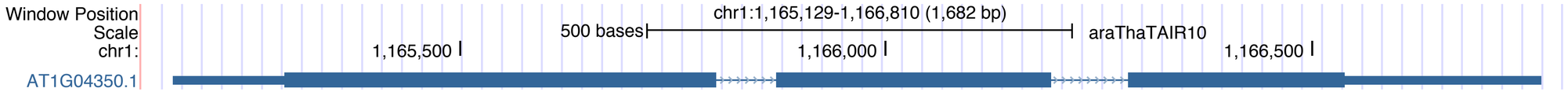
Moose In 30 Seconds

A postmodern object system for Perl 5 that makes Object Oriented programming easier, more consistent, and less tedious

```
package Point;
use Moose;
has 'x' => (is => 'rw', isa => 'Int');
has 'y' => (is => 'rw', isa => 'Int');

sub clear {
  my $self = shift;
  $self->x(0);
  $self->y(0);
}

package Point3D;
use Moose;
extends 'Point';

has 'z' => (is => 'rw', isa => 'Int');

after 'clear' => sub {
 my $self = shift;
 $self->z(0);
};
```

```
use Point;
use Point3D;

my $pt2D = Point->new(x => 2, # x:2
                      y => 4, # y:4
                     );
$pt2D->clear(); # x:0 y:0

my $pt3D = Point3D->new(x => 10, # x:10
                        y => 20, # y:20
                        z => 30  # z:30
                        );
$pt3D->clear; # x:0 y:0 z:0
```

The BED Annotation Format



chr1 1165164 1166768 AT1G04350.1 0 + 1165295 1166538 0 3 637,322,485, 0,708,1119,

1. **chrom** - The name of the chromosome (e.g. chr3, chrY, chr2_random) or scaffold (e.g. scaffold10671).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart=0, chromEnd=100*, and span the bases numbered 0-99.

The 9 additional optional BED fields are:

4. **name** - Defines the name of the BED line. This label is displayed to the left of the BED line in the Genome Browser window when the track is open to full display mode or directly to the left of the item in pack mode.
5. **score** - A score between 0 and 1000. If the track line *useScore* attribute is set to 1 for this annotation data set, the *score* value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). This table shows the Genome Browser's translation of BED score values into shades of gray:

shade									
score in range	≤ 166	167-277	278-388	389-499	500-611	612-722	723-833	834-944	≥ 945

6. **strand** - Defines the strand - either '+' or '-'.
7. **thickStart** - The starting position at which the feature is drawn thickly (for example, the start codon in gene displays). When there is no thick part, thickStart and thickEnd are usually set to the chromStart position.
8. **thickEnd** - The ending position at which the feature is drawn thickly (for example, the stop codon in gene displays).
9. **itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0). If the track line *itemRgb* attribute is set to "On", this RBG value will determine the display color of the data contained in this BED line. NOTE: It is recommended that a simple color scheme (eight colors or less) be used with this attribute to avoid overwhelming the color resources of the Genome Browser and your Internet browser.
10. **blockCount** - The number of blocks (exons) in the BED line.
11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to *blockCount*.
12. **blockStarts** - A comma-separated list of block starts. All of the *blockStart* positions should be calculated relative to *chromStart*. The number of items in this list should correspond to *blockCount*.

Generic Feature Annoation 1/2

Bio::ViennaNGS::MinimalFeature

```
has ‘chromosome’ => (isa => ‘Str’)
has ‘start’      => (isa => ‘Int’)
has ‘end’        => (isa => ‘Int’)
has ‘strand’     => (isa => ‘PlusOrMinus’) # +/-/.
```

Bio::ViennaNGS::Feature

```
extends Bio::ViennaNGS::MinimalFeature
has ‘name’  => (isa => ‘Str’)
has ‘score’ => (isa => ‘Value’)
```

Bio::ViennaNGS::ExtFeature

```
extends Bio::ViennaNGS::Feature
has ‘extension’  => (isa => ‘Str’)
```

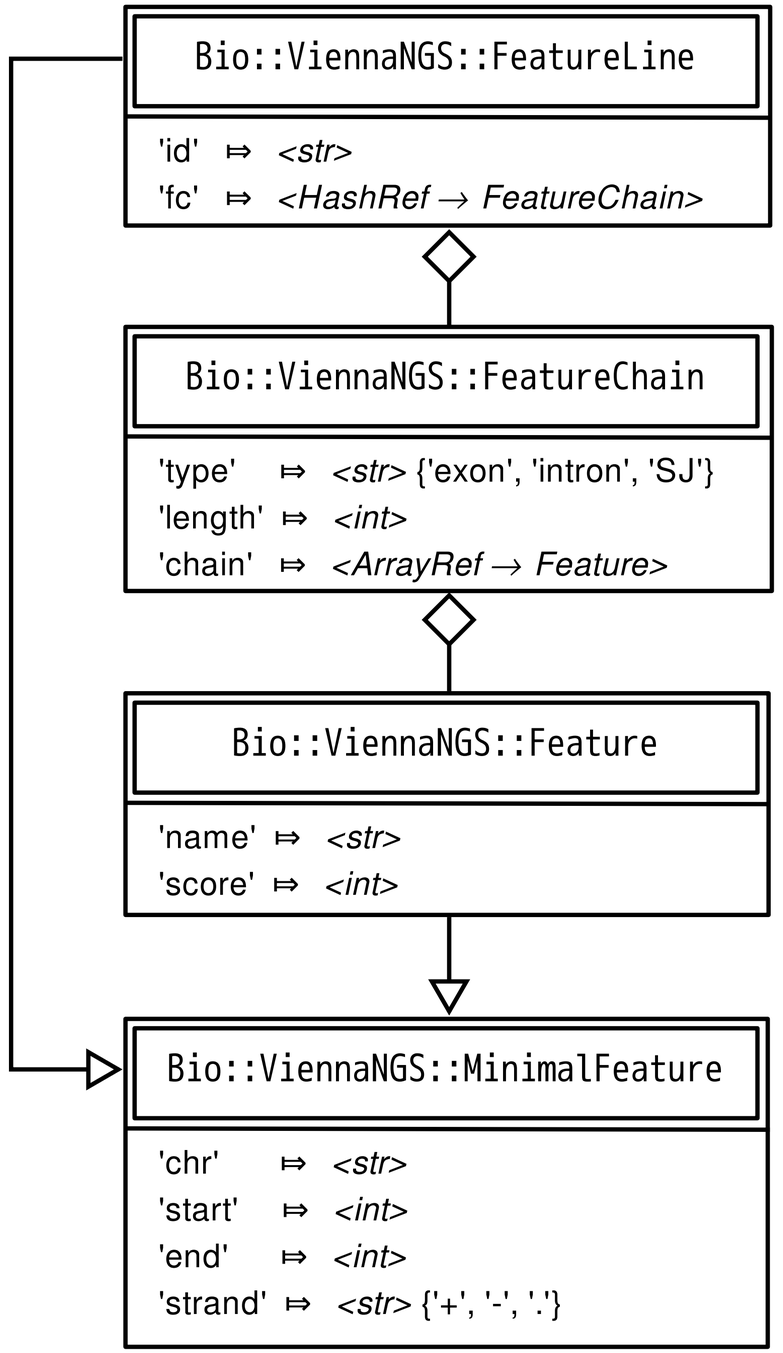
Generic Feature Annoation 2/2

Bio::ViennaNGS::FeatureChain

```
has ‘type’  => (isa => ‘Str’)
has ‘chain’ => (isa => ‘ArrayRef’)
```

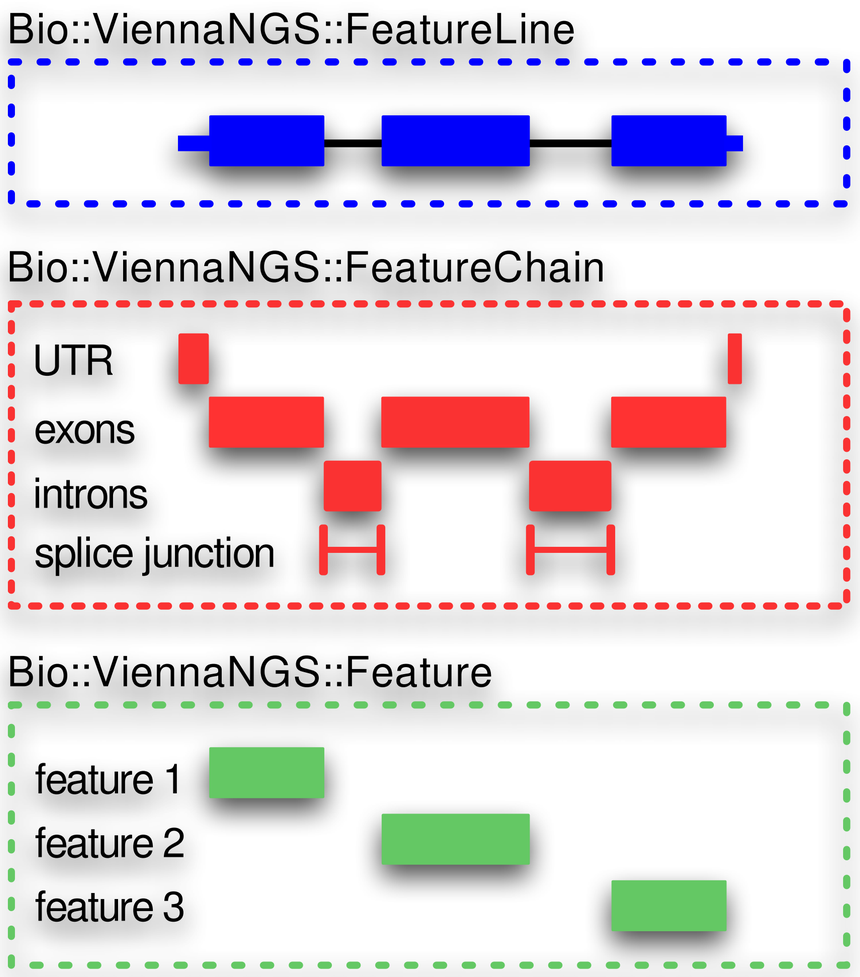
Bio::ViennaNGS::FeatureLine

```
extends Bio::ViennaNGS::MinimalFeature
has ‘id’ => (isa => ‘Str’)
has ‘fc’ => (isa => ‘HashRef’)
```



ViennaNGS Interval Classes

- **`Feature`** extends **`MinimalFeature`** by two attributes, thereby representing a BED6 entry
- **`FeatureChain`** bundles **`Feature`** elements, creating individual annotation chains for e.g. exons, introns,UTRs etc.
- **`FeatureLine`** combines a set of individual **`FeatureChain`** objects, thereby providing a convenient means of representing transcripts



ViennaNGS Documentation and Tutorials

- **`ViennaNGS`** comes with extensive documentation based on Perl's POD system, thereby providing a single documentation base
- **`ViennaNGS::Tutorial`** guides prospective users through the development of basic NGS analysis pipelines
- The tutorial is split into different chapters, each covering a common use case in NGS analysis and describing a possible solution step by step

ViennaNGS Utilities

`ViennaNGS` comes with a collection of complementary executable Perl scripts for accomplishing routine tasks often required in NGS data processing.

Utility	Description
`assembly_hub_constructor.pl`	Construct Assembly Hubs for UCSC genome browser visualization
`bam_quality_stat.pl`	Compute mapping/quality statistics along with publication-ready figures
`bam_split.pl`	Split BAM files by strand
`bam_to_bigwig.pl`	Produce BigWig coverage profiles from BAM files for visualization
`bam_uniq.pl`	Filter uniquely and multi mapped reads from BAM files
`bed2bedGraph.pl`	Convert BED to (strand specific) bedGraph format
`extend_bed.pl`	Extend genomic intervals in BED format at the 5', 3', or both ends
`gff2bed.pl`	Convert bacterial RefSeq GFF3 annotation to BED12 format
`kmer_analysis.pl`	Count k-mers of predefined length in FastQ and Fasta files
`MEME_xml_motif_extractor.pl`	Compute basic statistics from MEME XML output
`newUCSCdb.pl`	Create a new genome database in a local UCSC genome browser instance
`normalize_multicov.pl`	Compute normalized expression data in TPM from read counts
`sj_visualizer.pl`	Convert splice junctions in segemehl BED6 splice junction format to BED12
`splice_site_summary.pl`	Identify and characterize splice junctions from RNA-seq data
`track_hub_constructor.pl`	Construct Track Hubs for UCSC genome browser visualization
`trim_fastq.pl`	Trim sequence and quality fields in FastQ format

These CLI utilities serve as reference implementations of the library routines and can readily be used for atomic tasks in NGS data processing.

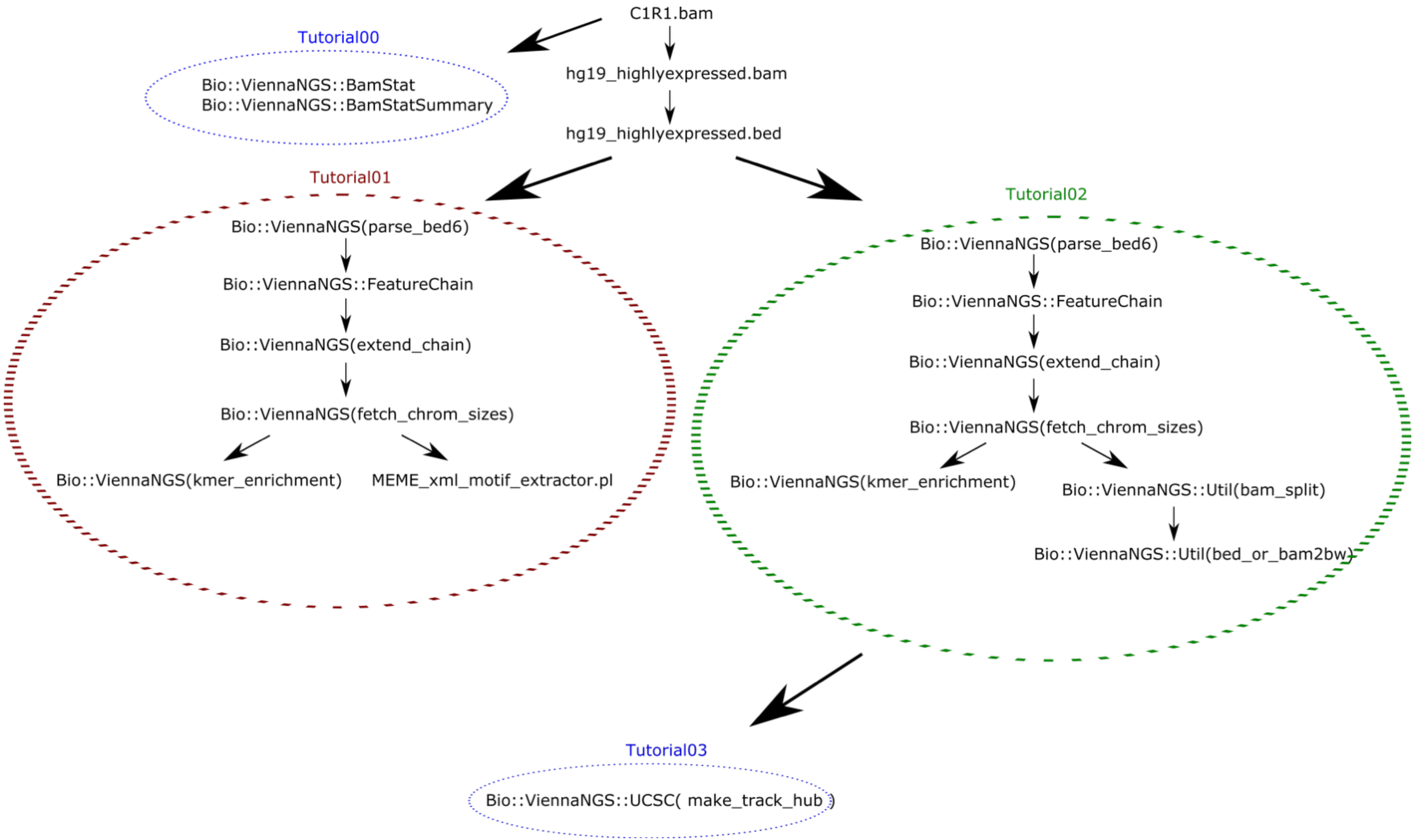
ViennaNGS Availability

The ViennaNGS Perl distribution is available from GitHub & CPAN

https://github.com/mtw/Bio-ViennaNGS

http://search.cpan.org/dist/Bio-ViennaNGS

“ViennaNGS: A toolbox for building efficient next-generation sequencing analysis pipelines”
M.T. Wolfinger, J. Fallmann, F. Eggenhofer, F. Amman
bioRxiv preprint DOI:10.1101/013011


C1R1.bam
Tutorial00
Bio::ViennaNGS::BamStat
Bio::ViennaNGS::BamStatSummary
hg19_highlyexpressed.bam
hg19_highlyexpressed.bed
Tutorial01
Bio::ViennaNGS(parse_bed6)
Bio::ViennaNGS::FeatureChain
Bio::ViennaNGS(extend_chain)
Bio::ViennaNGS(fetch_chrom_sizes)
Bio::ViennaNGS(kmer_enrichment)
MEME_xml_motif_extractor.pl
Tutorial02
Bio::ViennaNGS(parse_bed6)
Bio::ViennaNGS::FeatureChain
Bio::ViennaNGS(extend_chain)
Bio::ViennaNGS(fetch_chrom_sizes)
Bio::ViennaNGS(kmer_enrichment)
Bio::ViennaNGS::Util(bam_split)
Bio::ViennaNGS::Util(bed_or_bam2bw)
Tutorial03
Bio::ViennaNGS::UCSC(make_track_hub)