Julia简易教程之BioStructures

前言

Julia是一种高级通用动态编程语言，它最初是为了满足高性能数值分析和计算科学的需要而设计的，不需要解释器，速度快，也可用于客户端和服务器的Web用途、低级系统编程或用作规约语言。

使用**Julia**处理生物数据具有速度快、并行易等优势。本文档将介绍Julia中的BioStructures在处理如Protein Data Bank (PDB), mmCIF 和 MMTF等格式蛋白质结构数据中的基本用法。

安装Julia和BioStructures

Step.1 下载并安装Julia

预编译的二进制文件是安装Julia的推荐方法，但如果需要，也可以选择从源代码编译Julia。在本教程中，将从Julia的官方下载页面下载官方预编译的二进制文件。请确保位于主目录中，然后开始下载：

```
wget https://julialang-s3.julialang.org/bin/linux/x64/1.8/julia-1.8.1-linux-x86_64.tar.gz
```

如果下载过慢，则也可以在网页Julia官方网站上下载好上传至集群（选择Linux x86版本）

Current stable release: v1.9.0 (May 7, 2023)

Checksums for this release are available in both MD5 and SHA256 formats.

Windows [help]	64-bit (installer), 64-bit (portable)		32-bit (installer), 32-bit (portable)
macOS x86 (Intel or Rosetta) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
macOS (Apple Silicon) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
Generic Linux on x86 [help]	64-bit (glibc) (GPG), 64-bit (musl)[1] (GPG)		32-bit (GPG)
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)		
Generic Linux on PowerPC [help]	64-bit (little endian) (GPG)		
Generic FreeBSD on x86 [help]	64-bit (GPG)		
Source	Tarball (GPG)	Tarball with dependencies (GPG)	GitHub

然后解压

```
tar zxvf julia-1.8.1-linux-x86_64.tar.gz #请注意修改下载的Julia版本
```

Step.2 添加Julia到环境变量

修改 `bashrc`

```
vim ~/.bashrc
```

添加Julia到环境变量

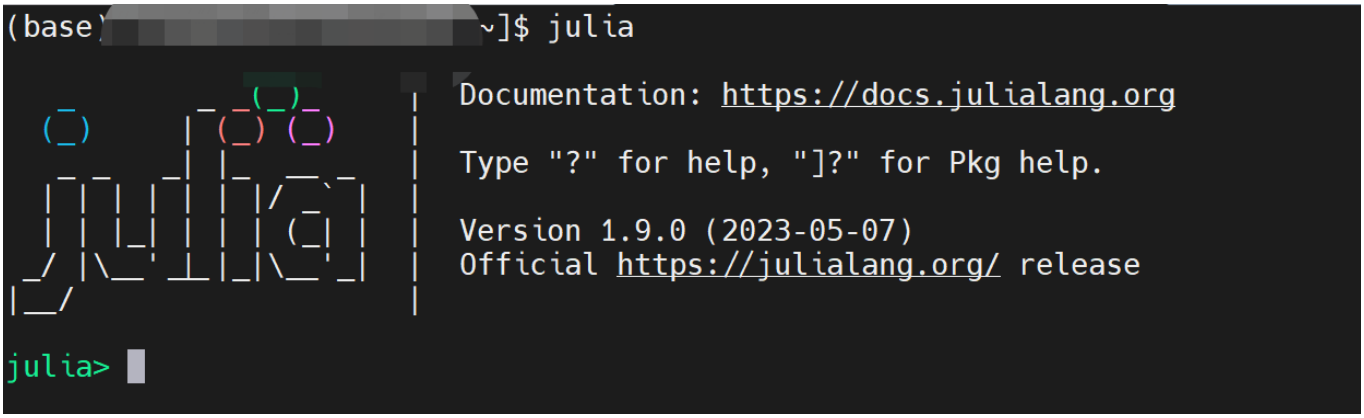
```
export PATH=$PATH:/home/your/home/path/julia-1.8.1/bin #修改安装路径和Julia版本
```

执行bashrc

```
source ~/.bashrc
```

Step.3 运行Julia REPL并安装BioStructures

在命令行中输入Julia，如果前面的安装都没有问题的话，则可以看到如下界面



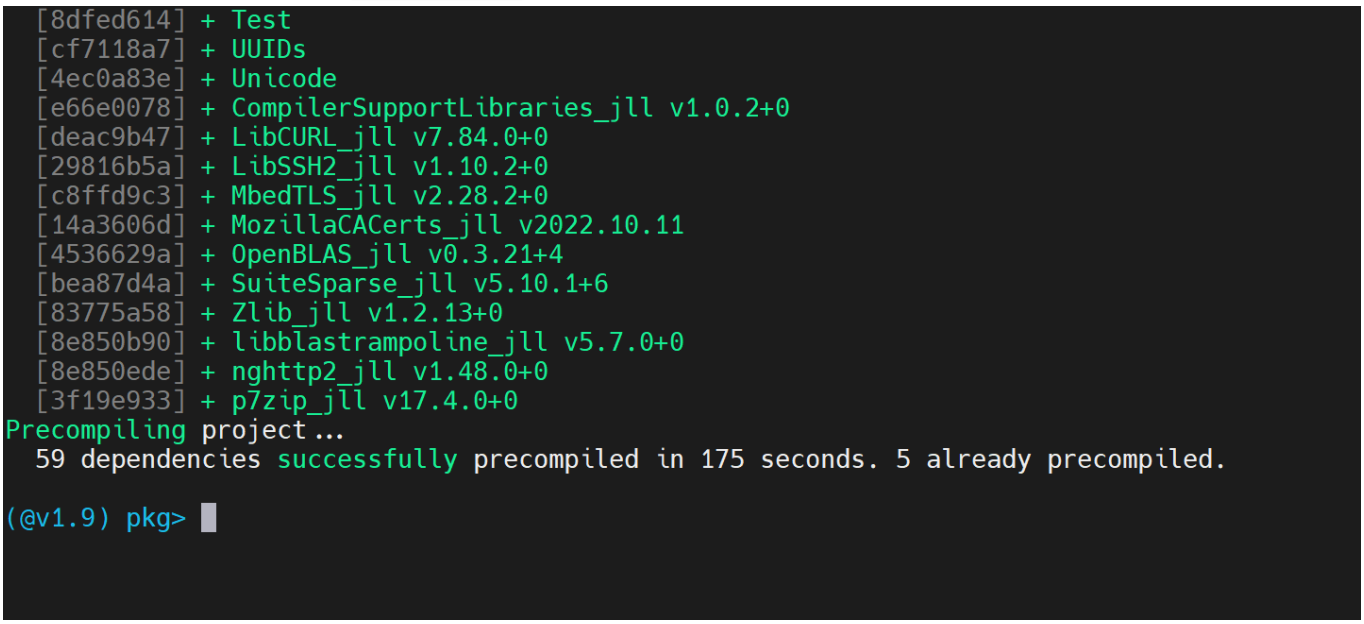
接下来安装BioStructures

键入 `]`，会看到 原来的julia变成了pkg，表示进入安装包的状态。再键入

```
add BioStructures
```

等待安装完成。此外还有一些常用的包比如 `CSV`，`Parse`，`DataFrames`，`Tables` 等都可以通过 `add` 来安装。

安装好的界面如下，`Backspace` 退出安装



至此**Julia**和**BioStructures**就安装结束了

BioStrutures简介

BioStructures.jl包提供了操作大分子结构的功能，特别是读取和写入蛋白质数据库（PDB）、mmCIF和MMTF文件的功能。它被设计用于标准结构分析任务，以及作为一个平台，其他人可以在其上构建以创建更具体的工具。在性能方面是要优于其他的PDB解析器的。

和Python一样，可以在 REPL中交互运行，也可以把程序写成可执行文件运行（`julia juliafilename.jl`）。本文档使用REPL演示BioStructures的常用函数，使用Julia脚本来绘制contactmap。

BioStrutures使用基础

下载PDB文件

```
using BioStructures

# Stored in the current working directory by default
downloadpdb("1EN2")
```

将PDB文件解析为Model-Chain-Residue-Atom框架

```
julia> struc = read("/path/to/pdb/file.pdb", PDB)
ProteinStructure 1EN2.pdb with 1 models, 1 chains (A), 85 residues,
754 atoms
```

mmCIF文件可以通过 `read（"/path/to/cif/file.cif", mmCIF）` 读取到相同的数据结构中。关键字参数gzip（默认值为false）确定文件是否使用了gzip。

struc的元素可以访问如下

命令	返回值	返回类型
`struc[1]`	Model 1	`Model`
`struc[1]["A"]`	Model 1, chain A	`Chain`

命令	返回值	返回类型
`struc[1]['A']`	Shortcut to above if the chain ID is a single character	`Chain`
`struc["A"]`	The lowest model (model 1), chain A	`Chain`
`struc["A"]["50"]`	Model 1, chain A, residue 50	`AbstractResidue`
`struc["A"][50]`	Shortcut to above if it is not a hetero residue and the insertion code is blank	`AbstractResidue`
`struc["A"]["H_90"]`	Model 1, chain A, hetero residue 90	`AbstractResidue`
`struc["A"][50]["CA"]`	Model 1, chain A, residue 50, atom name CA	`AbstractAtom`
`struc["A"][15]["CG"]['A']`	For disordered atoms, access a specific location	`Atom`

按如下方式检索属性：

函数	返回值	返回类型
`serial`	原子编号	`Int`
`atomname`	原子名称	`String`
`altlocid`	原子的替代位置ID	`Char`
`altlocids`	`DisorderedAtom` 的所有替代位置ID	`Array{Char,1}`
`x`	原子的 x 坐标	`Float64`
`y`	原子的 y 坐标	`Float64`
`z`	原子的 z 坐标	`Float64`
`coords`	原子坐标	`Array{Float64,1}`
`occupancy`	原子占有率 (默认为 `1.0`)	`Float64`

函数	返回值	返回类型
`tempfactor`	原子的温度因子 (默认为 `0.0`)	`Float64`
`element`	原子元素 (默认为 `""`)	`String`
`charge`	原子电荷 (默认为 `""`)	`String`
`residue`	原子归属残基	`Residue`
`ishetero`	残基或原子是否是杂残基/原子	`Bool`
`isdisorderedatom`	是否为无序	`Bool`
`pdbline`	原子的PDB ATOM/HETATM记录	`String`
`resname`	残基或原子的残基名	`String`
`resnames`	所有在 `DisorderedResidue` 的残基名	`Array{String,1}`
`resnumber`	原子或残基的残基编号	`Int`
`sequentialresidues`	判断下一个残基是否和上一个残基连续	`Bool`
`inscode`	原子或残基的插入码	`Char`
`resid`	原子或残基的残基ID	`String`
`atomnames`	残基里所有的原子名称	`Array{String,1}`
`atoms`	残基中原子的字典	`Dict{String,AbstractAtom}`
`isdisorderedres`	残基是否有多个残基名	`Bool`

函数	返回值	返回类型
`disorderedres`	访问 `DisorderedResiduce` 中的特定残基名	`Residue`
`chain`	残基或原子所属的链	`Chain`
`chainid`	链、残基或原子的链 ID	`String`
`resids`	链中已排序的残基ID	`Array{String,1}`
`residues`	链中残基的字典	`Dict{String,AbstractResidue}`
`model`	链、残基或原子所属的模型	`Model`
`modelnumber`	模型、链、残基或原子的模型编号	`Int`
`chainids`	模型或结构的排序的链 ID	`Array{String,1}`
`chains`	模型或结构中链的字典	`Dict{String,Chain}`
`structure`	模型、链、残基或原子所属的结构	`ProteinStructure`
`structurename`	元素所属结构的名称	`String`
`modelnumbers`	结构的排序的模型	`Array{Int,1}`
`models`	结构中模型的字典	`Dict{Int,Model}`

操纵结构

可以对一个struc进行如下循环，得到原子并进行操作

```
for mod in struc
    for ch in mod
```

```
        for res in ch
            for at in res
                # Do something
            end
         end
    end
end
```

`collect` 可以用于获取子元素的数组。`collectatoms`、`collectresibutes`、`collectchains` 和 `collectmodels` 从结构元素或元素数组返回特定类型的数组。

命令	操作	返回类型
`collect(struc['A'][50])`	Collect the sub-elements of an element, e.g. atoms from a residue	`Array{AbstractAtom,1}`
`collectresidues(struc)`	Collect the residues of an element	`Array{AbstractResidue,1}`
`collectatoms(struc)`	Collect the atoms of an element	`Array{AbstractAtom,1}`
`collectatoms(struc, calphaselector)`	Collect the Cα atoms of an element	`Array{AbstractAtom,1}`
`collectatoms(struc, calphaselector, disorderselector)`	Collect the disordered Cα atoms of an element	`Array{AbstractAtom,1}`

可用的选择器如下：

Function	Acts on	Selects for
`standardselector`	`AbstractAtom` or `AbstractResidue`	Atoms/residues arising from standard (ATOM) records
`heteroselector`	`AbstractAtom` or `AbstractResidue`	Atoms/residues arising from hetero (HETATM) records
`atomnameselector`	`AbstractAtom`	Atoms with atom name in a given list
`calphaselector`	`AbstractAtom`	Cα atoms
`cbetaselector`	`AbstractAtom`	Cβ atoms, or Cα atoms for glycine residues
`backboneselector`	`AbstractAtom`	Atoms in the protein backbone (CA, N, C and O)
`heavyatomselector`	`AbstractAtom`	Non-hydrogen atoms
`hydrogenselector`	`AbstractAtom`	Hydrogen atoms
`resnameselector`	`AbstractAtom` or `AbstractResidue`	Atoms/residues with residue name in a given list
`waterselector`	`AbstractAtom` or `AbstractResidue`	Atoms/residues with residue name HOH
`notwaterselector`	`AbstractAtom` or `AbstractResidue`	Atoms/residues with residue name not HOH
`disorderselector`	`AbstractAtom` or `AbstractResidue`	Atoms/residues with alternative locations

Function	Acts on	Selects for
`allselector`	`AbstractAtom` or `AbstractResidue`	All atoms/residues

蛋白质的氨基酸序列可以通过使用任选的残基选择器将元件传递到 `LongAA` 来检索：

```
julia> LongAA(struc['A'], standardselector)
85aa Amino Acid Sequence:
RCGSQGGGSTCPGLRCCSIWGWCGDSEPYCGRTCENKCW…
RCGAAVGNPPCGQDRCCSVHGWCGGGNDYCSGGNCQYRC
```

`gaps` 关键字参数决定是否根据缺失的残数将 `gaps` 添加到序列中（默认为true）。如果需要的话，`three-letter_toaa` 提供了氨基酸代码的查找表。有关如何处理序列的更多信息，请参见 `BioSequences.jl` 和 `BioAlignments.jl`。`LongAA` 是 `LongSequence{AminoAcidAlphabet}` 的别名。例如，要查看CDK1和CDK2的对齐（此示例还利用了Julia的广播）：

```
julia> struc1, struc2 = retrievepdb.(["4YC6", "1HCL"])
2-element Array{ProteinStructure,1}:
 ProteinStructure 4YC6.pdb with 1 models, 8 chains (A,B,C,D,E,F,G,H),
1420 residues, 12271 atoms
 ProteinStructure 1HCL.pdb with 1 models, 1 chains (A), 294 residues,
2546 atoms

julia> seq1, seq2 = LongAA.([struc1["A"], struc2["A"]],
standardselector, gaps=false)
2-element Vector{LongAA}:
 MEDYTKIEKIGEGTYGVVYKGRHKTTGQVVAMKKIRLES…
SHVKNLDENGLDLLSKMLIYDPAKRISGKMALNHPYFND
 MENFQKVEKIGEGTYGVVYKARNKLTGEVVALKKIRTEG…
RSLLSQMLHYDPNKRISAKAALAHPFFQDVTKPVPHLRL

julia> using BioAlignments

julia> scoremodel = AffineGapScoreModel(BLOSUM62, gap_open=-10,
```

```
gap_extend=-1);

julia> alres = pairalign(GlobalAlignment(), seq1, seq2, scoremodel)
PairwiseAlignmentResult{Int64, LongAA, LongAA}:
  score: 945
  seq:   1
MEDYTKIEKIGEGTYGVVYKGRHKTTGQVVAMKKIRLESEEEGVPSTAIREISLLKELRH  60
           ||   | |||||||||||| | | || ||| |||| |    ||||||||||||||
|
  ref:   1 MENFQKVEKIGEGTYGVVYKARNKLTGEVVALKKIRTE----
GVPSTAIREISLLKELNH  56

  seq:  61 PNIVSLQDVLMQDSRLYLIFEFLSMDLKKYLD-
SIPPGQYMDSSLVKSYLYQILQGIVFC 119
           |||| | ||      ||| ||||  ||||  | |   |      | |||| | |||
||
  ref:  57 PNIVKLLDVIHTENKLYLVFEFLHQDLKKFMDASALTG--
IPLPLIKSYLFQLLQGLAFC 114

  seq: 120 HSRRVLHRDLKPQNLLIDDKGTIKLADFGLARAFGV----
YTHEVVTLWYRSPEVLLGSA 175
           || ||||||||||||   | ||||||||||||    |||||||||| || |||
  ref: 115
HSHRVLHRDLKPQNLLINTEGAIKLADFGLARAFGVPVRTYTHEVVTLWYRAPEILLGCK 174

  seq: 176
RYSTPVDIWSIGTIFAELATKKPLFHGDSEIDQLFRIFRALGTPNNEVWPEVESLQDYKN 235
           ||| ||||| | ||||  |   || |||||||||||| ||||   ||| | |  |||
  ref: 175
YYSTAVDIWSLGCIFAEMVTRRALFPGDSEIDQLFRIFRTLGTPDEVVWPGVTSMPDYKP 234

  seq: 236 TFPKWKPGSLASHVKNLDENGLDLLSKMLIYDPAKRISGKMALNHPYFND---------
- 285
           ||||        |  ||| |  ||| || ||| |||| | || || | |
  ref: 235
SFPKWARQDFSKVVPPLDEDGRSLLSQMLHYDPNKRISAKAALAHPFFQDVTKPVPHLRL 294
```

空间计算

提供了各种函数来计算蛋白质的空间量：

Command	Returns
`distance`	Minimum distance between two elements
`sqdistance`	Minimum square distance between two elements
`coordarray`	Atomic coordinates in Å of an element as a 2D `Array` with each column corresponding to one atom
`bondangle`	Angle between three atoms
`dihedralangle`	Dihedral angle defined by four atoms
`omegaangle`	Omega dihedral angle between a residue and the previous residue
`phiangle`	Phi dihedral angle between a residue and the previous residue
`psiangle`	Psi dihedral angle between a residue and the next residue
`omegaangles`	`Vector` of omega dihedral angles of an element
`phiangles`	`Vector` of phi dihedral angles of an element
`psiangles`	`Vector` of psi dihedral angles of an element
`ramachandranangles`	`Vector`s of phi and psi angles of an element
`ContactMap`	`ContactMap` of two elements, or one element with itself
`DistanceMap`	`DistanceMap` of two elements, or one element with itself
`showcontactmap`	Print a representation of a `ContactMap` to `stdout` or a specified `IO` instance
`Transformation`	The 3D transformation to map one set of coordinates onto another
`applytransform!`	Modify all coordinates in an element according to a transformation

Command	Returns
`applytransform`	Modify coordinates according to a transformation
`superimpose!`	Superimpose one element onto another
`rmsd`	RMSD between two elements, with or without superimposition
`displacements`	`Vector` of displacements between two elements, with or without superimposition
`MetaGraph`	Construct a `MetaGraph` of contacting elements

`omegaangle`、`phiangle`和`psiangl`e函数可以取一对残基，也可以取一个链和一个位置。`omegaangle`和`phiangle`函数测量给定索引处的残基与之前的残基之间的角度。`psiangle`函数在给定指数和之后的指数之间进行测量。例如：

```
julia> distance(struc['A'][10], struc['A'][20])
10.782158874733762

julia> rad2deg(bondangle(struc['A'][50]["N"], struc['A'][50]["CA"],
struc['A'][50]["C"]))
110.77765846083398

julia> rad2deg(dihedralangle(struc['A'][50]["N"], struc['A'][50]
["CA"], struc['A'][50]["C"], struc['A'][51]["N"]))
-177.38288114072924

julia> rad2deg(psiangle(struc['A'][50], struc['A'][51]))
-177.38288114072924

julia> rad2deg(psiangle(struc['A'], 50))
-177.38288114072924
```

结构元素可以叠加，并且可以计算叠加相关属性，例如RMSD。为了进行叠加，`BioStructures.jl`进行序列比对，并使用Kabsch算法叠加比对的残基。例如：

```
# Change the coordinates of element 1 to superimpose it onto element 2
# Do sequence alignment with standard residues and calculate the
```

```
transformation with Cα atoms (the default)
superimpose!(el1, el2, standardselector)

# The transformation object for the above superimposition
Transformation(el1, el2, standardselector)

# Calculate the transformation with backbone atoms
superimpose!(el1, el2, standardselector, alignatoms=backboneselector)

# Calculate RMSD on Cα atoms (the default) after superimposition
rmsd(el1, el2, standardselector)

# Superimpose based on backbone atoms and calculate RMSD based on Cβ
atoms
rmsd(el1, el2, standardselector, alignatoms=backboneselector,
rmsdatoms=cbetaselector)

# Do not do a superimposition - assumes the elements are already
superimposed
rmsd(el1, el2, standardselector, superimpose=false)
```

写出PDB文件

```
writepdb("filename.pdb", struc)
```

可视化结构

`Bio3DView.jl`软件包可用于可视化分子结构。例如：

```
using Bio3DView
using Blink
viewpdb("1CRN")
```

案例展示：使用 Julia 绘制蛋白质残基接触图

ContactMap表示使用二元二维矩阵的表示蛋白质结构的所有残基对之间的临近关系。如果在给定阈值 *t* 内，如果残基 *i* 与 *j* 之间的距离 *d* 小于 *t* , 那么接触图的a_{ij}及 a_{ji} 均为 1，反之则为 0。

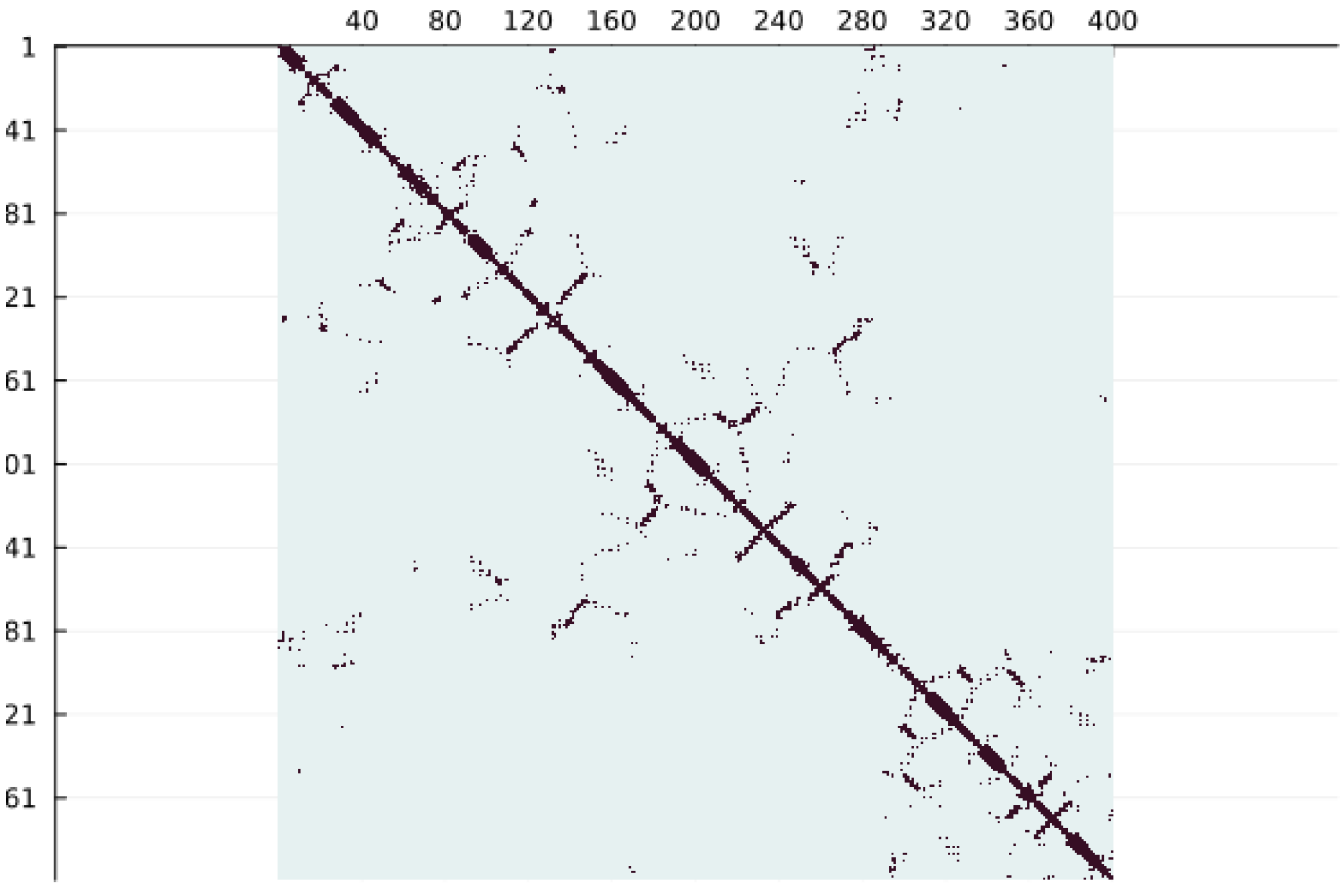
Julia 中的 `ContactMap` 函数可用来绘制残基接触图。 `ContactMap` 接受结构元素或列表，如 `Chain` 或 `Vector` ，并返回一个 `ContactMap` 对象，显示指定距离内元素之间的接触。 `ContactMap` 还可以被赋予两个结构元素作为参数，在这种情况下，返回一个非对称的2D数组，显示元素之间的接触。如果需要，可以使用 `contacts.data` 访问 `ContactMap` `contacts` 。

安装好 `BioStructures` 和 `Plots` 包，将以下文件保存为 `contactmap.jl`

```
using Plots
using BioStructures

PDBID=ARGS[1]
Chainid=ARGS[2]
threshold=parse(Float64, ARGS[3])
downloadpdb(PDBID)
struc = read(PDBID * ".pdb", PDB)
contacts = ContactMap(collectatoms(struc["$Chainid"], cbetaselector),
threshold)
p=plot(contacts)
savefig(p,PDBID * "-" * Chainid * "-" * "$threshold" * ".png")
```

使用方法：`julia contactmap.jl PDBID Chainid threshold`
例如：`julia contactmap.jl 1GVE A 8` 表示将PDBID 为1GVE的蛋白质结构文件下载，取其A链，并以8Å为阈值绘制contactmap



参考资料

1. Julia (programming language) - Wikipedia
2. Documentation · BioStructures.jl (biojulia.dev)