VO 講習会 2015a： AGN と銀河のクラスタリング測定

ver. 2015-02-20

白崎 裕治 [1]

1 本課題の目的

本課題では、VO から取得したデータを利用して、AGN と銀河のクラスタリング度の測定することと通じて、主としてプログラミングを行うことにより大量の VO データを扱う技法を取得することを目的とする。

VO を活用する利点としては大きく分けて２点ある。ひとつは、世界中の種々多様な大量のデータの中から、必要なデータを簡単に見つけ出することができる点である。これは、VO ポータルサイトや、VO 対応のデスクトップアプリケーションを利用することにより実現でき、共通のユーザインターフェイスで様々なデータアーカイブに検索を実行することができる

もうひとつの利点としては、データサービスに対して利用者独自のデータアクセス方式をプログラミングすることができることにある。この利用方式では、大量のデータを自動で取得し、それを自分の解析環境と連携することが可能となり、単に簡単にデータを探すといった段階から一歩すすんだ、データアーカイブの利用方法を提供する。

本課題では、後者の利点を最大限に活用できるよう、VO へのアクセスをコマンドラインから行う方法を学び、さらにシェルスクリプト言語やプロットツールも駆使して研究を行う方法についてマスターすることを目的としている。

そのために、研究課題として、AGN と銀河のクラスタリング度から AGN が属するダークマターハローの質量を推定するというテーマを設定し、この課題を VO も利用して実施する。この利用方式では、単純に VO アクセス方法だけをマスターするだけではなくシェルスクリプト等によるワークフローの記述方法や、解析結果の可視化で必要となるプロットの自動生成の方法についてもマスターする必要があるめ、そうした方法についても学ぶ。

2 研究の背景

我々の銀河系を含め、大部分の銀河にはその中心に質量が $10^6 M_\odot$ を超える大質量ブラックホールが存在することが明らかになってきている。また、そのブラックホール質量は銀河バルジの速度分散と強い相関があることが分かってきており、銀河の星形成と大質量ブラックホールの形成は密接にリンクしていると考えられる。そうした大質量ブラックホールの形成シナリオが、多くの研究者によって提案されている。有力な理論モデルの一つによると、まず、銀河同士の衝突・合体により星形成が活発に行われることで大質量星が多数誕生し、それらの重力崩壊により中質量ブラックホールが多数形成される。それら中質量ブラックホールが銀河中心部に落ち込み合体することで、大質量ブラックホールが形成されると考えられている。このように考えることにより、銀河バルジと大質量ブラックホールの共進化がうまく説明できる。したがって、銀河同士の衝突が起こりやすい銀河が密集した領域では、大質量ブラックホールが活発に成長していると考えられる。

[1]国立天文台 National Astronomical Observatory of Japan

銀河中心周辺のガスが大質量ブラックホールに落ち込む際には強力な放射をともない、活動銀河中心核 (AGN) と呼ばれる天体として観測される。その放射強度は、太陽系程度の広さの領域から銀河全体に匹敵またはそれを凌駕するほどのエネルギーに達するものも存在する。特に放射強度の高い AGN はクエーサーとよばれ、その放射強度を説明するには銀河全体の質量に匹敵する量のガスが短時間の間にブラックホールに流入する必要があり、銀河同士の合体のような激しい相互作用がクエーサーの明るさの起源であると考えられている。

AGN が実際に銀河数密度の高いところに発生しているかは、AGN と銀河の相互相関関数を求めることにより知ることができる。求められた相互相関関数を通常の銀河の自己相関関数とくらべ、優位に大きな値を示すなら AGN は銀河の密集した環境に発生したと考えることができる。また、この相互相関関数をダークマターの自己相関関数と比較することにより、バイアスパラメータが求められると、AGN が含まれているダークマターハローの質量について推定することが可能であり、宇宙の大規模構造形成過程のどの段階でどれだけの大質量ブラックホールあるいは AGN/QSO が発生するのかについての知見が得られる。

3 AGN と銀河のクラスタリング測定方法

AGN と銀河の相互相関係数の測定方法について説明する [1, 3]。

銀河の空間分布を定量的に表すための指標として、銀河の 2 点相関関数と呼ばれる統計量がよく使われる。2 点相関関数の定義は以下の通りである。距離 r だけ離れた任意の 2 点 x_1, x_2 を考え、それぞれの点を中心とする微小体積 d^3x_1,d^3x_2 内に銀河が含まれる確率を $P(r)d^3x_1d^3x_2$ とする。銀河が一様に分布しているならば、$P(r)$ は r によらず一定値となり、銀河の数密度分布 $\overline{n}^2$ に等しくなる。実際の宇宙では銀河は一様に分布しているのではなく、銀河団等のように局所的に密集している領域もあれば、ボイドのように銀河が少なくなっている領域もある。したがって、$P(r)$ は $\overline{n}^2$ からずれが生じる。そのずれの量を $\xi(r)$ として、$P(r) = \overline{n}^2(1+\xi(r))$ で定義される $\xi(r)$ が相関関数と呼ばれる統計量である。

$P(r)$ を求めるには、愚直に全ての x_1, x_2 の組み合わせに対して両者に銀河が含まれているかを判断すればよいが、この方法だと計算の効率が極めて悪いため、通常は次のように求められる。銀河が存在する場所 (座標 x とする) から r だけ離れた微小体積内に銀河がある確率 $P(r;x)$ を求め、この条件付確率から $P(r) = \overline{n}P(r;x)$ によって $P(r)$ を求める。$P(r;x)$ は、特定の銀河に着目した時に求められる、そこから $r \sim r+dr$ だけ離れた場所にある銀河のカウント数を、多数の銀河について求め平均化した値 $N(r)$ より、$P(r;x) = \frac{1}{4\pi r^2 dr}N(r)$ のように求めることができる。したがって、相関関数は

$$\xi(r) = \frac{N(r)}{4\pi r^2 \overline{n} dr} - 1 = \frac{\rho(r)}{\rho_0} - 1 \tag{1}$$

と与えられる。ここで、$\rho(r)$ は銀河から r の距離における銀河数密度、ρ_0 は銀河数密度の平均値である。

以上は銀河同士の自己相関関数の求め方であるが、AGN と銀河の相互相関関数も同様にして、銀河の数密度を AGN からの距離の関数として求め、それをその赤方偏移における銀河数密度で割った値から 1 を引くことで求めることができる。今回のデータでは銀河までの距離は求められていなので、相関関数を視線方向に積分した量である射影相関関数 $\omega(r_p)$ が求められる統計量であり、次のように定義される。

$$\omega(r_p) = 2\int_0^\infty \xi(r_p,\pi)d\pi = 2\int_{r_p}^\infty rdr\xi(r)(r^2-r_p^2)^{-0.5}, \tag{2}$$

これまでの観測から $\xi(r)$ は r の冪関数で表されることが分かっており、$\xi(r) = (r_0/r)^\gamma$ とすると、式 (2) は解析的に積分することができ、

$$\omega(r_p) = r_p \left(\frac{r_0}{r_p}\right)^\gamma \frac{\Gamma(1/2)\Gamma((\gamma-1)/2)}{\Gamma(\gamma/2)}, \tag{3}$$

となる。ここで、 Γ はガンマ関数である。観測量から $\omega(r_p)$ を求めるには、式 (1) を積分して得られる次の関係式を利用する。

$$\omega(r_p) = \frac{1}{\rho_0}\int_{-\infty}^{\infty}(\rho(r)-\rho_0)d\pi = \frac{n(r_p)-n_{\rm bg}}{\rho_0}, \tag{4}$$

ここで $\rho(r)$ と ρ_0 の積分値はそれぞれ、AGN 近傍における銀河の射影数密度 $n(r_p)$ と AGN から十分離れた距離にある前景・背景銀河の数密度 $n_{\rm bg}$ で置き換えた。

AGN と銀河の相互相関関数が求められると、それからホストダークマターハローの質量 $M_{\rm h}$ が推定できる (例えば [4, 5, 6, 7, 8, 9, 16])。大まかな計算の流れは以下のとおりである。まず、バイアスパラメータ b を以下のように求める。

$$b = \frac{\sigma_{8,\rm AGN}}{\sigma_{8,\rm DM}} \tag{5}$$

$\sigma_{8,\rm AGN}$, $\sigma_{8,\rm DM}$ はそれぞれ、AGN, ダークマター (DM) の 8 $\rm h^{-1}$Mpc 距離内での相関関数の二乗平均の平方根であり、。

$$\sigma_{8,\rm AGN} = J_2(\gamma)^{1/2}\left(\frac{r_{\rm AA}}{8}\right)^{\gamma/2}, \tag{6}$$

$$J_2(\gamma) = \frac{72}{(3-\gamma)(4-\gamma)(6-\gamma)2^\gamma}, \tag{7}$$

$$\sigma_{8,\rm DM} = \sigma_8 \frac{D(z)}{D(0)}, \tag{8}$$

$$D(z) = \frac{5\Omega_{\rm m}E(z)}{2}\int_z^\infty \frac{1+y}{E^3(y)}dy, \tag{9}$$

$$E(z)^2 = \Omega_m(1+z)^3 + \Omega_\Lambda \tag{10}$$

によって計算できる。ここで、$r_{\rm AA}$ は AGN の自己相関距離であり、銀河と AGN の相互相関関数と銀河の自己相関関数が同じべき指数 γ を持つべき関数で表されるとすると、$r_{\rm AA} = r_{\rm AG}^2/r_{\rm GG}$ である。$r_{\rm AA}$ は AGN の自己相関距離、$r_{\rm AG}$ は AGN と銀河の相互相関距離 r_0 である。

以上の式からバイアスパラメータは、

$$b = \left(\frac{r_{\rm AA}}{8}\right)^{\gamma/2} J_2(\gamma)^{1/2}\left(\frac{\sigma_8 D(z)}{D(0)}\right)^{-1} \tag{11}$$

と計算される。

バイアスパラメータの値とダークマターハローの質量との間の関係式は次のように表される [15]。

$$b = 1 + \frac{1}{\sqrt{a}\delta_{sc}(z)}\left[\sqrt{a}(a\nu^2) + \sqrt{a}b(a\nu^2)^{1-c} - \frac{(a\nu\ 2)^c}{(a\nu^2)^c + b(1-c)(1-c/2)}\right] \tag{12}$$

ここで a = 0.707, b = 0.5, c = 0.6, $\nu = \sigma_{sc}(z)/\sigma(M_{\rm h})$ であり、

$$\delta_{sc}(z) = \frac{0.15(12\pi)^{2/3}\Omega(z)^p}{D(z)}, \tag{13}$$

$$\Omega(z) = \frac{\Omega_m(1+z)^3}{\Omega_\Lambda + \Omega_m(1+z)^3}. \tag{14}$$

$$\sigma(M_{\rm h}) = \sigma_8 \frac{f(u)}{f(u_8)} \tag{15}$$

ここで、$u_8 = 32\Gamma$, $\Gamma = 0.173$,

$$u = 3.804 \times 10^{-4}\Gamma \left(\frac{M_h}{\Omega_m}\right)^{1/3}, \tag{16}$$

$$f(u) = 64.087(1 + 1.074u^0.3 - 1.581u^{0.4} + 0.954u^{0.5} - 0.185u^{0.6})^{-10} \tag{17}$$

である [13]。

4 利用するデータとソウフトウエア

4.1 利用するデータベース

本課題では、Veron QSO & AGN カタログ第 13 版 [14] と UKIDSS DR8 カタログ [1] を利用する。データサービスの identifier はそれぞれ、ivo://jvo/agn, ivo://jvo/ukidss である。これらのカタログを提供するサービスは多数あるが、ADQL 検索が可能で複数天体一括検索機能がある JVO のサービスを利用する。

4.2 利用するソフトウエア

ROOT, jc クライアント [2], 講習会用ソフトウエアパッケージを利用する。

ROOT は次の URL からダウンロードしインストールすること。`http://root.cern.ch/drupal/`. 講習会用に用意された VMWare イメージを利用する場合には、既にインストール済みであるので、自分でインストールを行う必要はない。

jc クライアントは `http://jvo.nao.ac.jp/jc_client` から最新版が取得できる。インストール方法は配布パッケージに含まれる INSTALL ファイルに記述されている。展開すると `jc_client` というディレクトリが作成されるので、そこに移動し `install.sh` を実行するだけである。

```
1  $ tar -xzf jc_client.<version>.tgz
2  $ cd jc_clinet
3  $ ./install.sh
```

ホームディレクトリ直下の `jvo_tools` にインストールされる。jc クライアントの実行には Oracle java がインストールされている必要がある [3]。java がインストールされていない場合は、次の URL からダウンロードしインストールすること。`http://www.oracle.com/technetwork/java/javase/downloads/index.html`. インストール後は java の実行ファイルが格納されているディレクトリを環境変数 `PATH` に追加する必要がある。例えば、`/usr/local/jdk` にインストールした場合は `export PATH=/usr/local/jdk/bin:${PATH}` を実行する。ログイン時に自動的に実行されるようにするには、この文をホームディレクトリ直下にある `.bashrc` に追加するとよい。講習会

[1] http://www.ukidss.org/

[2] jc は jvo command の略である。これまで jc コマンドと呼んでいたが、コマンドという単語が重複しているので jc クライアントと呼ぶようにする

[3] OS に付属のオープンソース系の Java でも動くようである

用 VMWare イメージには java は /usr/local/jdk/bin にインストール済みであり、実行ファイルがあるディレクトリもすでに ${PATH} に含まれているのでこの設定を行う必要はない。

講習用に用意した, C 言語 や perl、シェルスクリプトで書かれたプログラムを下記の URL よりダウンロードしインストールすること。http://jvo.nao.ac.jp/vos2015a/A/vo2015a.tgz。インストール方法は配布パッケージに含まれる INSTALL ファイルに記載されている。以下のコマンド例のように、展開し、作成されたディレクトリ vo2015a に移動し、コンパイルとインストールのコマンドを実行すると、~/jvo_tools にインストールされる。

```
$ tar -xzf vo2015a.tgz
$ cd vo2015a
$ ./configure
$ make
$ make install
$ cp lib/jvo-infra.jar ~/jvo_tools/lib
$ rsync -a classes/ ~/jvo_tools/classes/
```

以上で jc クライアントならびに講習会で必要となるプログラムが ~/jvo_tools にインストールされたはずである。インストールされてプログラムを利用できるようにするために環境設定を行うスクリプトを以下のように実行する必要がある。

```
$ . vo2015a/jvorc.sh
```

これで必要な環境変数が設定される。

4.3 jc クライアント

jc クライアントを利用するには JVO ポータルのユーザアカウントを取得する必要がある。JVO ポータル http://jvo.nao.ac.jp/portal の左枠一番下の “Registration” と書かれた文字のリンク先ページで必要事項を入力し、アカウント発行を行う。アカウントが取得できたら、jc クライアントが参照するパスワードファイルの作成を以下のように行う。

```
$ jc passwd
user: <ユーザ名>
password: <パスワード>
retype password: <パスワード>
access information has been updated.
```

パスワードファイルは ~/.jvocommand_passwd.txt に作成される。パスワードファイル中のパスワードは暗号化されているが、JVO ポータルにパスワードを送信する際には暗号化されていない文字列が渡される。暗号化したい場合には JVO ポータルのアクセス先 URL として https://jvo.nao.ac.jp/ を指定する必要があるが、講習会では http://jvo.nao.ac.jp/ を利用し、暗号化なしの環境で行う。

jc クライアントは そのプログラム名に引き続いてオペレーション名を記述し、各オペレーション毎に定義されている引数を与えて実行する。サポートされているオペレーション名の一覧は --help オプションを指定して jc コマンドを実行することで表示される。

```
$ jc --help
usage: jc [GLOBAL_OPT] ... COMMAND [COMMAND_OPT] ... COMMAND_ARG ...
GLOBAL_OPT:
    --help show this help
 -v,--verbose increase verbosity
```

```
COMMAND:
 abort abort executing tool
 conf configure account
 copy2l download files from VOSpace
 copy2v upload files to VOSpace
 delete alias of remove
 dummy print informations for debug
 get alias of copy2l
 join join votable file
 list list VOSpace nodes
 ls alias of list
 passwd alias of conf
 ps search exec tool
 put alias of copy2v
 registry service search
 remove remove VOSpace nodes -- ALPHA VERSION
 resume resume executing tool
 rm alias of remove
 rsync2l search for updated files and download them
 rsync2v search for updated files and upload them
 run run command with toolName
 search data search
 select display file
 suspend suspend executing tool
 union union file
 version show latest version of this tool
```

また、各オペレーションがとれる引数の一覧は、オペレーション名に引き続き--help オプションを指定したコマンド、jc <operation> --help を実行することにより表示される。

本課題では、search オペレーション と select オペレーションのみ利用する。search オペレーションがサポートする引数は以下のとおりである。

```
$ jc search -h
usage: jc [GLOBAL_OPT] ... search [OPTION] ...
OPTION:
-c,--center-coord <target | ra,dec> specify center coord by target |
                                    ra,dec .if you want multiple
                                    values, use -F option
-C print results as CSV
-f,--frame <coordinate> specify the coordinate system. you
                                    can only use the following argument
                                    as coordinate.[ FK4 | FK5 | IRC |
                                    GALACTIC ] : default is FK5
-F,--coords-file <fileName> specify a file that contains
                                    multiple center coordinates.
                                    coordinates are specified by target
                                    | ra,dec .one coordinate per one
                                    line.
-h,--help show this help
-i,--in <inputFile> specify the input file which
                                    contain JVOQL
-o,--out <fileName> specify the output file name
-p print only JVOQL without search
-r,--region <radius> specify the search radius. :
                                    default is 10.0
-s,--service <identifier> specify the identifier
-u,--unit <unit> specify the search radius unit. you
```

```
                                    can only use the following argument
                                    as coordinate.[ deg | arcmin |
                                    arcsec] you have to set search
                                    radius less than 1.0 degree:
                                    default is arcsec
  -V print results as VOTable
```

本課題では、-i オプションと -o オプションのみ使用する。JVOQL を ファイルに記述して、そのファイル名を -i オプションで指定する。検索結果の保存先ファイル名を -o オプションで指定する。

```
$ jc search -i <jvoql_file> -o <outout_file>
```

検索結果は XML フォーマットで記述される VOTable で書き出される。

VOTable 中の検索結果を CSV 形式で取り出すのに select オペレーションを使用する。select オペレーションがサポートする引数は以下のとおりである。

```
$ jc select -h
usage: jc [GLOBAL_OPT] ... select [OPTION] ...
OPTION:
 -c <arg> cut condition. if -f is votable, set votable column
                    names. default all column names.
 -f <arg> file type (only votable)
 -F <arg> field delimiter. default tab
 -h display the way how to use this command
 -o,--out <fileName> specify the output file name
 -v set file system (default local file system. if set
                    -v, vospace file system
```

本課題では -F, -o オプションのみを利用する。-F オプションでカラムの区切り文字を指定して、-o オプションで出力先ファイル名を指定する。最後に VOTable ファイルを指定する。

```
$ jc select -F, -o <csv_file> <votable_file>
```

4.4 JVOQL 文法

ここでは、JVO ポータルならびに jc クライアントで利用できる SQL の拡張仕様である JVOQL について、本課題で必要となる文法についてのみ解説する。

JVOQL の構文は以下の通りである。

JVOQL 構文 (基本構文 1)

```
<JVOQL> ::= SELECT [ DISTINCT ]
                   <select_item> [ , <select_item> ]*
          [ FROM   <table> [ , <table> ]* ]
          [ WHERE  <condition> ]
```

構文定義式は以下のルールに従って記述されている。

- 記号 ::= は左辺の構文を右辺で定義するという意味である。
- 大文字で示した単語は JVOQL 予約語で、そのまま記述する。

- 大文字、小文字は区別しない。
- 角括弧 [] 内の単語は省略可能である。
- アスタリクス * は複数回指定可能であることを示す。

上記の山括弧 <> で示される文の構文は以下のように定義される。

JVOQL 構文 (基本構文 2)

```
<select_item> ::= <カラム名> | "*"
<table> ::= <サービスの Identifier> : <テーブル名>
            [ [ AS ] <テーブルアリアス名> ]
<condition> ::= <評価値が boolean となる演算式>
<カラム名> ::= [ <テーブルアリアス名> "." ] カラム名
```

JVOQL の使い方を以下で説明する。

- 検索対象のテーブルまたはサービスの指定は、FROM 句中の<table> にテーブル識別子で記述する。テーブル識別子はサービスの Identifier とテーブル名をコロンでつなげて記述する。サービスによってはテーブル名が定義されていない場合があるが、その場合はサービスの Identifier の後にコロンを付けるだけでよい。
- テーブル識別子にはアリアス名を付けることができる。省略した場合は検索実行時に自動的に付与される。FROM 句において複数テーブルを指定した場合には、カラムの指定にテーブル修飾子が必要となるため、テーブルアリアス名の定義は必須となる。
- 取得したいカラムを限定して指定したい場合は、<select_item> でカラム名またはデータ名をカンマ区切りで指定する。全カラムを指定したい場合アスタリスクで指定する。FROM 句において複数テーブルを指定した場合は、カラム名をテーブルのアリアス名で修飾（テーブル修飾子）する必要がある。
- DISTINCT を指定すると、選択するレコードのうち値が重複するレコードは削除された結果を得ることができる。

条件式は SQL で利用される以下の構文が利用可能である。

JVOQL 構文 (条件式)

```
<condition> ::= [ <REGION 関数> ] [ [ AND ] <比較式> ]*
<比較式> ::= <カラム名>  <比較演算子> <数値>
<比較演算子> ::= "<" | ">" | "<=" | ">=" | "<>" | "!="
<REGION 関数> ::= REGION( <領域指定文> )
<領域指定文> ::= <円領域指定文>
<円領域指定文> ::= "'" CIRCLE <coord1> <coord2> <radius> "'"
```

JVOQL の記述例を以下に示す。

次の例は、Veron-Cetty et al. による Quasars and Active Galactic Nuclei Catalog (12th Ed.) に対し、座標赤経=30.0 度、赤緯=0 度を中心とする、半径 0.02 度の領域の QSO または AGN を検索する JVOQL である。

JVOQL の例 1 :

```
SELECT  agn.*
FROM    ivo://jvo/agn:veron_2010 agn,
WHERE   agn.vmag < 20
        AND Region('Circle 30 0 1.0')
```

同様に UKIDSS DR8 カタログに対する検索は次のようになる。

JVOQL の例 2 :

```
SELECT  ir.ra2000, ir.de2000
FROM    ivo://jvo/ukidss:catalog_dr8 ir
WHERE   Region('Circle 30 0 0.02')
```

この例は、UKIDSS DR8 カタログから例 1 と同じ領域を検索し、座標 ra2000, de2000 のみを取得する JVOQL である。

次の例は、同じ領域の AGN を検索し、その AGN のデータとその AGN　から 0.02 度の範囲にある UKIDSS カタログのデータを取得する JVOQL である。

JVOQL の例 3 :

```
SELECT  agn.*, ir.ra2000, ir.de2000
FROM    ivo://jvo/agn:veron_2010 agn,
        ivo://jvo/ukidss:catalog_dr8 ir
WHERE   Region('Circle 30 0 1.0')
        AND
        distance((ir.ra2000, ir.de2000), (agn.ra, agn.dec)) < 0.02
```

distance 関数は引数に天球上での位置を表す位置型の値を二つとり、それら二つの位置の角距離を度単位で返す。位置型の値は、二つの数値型のカラムまたは定数値を丸括弧内にカンマ区切りで列挙することで定義できる。この例では、UKIDSS カタログのカラム ra2000, de2000 で定義される位置型の値と、AGN カタログの ra, dec で定義される位置型との値を distance 関数の引数にとっている。カラムが位置型として定義されれている場合には直接そのカラムを引数としてとることができる。

最後に AGN を検索し、その AGN を含む Suprime-Cam のデータを検索する JVOQL の例を紹介する。

JVOQL の例 4 :

```
SELECT agn.*, img.*
FROM   ivo://jvo/agn:veron_2010 AS agn,
       ivo://jvo/subaru/spcam:image_cutout AS img
WHERE  agn.z >= 1.0 AND agn.z < 1.001
       AND img.region = Circle(agn.ra, agn.dec, 0.14)
```

img.region = Circle(qso.ra, qso.dec, 0.14) の条件式により、テーブル veron_2010 から取得した座標を中心とした 0.14 度の半径の領域で Suprime-Cam の画像検索用テーブルに対し検索を実行する。

4.5 ROOT

ROOT の使いかたについて簡単にまとめる。

ROOT は CERN で開発されているデータ解析フレームワークである。ROOT の主要な機能は、グラフや関数の作図、ヒストグラムの作成と操作、データーの任意関数によるフィットといったものがあげられる。また、大量のデータも処理可能であるという特徴がある。ROOT は C++ で開発されており、C++ のプログラム中で利用できるほか、C++ の文法に則ったマクロ機能がある。C++ の基本的な文法を覚えれば、マクロを作成してコマンドラインから簡単にプロットできるので、同じようなグラフを何度も描く場合に便利に使える。

次の URL からソースコードまたはバイナリをダウンロードできる: `http://root.cern.ch/drupal/content/downloading-root`。講習会用に配布される VMWare 上の Linux には既にインストール済みである。

コマンド `root` を実行すると、ROOT のインタラクティブ実行モードとなる。インタラクティブモードを終了するには `.q` を実行する。

```
$ root
  ****************************************
  * *
  * W E L C O M E to R O O T *
  * *
  * Version 5.28/00b 14 March 2011 *
  * *
  * You are welcome to visit our Web site *
  * http://root.cern.ch *
  * *
  ****************************************

ROOT 5.28/00b (tags/v5-28-00b@38397, Mar 14 2011, 15:15:50 on linuxx8664gcc)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0]
```

`-l` オプションをつけると余分な Window が開いたりせず、高速に起動できる。

ここでは、TNtuple と TGraph を利用したプロットを紹介する。TNtuple は ROOT で定義されている C++ クラスの一つであり、データをテーブル形式で取り扱うことができる。指定されたカラムの値、またはカラムの演算値を、指定された条件にマッチするレコードのみでプロットすることが簡単に行える。この機能を使うことで、例えば指定した等級より明るいもののみをプロットしたり、等級別に異なる色でプロットしたりすることができる。また、フィッティングを行ったり、ヒストグラムを作成するのにも利用できる。

一方の TGraph は配列に格納されたデータをプロットするのに利用され、誤差棒付きのプロットを作成したり、関数フィットする場合に利用する。

4.5.1 TNtuple を利用したプロット

まずは、ファイルに書かれたデータを読み、それを TNtuple に格納しプロットする方法について説明する。つぎのようなデータを含むデータファイル `input.dat` があるとする。

```
$ cat input.dat
12.3 24.5 23.0
12.5 22.3 22.7
14.7 23.8 19.3
10.3 20.4 20.0
```

このデータを ROOT でプロットするには、root コマンドを実行し、ROOT のインタラクティブモードで以下のようにコマンドを一行ずつ実行する。

```
root [0] double x,y,z;
root [1] TNtuple *nt = new TNtuple("", "", "x:y:z");
root [2] ifstream data("input.dat");
root [3] while (data>>x>>y>>z) { nt->Fill(x, y, z);}
root [4] nt->Draw("y:x", "z>20");
```

まず一行目でファイルから読み込むデータを一旦格納する変数を定義している。二行目でプロットするデータを格納する TNtuple のインスタンスを作成している。インスタンスとはデータが格納される入れ物に相当する。一方クラスはこの入れ物にどういう値が入れられるかを定義するものである。C++ では、あるクラスのインスタンスを生成する際には new 演算子を用いて、<クラス名> *<インスタンス名> = new <クラス名>(<引数リスト>); のように記述する。または <クラス名> <インスタンス名>(<引数リスト>) のようにしてもよい。通常は前者の方法でインスタンスを生成する。TNtuple クラスのインスタンス生成を行う際の引数リストには第一引数にこのインスタンスの名前、第二引数にタイトル、第三引数にコロン区切りのデータ名リストを指定する。第一、第二引数は空文字でも不便はない。

三行目で入力ファイルをオープンし、四行目でファイルから一行ずつデータを変数 x,y,z に入力し、TNtuple インスタンスに Fill メソッドによってデータを追加している。三行目は ifstrem インスタンスの生成を行い、同時にファイルをオープンする初期化の文であるが、new を使う方法では以下のようになるのであろう。

```
root[0] ifstream *data = new ifstream("input.dat");
```

しかし、この書き方は C++ ではあまり使われないようである。四行目のデータ読み込みは C++ を使ったことがない人には奇妙に思われる記述であるが、スペース区切りで単語を一つずつ読み込むための文法である。一行目から四行目まではこのパターンを丸覚えするか、どこかのファイルまたは Web ページに書いておき、カットアンドペーストで再利用することをお薦めする。その際にはもちろん、入力ファイルのデータ項目数によって定義して使う変数の数や変数名はその都度変えて利用する必要はある。

五行目は横軸に一列目の値、縦軸に二列目の値をとりプロットしている。Draw メソッドの第一引数で x 軸と y 軸にどのデータを利用するかを指定し、第二引数でプロットするデータの条件を指定している。この例の場合、プロットするデータは第三列目の値が 20 より大きい行のものに限定している。また第三引数を追加し、同じグラフ上にオーバープロットするかどうかなどのオプションを指定ができる。この例を実行しただけではプロットした点が小さすぎてほとんど見えないので、SetMarkerStyle メソッドによりプロットするマーカースタイルを●に設変更する。デフォルトでは一ピクセルの点である。利用できるマーカースタイルの例を図 1 に示した。マーカーの色を変えたければ、SetMarkerColor メソッドを使う。利用できる色の例を図 2 に示した。以下のコマンドを試してみるとよい。

```
root [5] nt->SetMarkerStyle(20); # 黒丸
```

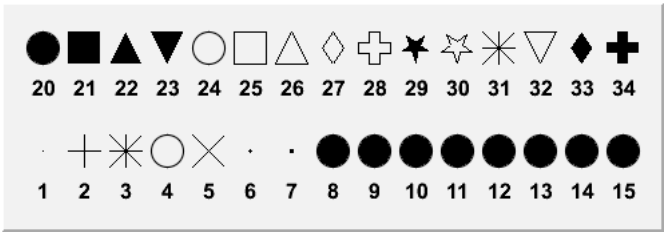

図 1: ROOT で利用できるマーカーの一覧

図 2: ROOT で利用できるカラーの一覧

```
root [6] nt->SetMarkerColor(2); # 赤
root [7] nt->SetMarkerSize(2); # マーカーのサイズを指定
root [8] nt->Draw("y:x", "z>20"); # マーカーでプロット
root [9] nt->Draw("y:x", "z>20", "L"); # 線でプロット
root [10] nt->Draw("y:x", "", "PL"); # マーカーでプロットし線でつなげる
root [11] nt->Draw("y:x", "", "L"); # 線でプロット
root [12] nt->Draw("y:x", "", "P,SAME"); # 同じグラフに上書き
```

Draw メソッドのオプションのリストは以下にある。http://root.cern.ch/download/doc/ROOTUsersGuideHTML/ch03s08.html

4.5.2 TGraph を利用したプロット

次に TGraph クラスを利用したプロットの方法について説明する。TGraph でプロットするデータを一旦配列変数によみこみ、TGraph のインスタンスを生成する際にその配列引数で渡す。以下が実行例である。

```
root [0] double x[4], y[4], z[4];
root [1] ifstream data("input.dat");
root [2] int ind = 0;
root [3] while (data>>x[ind]>>y[ind]>>z[ind]) { ind++; }
root [4] data.close();
root [5] TGraph graph = new TGraph(4, x, y);
root [6] graph->Draw();
```

4.5.3 複数プロットの表示

複数のプロットを一つの画面上に描きたい場合は、TCanvas クラスのインスタンスを生成し、Divide メソッドで画面分割パタンを指定し、cd メソッドでプロットする場所を指定する。

```
1    TCanvas* c1 = new TCanvas("c1", "c1", 300, 600);
2    c1->Divide(1,2);
3    c1->cd(1);
4    <プロット 1>
5    c1->cd(2);
6    <プロット 2>
```

この例では、描画用のウィンドウを”300x600 ピクセル” のサイズで表示し、全体を横一列縦二行に分割する。そして、上段に<プロット 1>を、下段に<プロット 2>を描画する。横二列縦二行に分割したい場合は c1->Divide(2,2) である。

4.5.4 プロットのファイルへの保存

プロットをファイルに保存したい場合は、Print メソッドを利用する。

```
1    c1->Print("plot.eps"); # eps で保存
2    c1->Print("plot.gif"); # gif で保存
3    c1->Print("plot.png", "png"); # png で保存
```

4.5.5 マクロの利用方法

マクロの書き方は、無名マクロとして以下のように波括弧内にコマンドを記述する。

```
1 {
2    コマンドを記述
3 }
```

引数を渡したい場合はファイル名と一致する名前で関数を記述する。たとえば、my_macro.C というファイルに以下のようにマクロを書く。

```
1 void my_macro(double a, double b) {
2    コマンドを記述
3 }
```

マクロの実行方法は、ROOT のコマンドプロンプトで以下のように行う。

```
1 root [0] .x my_macro.C; # 引数がない場合または無名マクロの場合
2 root [0] .x my_macro.C(1.2, 1.0); # 引数が定義されたマクロの場合
```

OS のシェルコマンドから実行するには、つぎのようにする。

```
1 $ root -b -q "my_macro.C"
2 $ root -b -q "my_macro.C(1.2, 1.0)"
```

4.5.6 その他

root のユーザーガイドは以下にある。http://root.cern.ch/drupal/content/users-guide また、検索サイトでキーワード ”root cern” で検索すると様々な使い方マニュアル（メモ）が公開されていることが分かる。

4.6 Unix シェルスクリプト

Unix シェルスクリプト (BASH) の基本文法について、本課題で必要となる文法に限って紹介する。シェルスクリプトについて詳しい人は読み飛ばして構わない。

4.6.1 宣言

BASH スクリプトを作成するには、ファイルの先頭に以下の行を記述する。

```
1 #!/bin/bash
```

4.6.2 変数

変数は宣言することなく利用することができるが、関数内では局所変数として宣言して利用すること推奨する。宣言をしないで定義される変数は大域変数としてあつかわれるので、呼出し元で同じ変数を利用している場合には、関数内で書き換えられてしまう場合がありえる。短い簡単なスクリプトの場合は変数名が同じにならないように注意をすることでそうした問題は防げるが、スクリプトが長くなり複雑化すると予期せぬところで、変数が書き換えられてしまうということがおきやすくなる。

大域変数への値の代入は

```
1  <変数名>=<値>
2 または
3  <変数名>="<値>"
```

のように記述する。=の前後にはスペースを入れないことに注意をする必要がある。ダブルクォーテーションで囲う場合、囲われた中身が代入される。値はリテラル文と変数を混在されることが可能であり、変数の値を参照するには変数名の前に $ をつける。変数名を波括弧 {} で括ってその前に$をつけることもできる。そのように書くことで変数名の区切りがはっきりさせることができる。たとえば、echo ${a}a は echo $aa とした場合、変数 aa を参照するように解釈されてしまう。ここで、echo はそれに引き続く式を変数展開した結果を標準出力に表示するためのコマンドである。

a=test と a="test" は同じである。a=test; b=${a} dayo を実行すると、b には “test dayo” という文字列が代入される。リテラルに空白文字などが含まれる場合にはダブルクォーテーションで囲う必要がある。シングルクォーテーション内で変数を参照してもその変数は値に展開されないことに注意する必要がある。

局所変数の定義は

```
1 local <変数名>
2  または
3 local <変数名>=<値>
```

である。

4.6.3 引数

シェルスクリプト実行時に指定した引数はシェルスクリプト内で参照することができる。第一引数は $1 で参照でき、第二引数は $2 である。第三引数以降も同様に $3，$4，... で参照できる。指定された引数の数は $# で参照できる。

```
a=$1 # 第一引数
b=$2 # 第二引数
n=$# # 引数の数
```

4.6.4 関数

BASH では関数が定義できる。CSH という Unix シェルがあるが、こちらでは関数が利用できないので、BASH の利用を推奨する。あるまとまった処理をひとつの関数とすることで、その処理が再利用しやすくなり、スクリプト自体も読み易くなる。

関数の定義は以下のとおりである。

```
function 関数名 {
   処理
   return 返り値
}
```

関数に渡される引数は $1，$2，$3，... のように参照する。返り値は整数値である必要がある。また、一般的なプログラミング言語でよく行われるように、呼出元で a=func のように戻り値を直接変数に代入することはできない。返り値を参照するには、関数呼び出し直後に $? を参照する必要がある。$? はシェルスクリプトの実行文を一回実行するごとにその実行文の返り値が設定されるので、関数呼び出し直後に参照しないと、べつの実行文の返り値で書き換えられてしまうことに注意する。

4.6.5 条件分岐

bash での条件分岐は if 文, case 文がよく使われる。if 文は

```
if <条件式>; then
   <実行文>
elif <条件式>; then
   <実行文>
else
   <実行文>
fi
```

のように記述する。もちろん elif, else は省略可能である。

条件式はコマンド実行文または角括弧 [] による条件判断文、そしてそれらの &&, || による論理結合が利用される。コマンド実行文を利用する例として、あるファイルが指定された文字列を含むかどうかを判断するには以下のように行う。

```
if cat test.dat | grep "abc" > /dev/null; then
   echo "test.dat contains abc"
else
   echo "test.dat doesn't contains abc"
fi
```

これは grep コマンドが指定した文字が見付からなかった場合に終了ステータス 1 を返すことを利用している。コマンド実行後の終了ステータスが 0 のときは、その条件式は true と判断されて if 文が実行される。終了ステータスが 0 以外の場合は false と判断されて else 文が実行される。

角括弧による条件式は様々ある。頻繁に使われるであろう条件式を列挙する。

```
1  if [ -e <file> ]; then # ファイル <file> が存在するか
2  if [ -s <file> ]; then # ファイル <file> が存在するし、サイズが 0以上か
3  if [ "$a" == "1" ]; then # 変数 a が "1" と一致するか
4  if [ "$a" <> "1" ]; then # 変数 a が "1" と一致しないか
```

その他については、以下の URL を参照すること。

http://www.gnu.org/software/bash/manual/bashref.html#Bash-Conditional-Expressions

4.6.6 ループ

変数値を変えながら、おなじような処理を繰り返し実行することはよくあることである。bash ではそうした繰り返し実行をサポートするための構文として、for 文と while 文が定義されている。for 文のよく使われる利用方法として、あるパターンにマッチするファイル名を変数に順次代入しつつ処理を繰り返す方法があげられる。以下はカレントディレクトリにある hist-*.dat パターンにマッチするファイル名を出力するループである。

```
1  #!/bin/bash
2  for fn in hist-*.dat; do
3     if [ ! -e "$fn" ]; then
4        cotinue
5     fi
6     echo $fn
7  done
8  exit 0
```

カレントディレクトリに hist-1.dat, hist-2.dat というファイルがある場合の実行結果は以下のとおりとなる。

```
1  $ sample.sh
2  hist1.dat
3  hist2.dat
```

if [! -e "$fn"]; then では、マッチするファイルがない場合、変数 fn にはファイル名展開が行われず、hist-*.dat という文字列がそのまま代入されてしまうので、それをスキップするための条件判断を行っている。このループを利用することにより、特定のファイル群に対して同じ処理を繰り返し行うことが可能である。

そのほか、一般的なプログラム言語で行われるような、決められた回数だけ変数を一ずつ変更しながら繰り返し実行するには、以下のように行う。

```
1  #!/bin/bash
2  for i in $(seq 1 100); do
3     echo $i
4  done
```

seq は単調増加 (減少) する数値列を表示するコマンドである。詳細は man seq または seq --help を実行することでみるもとができる。

while 文は指定された条件式が false となるまで do と done で囲まれた部分を繰り返し実行する。よく使うケースとしてはファイルから一行ずつ読み込み、それらに対して繰り返し処理をおこなうということがあげられる。以下のサンプルは、ファイル input.dat から一行ずつデータを読み込みそれを標準出力に表示する例である。

```
#!/bin/bash
while read line; do
   echo $line
done < input.dat
```

read は bash の組込みコマンドであり、ファイルから一行ずつよみ与えられた変数に代入する。

4.6.7 展開

コマンドの実行結果を参照する方法について説明する。これは、次節で説明される Unix コマンドの実行結果を参照することにより、算術演算の結果や、ファイルの行数のカウント、ファイルの先頭行、最終行、指定された行を取得し、変数に代入するといった使いかたがおこなえる。

```
a=10.0
b=0.5
c=$(echo $a $b | awk '{print $1*$2}')
```

この例では、awk コマンドで計算される 10.0*0.5 の演算結果を変数 c に代入している。シェルスクリプトでは、実数演算を行うことができないので、awk コマンドを利用し演算を行う。bc という演算用コマンドもあるが、利用方法は注意を要する。例えば、

```
a=10.0
b=-0.5
c=$(echo "${a}-${b}" | bc)
```

のような演算をするとエラーになってしまう。このエラーを避けるには ${b} を括弧で括る必要がある。できるだけ awk を常に使う方が、記述は長めであるが、安全である。

変数の値を置換して参照することができる。この機能はファイル名かか拡張子をのぞいた名前を使いたいとか、拡張子だけを参照したい場合、ディレクトリのパス名を取り除いてファイル名だけを参照したいという場合に利用される。

```
file=/home/watashi/tmp.dat
echo ${file%.dat}
echo ${file##*.}
echo ${file##*/}
```

これを実行すると、

```
/home/watashi/tmp
dat
tmp.dat
```

という結果が得られる。${変数%パターン} は変数値が指定されたパターンにマッチする文字列で終了する場合に、パターンと一致する最短の文字列をのぞいた値を返す。${変数%%パターン} では、パターンと一致する最長の文字列をのぞいた値を返す。${変数#パターン} は変数値が指定されたパターンにマッチする文字列で始まる場合に、パターンと一致する最短の文字列をのぞいた値を返す。${変数##パターン} では、パターンと一致する最長の文字列をのぞいた値を返す。

4.6.8 Unix コマンド

awk 数値演算を行ったり、書式に従った出力をしたい場合、文字列置換をしたい場合など様々な使われかたがされる。

```
$ x=10
$ y=2
$ z=$(echo $x $y | awk '{pritn $1/$2}')
$ echo $z
5
$ echo $z | awk '{printf("%8.2f\n", $1)}'
    5.00
$ echo $z | awk '{printf("%-8.2f\n", $1)}'
5.00
$ echo abc_defg | awk '{gsub("_", ",", $1); print $1}'
abc,defg
```

BASH では実数の演算が記述できないので、代わりに awk を利用するとよい。

本課題における利用例としては、例えば置換機能を利用して JVOQL の領域指定関数中の座標値を変数とするテンプレートを作成し、ファイルに書かれた座標値を読み込みながらその値で置換し、検索を行うということ可能である。

```
#!/bin/bash
#
# JVOQL Template
#
cat > templ.sql <<EOF
SELECT *
FROM ivo://jvo/ukidss/dr2:catalog_dr2 AS t
WHERE REGION('CIRCLE @ra@ @dec@ @rad@')
EOF
#
#
#
while read line; do
   ra=$(echo $line | awk '{print $1}')
   dec=$(echo $line | awk '{print $2}')
   rad=0.2
   cat templ.sql | \
       awk '{gsub("@ra@", ra); \
             gsub("@dec@", dec); \
             gsub("@rad@", rad); print $0}' \
             ra=$ra dec=$dec rad=$rad > ukidss.sql

   echo $line
   cat ukidss.sql
done < region.dat
```

検索する座標値の赤経・赤緯をスペース区切りで一行ずつ region.dat に列挙し、上記のスクリプトを実行すると、一行読む毎にファイル templ.sql に保存されているテンプレート JVOQL の @ra@, @dec@, @rad@ を数値で置き換えた JVOQL が ukidss.sql に保存される。

sort 入力ファイルの内容を行単位でソートし出力する。

```
$ cat <file> | sort # abc 順にソートする。
$ cat <file> | sort -n # 第一列目の数値でソートする。
```

```
$ cat <file> | sort -n -k +2 # 第二列目の数値でソートする。
$ cat <file> | sort -n -t, -k +2 # カンマ区切りで第二列目の数値でソートする。
$ cat <file> | sort -u # abc 順にソートし、重複をのぞく
```

本課題における利用例としては、AGN 周辺の天体分布からデータの欠損率分布を求め、その欠損率の最大値がある値を越える場合はそのサンプルは使わないということを行う場合に利用できる。そのためには、欠損率のリストを一行毎に出力し、その各行を欠損率で逆順ソートし、先頭一行目の値を取得することで、欠損率の最大を求めることができる。

```
#
# earea.dat は一列目に AGN からの射影距離、二列目に欠損率が
# 記述されているとする。射影距離が 3 以下における欠損率の最大値
# は以下のように求められる。
#
$ max=$(cat earea.dat | grep -v "#" | \
        awk '{if($1<3.0) print $2}' | \
        sort -nr | head -1)
```

上記の例により、変数 max に欠損率の最大値がセットされる。

cut　文字列から部分文字列を抜き出す場合に使う。

```
$ id=$(echo SUPA00211350 | cut -b 5-12)
$ echo $id
00211350
```

head, tail, cat, sed　ファイルの先頭行を参照 (head), ファイルの最終行を参照 (tail), ファイルの内容全部を参照 (cat), ファイルの特定の行を参照 (sed), するのに利用される。

```
$ head -1 <file> # ファイル <file> の先頭一行目を出力
$ cat <file> | head -1 # 標準入力の先頭一行目を出力
$ tail -1 <file> # ファイル <file> の最終行を出力
$ tail -f <file> # ファイル <file> が更新される毎に更新分を出力
$ cat <file> # ファイル <file> の内容を標準出力に出力
$ sed -n 10,12p <file> # ファイル <file> の 10行目から 12行目を出力
```

wc　ファイルの行数をカウントするのに使う。

```
$ n=$(cat <file> | wc -l) $ ファイル <file> の行数を変数 n に代入する。
```

5　研究の手順

実際にデータを VO を利用して取得し、AGN と銀河の相関距離を求める手順について解説する。

まずは、データを集める方法について説明し、集めたデータから AGN 周辺の銀河数密度分布を求め、それを可視化する方法について説明する。全 AGN の銀河数密度分布を足しあわせることにより、AGN と銀河のクラスタリングの S/N を高め、相関関数にフィットすることにより相関距離を求める方法について解説する。

5.1 データを集める

まずは銀河データが存在する領域の AGN を検索し、それら AGN の情報 (座標、明るさ、赤方偏移等) を集める。集める方法は、(1) JVO portal の Single Service Search の機能と VOTable Viewer の ADD COLUMN 機能を利用する、(2) JVO portal の JVOQL Search の機能を利用する、(3) jc コマンドによる JVOQL Search の機能を利用する等いくつかあるが、ここでは最も効率的に検索が行える JVOQL を用いた検索、(2) と (3) の方法について紹介する。

5.1.1 JVOQL による検索 (portal)

JVO portal を利用した JVOQL によるデータ検索は、図 3 にあるように、トップページにある “JVOQL Search” と書かれたリンクをクリックして表示される JVOQL 検索画面より行える。リスト 1 の例にある JVOQL を入力し、”Submit” ボタンをクリックすることにより、Veron カタログの AGN とその周辺の UKIDSS カタログ天体の検索が行える。

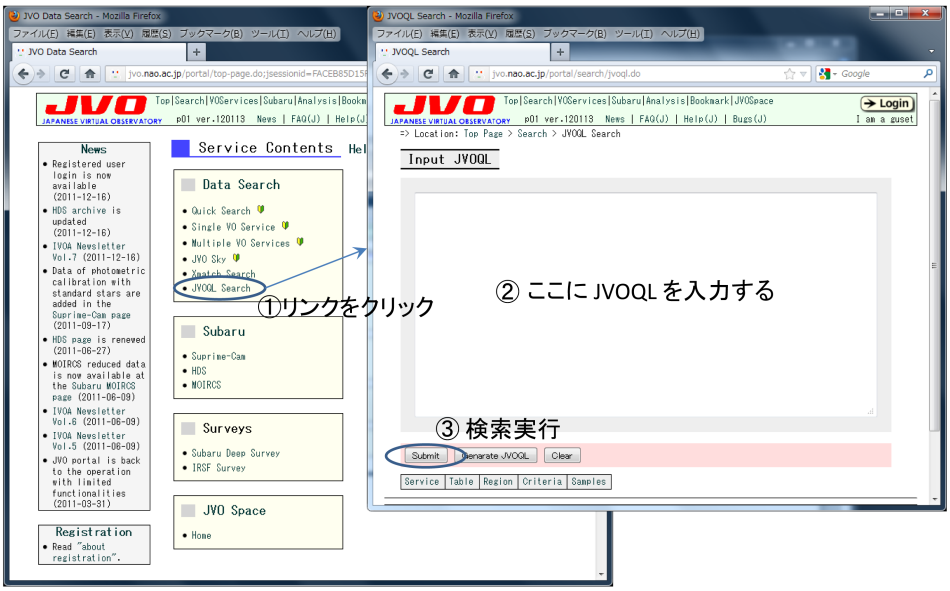

図 3: JVO portal 上での JVOQL 検索

```
1    SELECT agn.name, agn.z, agn.ra, agn.dec,
2           ir.ra2000, ir.de2000, ir.mag_k
3    FROM ivo://jvo/agn:veron_2010 agn,
4           ivo://jvo/ukidss:catalog_dr8 AS ir
5    WHERE agn.z >= 0.3 and agn.z < 0.31
6           AND
7           distance((ir.ra2000, ir.de2000), (agn.ra, agn.dec)) < 0.01
```

リスト 1: UKIDSS データが 0.01 度以内にある AGN を検索し、その AGN の名前、赤方偏移、座標と UKIDSS 天体の座標と K バンド等級を取得する JVOQL。AGN の赤方偏移は 0.3 以上 0.31 未満に制限している。

この例では、赤方偏移 0.3 から 0.31 である Veron カタログの AGN のうち、0.01 度以内に UKIDSS カタログの天体があるものについて検索が行われる。1228 件のデータがヒットする。検索結果として取得するデータ項目として、SELECT 節において、Veron カタログ中の AGN 名称 (agn.name)、赤方偏移 (agn.z)、座標 (agn.ra, agn.dec)、UKIDSS カタログ中の対応する天体の座標 (ir.ra2000, ir.de2000) と K バンドの明るさ (ir.mag_k) を指定している。distance 関数は第一引数と第二引数であたえられる座標間の距離を度を単位として計算する関数である。

データサービスが返すことができるデータ件数には多くの場合上限値が設けられているため、その上限を越えないように検索条件を指定する必要がある。この例では、赤方偏移の範囲を制限することで検索にヒットする件数を減らしている。上限を越えてしまったかどうかは、検索結果の INFO パラメータの値を見ることで知ることができる。INFO パラメータの値は VOTable Viewer で見ることができる (図 4)。

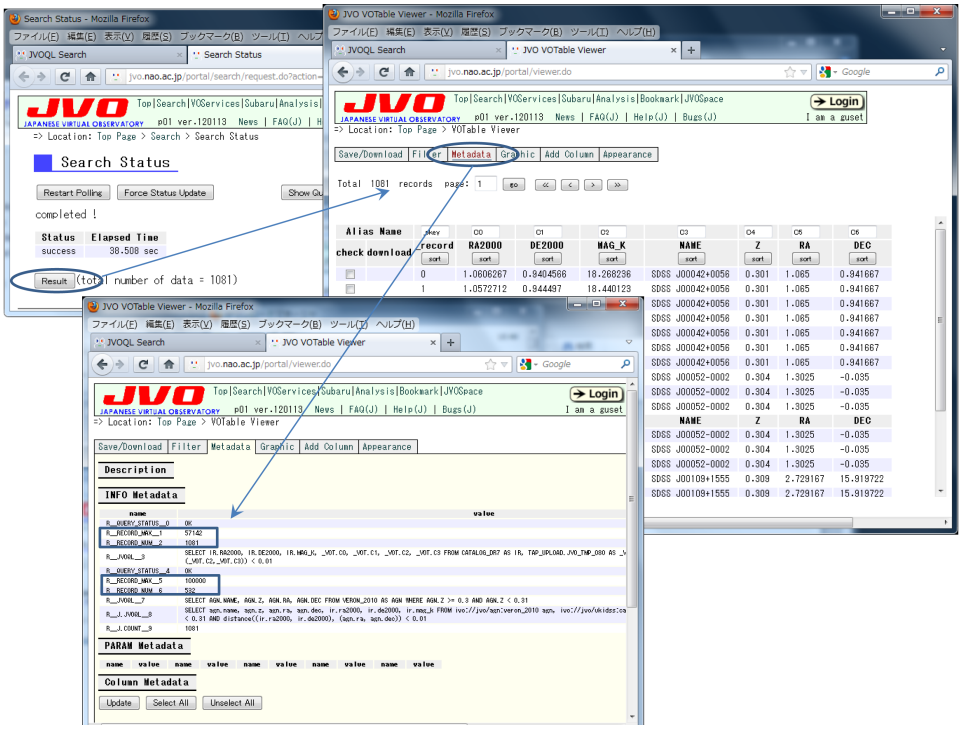

図 4: 検索結果の表示例

検索終了後、検索状況表示画面に表れる Result ボタンをクリックして、VOTable Viewer により結果を表示させる。INFO パラメータ等の値を確認するには、“Metadata” タブを開き、“INFO Metadata” と書かれた部分を見るとよい。JVO でホストしている VO サービスの場合は、RECORD_MAX でサービスが返すレコード数の上限値を、RECORD_NUM で実際にサービスが返したレコード数を示している。RECORD_NUM が RECORD_MAX と同じになっている場合は、検索結果のレコード数が上限に達してしまったことを意味するので、条件をきつくして検索を再実行する必要がある。

検索結果を自分の計算機に保存するには、VOTable Viewer の “Save/Download” タブを開いて csv フォーマットを選択し、“Download” ボタンをクリックする。

UKIDSS 天体のデータは取得せず、まずは UKIDSS 天体のデータがある AGN のデータをのみを取得することにより、一度に取得できる AGN の件数を増やすことが可能である。そのために

は、下記のように SELECT 節に DISTINCT というキーワードを追加し、ukidss カタログのカラムを削除した JVOQL を利用する。

```
SELECT DISTINCT agn.name, agn.z, agn.ra, agn.dec
FROM ivo://jvo/agn:veron_2010 agn,
       ivo://jvo/ukidss:catalog_dr8 AS ir
WHERE agn.z >= 0.3 and agn.z < 0.4
       AND
       distance((ir.ra2000, ir.de2000), (agn.ra, agn.dec)) < 0.01
```

リスト 2: UKIDSS データが 0.01 度以内にある AGN を検索する JVOQL。AGN の赤方偏移は 0.3 以上 0.4 未満。

1580 件のデータがヒットする。

以上で、UKIDSS データがある領域の AGN の検索が行えた。

5.1.2 JVOQL による検索 (jc コマンド)

ここでは、JVO portal を使ったインタラクティブな方法とは別にコマンドラインから検索を実行する方法について紹介する。ポータルを使う方法は簡便ではあるが、時間がかかる検索を何度も繰り返し行うのは非効率である。コマンドラインから行うことができると、スクリプトを書くことにより、多数の検索を自動化することが可能になる。まず、環境設定のためにドットコマンドにより jvorc.sh に記述されていいる設定コマンドを実行する。jvorc.sh は vo2015a パッケージを展開して作成されたディレクトリ vo2015a の下にある。

```
$ . vo2015a/jvorc.sh
```

リスト 2 の JVOQL をファイル (agn.sql) に保存し、jc コマンドの search オペレーションでそのファイルを指定することで、コマンドラインから検索が可能である。

```
$ jc search -i agn.sql -o agn.xml
```

上記の例では、検索結果は agn.xml に VOTable フォーマットで保存される。この VOTable のデータを処理しやすいフォーマットに変換するには、jc コマンドの select オペレーションを利用する。

```
$ jc vot2xsv -F, -o agn.csv agn.xml
```

オプション -F でカラムのセパレータを指定、-o で出力先ファイル名を指定している。両オプションを指定しない場合はタブ区切りで標準出力に表示される。上記のコマンドを実行した結果作成される agn.csv の中身は以下のようである。

```
# 0 NAME char(*) // Most common name of the object
# 1 Z double // redshift
# 2 RA double deg // Right Ascension J2000
# 3 DEC double deg // Declination J2000
SDSS J10207+0427,0.382,155.180833,4.466111
SDSS J01564+0007,0.361,29.12125,0.123611
2QZ J122655-0154,0.338,186.73125,-1.914722
SDSS J23594-0047,0.396,359.859583,-0.797222
SDSS J15223-0144,0.337,230.589583,-1.743056
2QZ J122724+0139,0.333,186.8525,1.658611
SDSS J12467+0747,0.389,191.695417,7.797778
```

```
2QZ J095748-0206,0.357,149.45,-2.102778
...
```

以上で UKIDSS サーベイ領域内の AGN の座標と赤方偏移のデータを jc コマンドにより取得することができたので、次に個々の AGN について UKIDSS カタログからその周辺天体のデータを取得する

使用する JVOQL は以下のようになる。

```
SELECT ID, SOURCE_ID, RA2000, DE2000, MAG_K
FROM ivo://jvo/ukidss:catalog_dr8 AS t
WHERE REGION('CIRCLE 42.861667 1.252222 0.2')
       AND (mergedClass = 1 OR mergedClass = -3)
       AND MAG_K > 0 AND kppErrBits <= 255
       AND SURVEY = 'las'
```

REGION 関数の引数には AGN を中心とした半径 0.2 度の円を表す文字列を記述する。0.2 度は赤方偏移 0.3 の距離では 4 Mpc (共動座標系) に相当する。`mergedClass` は天体の種別を表すフラグであり、1 は銀河と判定されたデータ、-3 は銀河かも知れないと判定されたデータである。このフラグが 1 か −3 のデータのみを取得する。`kppErrBits` が 255 より大きな場合はノイズデータの可能性が高いため、それ以下の値のデータのみ取得する。結果を保存し、それを処理しやすいように CSV 形式に変換する。

```
  $ jc search -i search-ukidss.sql -o ukidss.xml
  $ jc select -F, -o ukidss.csv ukidss.xml
```

課題 1. jc コマンドを用いて、赤方偏移が 0.3 から 0.4 までの AGN のうち、UKIDSS データが存在する AGN について、その座標と赤方偏移のデータを取得する。また、数個の AGN について、その周辺 0.2 度以内の UKIDSS データを取得する。

5.2 AGN 毎に周辺天体の数密度分布を求める

取得した UKIDSS データに欠損領域がないかを確認する。欠損領域は、観測領域の境界や明るい天体によりマスクされた領域に相当する。確認するには、検索範囲内にデータが一様に分布しているかを調べる必要がある。

ここでは、CERN で開発されているデータ解析フレームワークである ROOT を利用してプロットを行う方法について紹介する。ROOT の主要な機能は、グラフや関数の作図、ヒストグラムの作成と操作、データーの任意関数によるフィットといったものがあげられる。C++ の基本的な文法を覚えれば、マクロを作成してコマンドラインから簡単にプロットできるので、同じようなグラフを何度も描く場合に便利に使える。C++ を知らなくても、良く使われる記述パターンを覚えることで簡単なプロットであれば容易に描くことができる。以下の説明では root のインストール先として /usr/local を指定した場合について説明する。それ以外の場所を指定した場合には実行ファイルやライブラリは異なるパスにインストールされていることに注意すること。

以下の例にあるように、CSV 形式で保存した検索結果ファイルから座標データのみを抜き出してファイルに保存し、root を起動してコマンドを一行ずつ入力し実行する。

```
$ cat ukidss.csv | grep -v "#" | awk -F, '{print $3,$4}' > radec.dat
$ root -l
root [0] double x,y;
root [1] TNtuple *nt = new TNtuple("", "", "x:y");
root [2] ifstream data("radec.dat");
root [3] while (data>>x>>y) { nt->Fill(x, y);}
root [4] nt->Draw("y:x");
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [5]
```

root コマンド実行時に "-bash: root: コマンドが見つかりません" と表示されて root が起動しない場合は、環境変数 PATH に root の実行コマンドが含まれているディレクトリが指定されているかを確認すること。echo $PATH で設定を確認できる。講習会用に配布される VMWare 上では /usr/local/bin に root コマンドがインストールされている。PATH の設定は

```
export PATH=/usr/local/bin:${PATH}
```

のように行う。"/usr/local/bin/root.exe: error while loading shared libraries: libCore.so: cannot open shared object file: No such file or directory" というようなメッセージが表示される場合には root で利用するライブラリが認識されていないので認識されるよう設定する必要がある。root のライブラリは /usr/local/lib/root にインストールされている。root アカウントで /etc/ld.so.conf に /usr/local/lib/root という行を追加し、コマンド /sbin/ldconfig を実行する。または、環境変数 LD_LIBRARY_PATH に /usr/local/lib/root を追加する。LD_LIBRARY_PATH への設定は

```
export LD_LIBRARY_PATH=/usr/local/lib/root:${LD_LIBRARY_PATH}
```

のように行う。

中心座標 (42.861667,1.252222) 半径 0.2 度の検索結果の場合、図 5 のようなプロットが作成される。

見栄えを良くするためには、次の例のように設定コマンドをいくつか実行する必要がある。

```
/**
 *
 */
void plot_map(double ra, double dec, double rad)
{
    //
    // 描画範囲を計算
    //
    double xmin, xmin, ymin, ymax;
    double dy, dx;
    dx = rad / cos(dec/180.*3.141592654);
    dy = rad;
    xmin = ra - dx;
    xmax = ra + dx;
    ymin = dec - dy;
    ymax = dec + dy;

    //
    // データ入力
    //
```

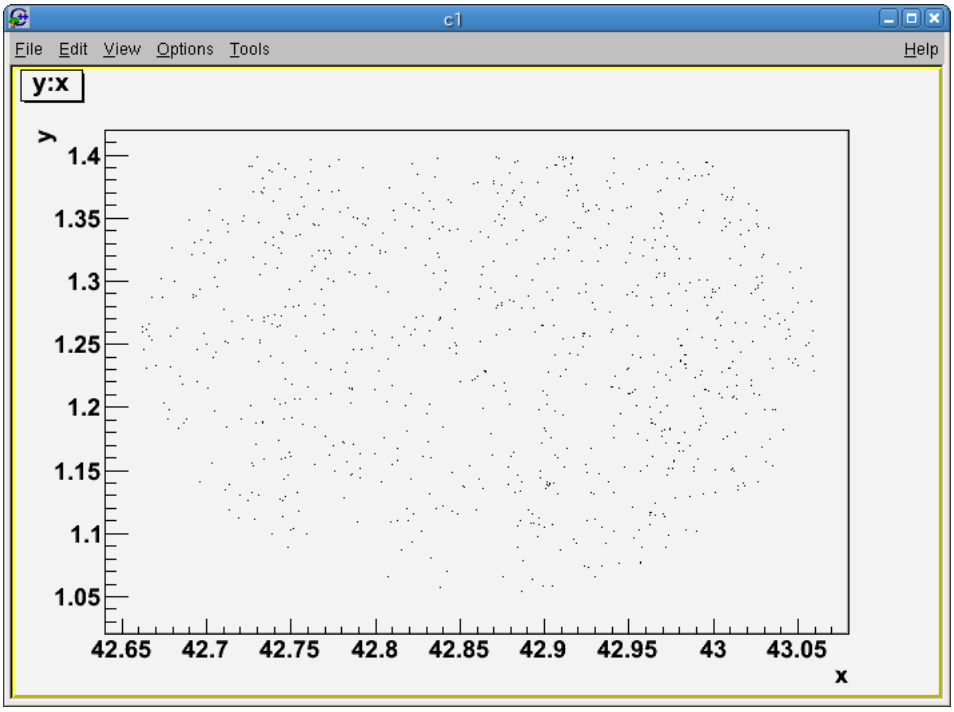

図 5: 銀河の分布。

```
21    double x, y;
22    TNtuple* nt = new TNtuple("object", "object", "x:y");
23    string buf;
24    ifstream data("radec.dat");
25    while (data && getline(data, buf)) {
26        if(buf.at(0) == '#') {
27            continue;
28        }
29        istrstream istr(buf.data());
30        istr>>x>>y;
31        nt->Fill(x, y);
32    }
33    data.close();
34
35    //
36    // グラフ描画の準備
37    //
38    gROOT->SetStyle("Plain"); // タイル表示を off
39    gStyle->SetOptTitle(kFALSE); // タイトル消す
40    gStyle->SetOptStat(kFALSE); // 統計情報消す
41    TCanvas* c1 = new TCanvas("c1", "c1", 400, 400);
42    gStyle->SetNdivisions(505, "X"); // X 軸の目盛設定
43    gStyle->SetNdivisions(505, "Y"); // Y 軸の目盛設定
44
45    //
46    // XY 表示範囲を指定するための空ヒストグラム
47    //
48    TH1F *waku = new TH1F("waku", "title", 2, xmin, xmax);
49    waku->SetMinimum(ymin);
```

```
50      waku->SetMaximum(ymax);
51      waku->Draw();
52      waku->GetXaxis()->SetTitle("RA"); // X 軸のタイトル
53      waku->GetYaxis()->SetTitle("Dec"); // Y 軸のタイトル
54
55      //
56      // 範囲(円)
57      //
58      TEllipse *region = new TEllipse(ra, dec, dx, dy);
59      region->Draw("same");
60
61      //
62      // 天体プロット
63      //
64      nt->SetMarkerStyle(20); // マーカースタイル
65      nt->SetMarkerColor(2); // マーカーの色
66      nt->SetMarkerSize(0.4); // マーカーサイズ
67      nt->Draw("y:x", "", "same");
68
69      //
70      // プロットを eps ファイルに書き出す
71      //
72      c1->Print("map.eps");
73  }
```

この例にあるマクロはダウンロードした vo2015a パッケージの clustering/samples/plot_map.C にあるので、これをコピーして、root のコマンドラインから

```
1  root [0] .x plot_map.C(42.861667, 1.252222, 0.2);
```

を実行すると、図 6 のようなプロットが作成される。引数に指定した値は、 AGN の座標 (赤経、赤緯) と UKIDSS データの検索半径である。UKIDSS データが赤い点で、検索領域が円で示されている。この例の場合、検索領域上部に観測領域の境界があることが確認できる。

root のコマンドラインから抜けるには、.q を実行する。バッチモードで実行するには、以下のコマンドを実行する。

```
1    $ root -b -l -q "plot_map.C(42.861667,1.252222,0.20)"
```

プロット画像は map.eps というファイルに eps フォーマットで保存されており、それを表示するには display コマンドを使う。display コマンドがない場合には ImageMagick をインストールすること。ImageMagick は http://www.imagemagick.org/ で配布されている。

png 形式の画像に変換するには convert コマンドを利用する。png 形式の画像を表示する場合も display コマンドを利用できる。

```
1    $ display map.eps
2    $ convert map.eps map.png
3    $ display map.png
```

このようにプロットして目で見て欠損領域を確認することも可能であるが、なんらかの基準値に基づき数値的に判断することで、大量のデータを扱うことができるようになる。ここでは、vo2015a パッケージに含まれている jp.ac.nao.jvo.research.clustering.CatalogEffectiveArea という Java プログラムを利用する。このプログラムは AGN を中心とした一定幅の円環領域毎に欠損領域の割合を計算する。UKIDSS 天体の座標 (ra dec) をスペース区切りで一行毎に書き出した

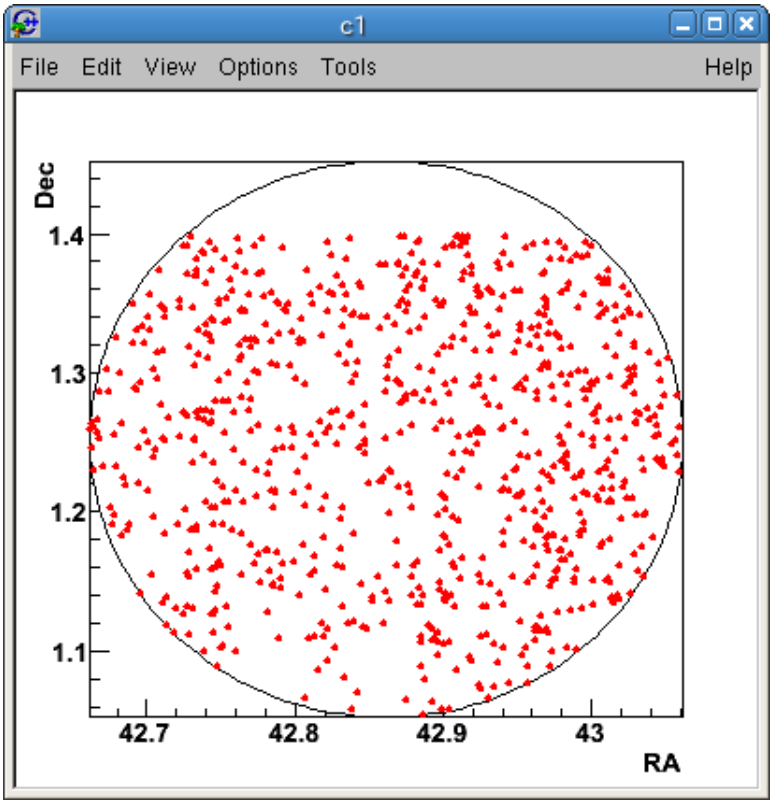

図 6: 銀河の分布。円はデータ検索範囲。

ファイルを用意すし、[4]。以下のように引数として、AGN の座標 <赤経>、<赤緯>、UKIDSS 天体の座標ファイル名、AGN 近傍のマスク領域 (arcsec)、円環幅 (arcsec)、ノイズレベル (sigma) の順で指定する。

```
$ java jp.ac.nao.jvo.research.clustering.CatalogEffectiveArea 42.861667 1.252222
     radec.dat 4.0 17.3 10
```

このコマンドの実行により、point.dat, mask.dat, dead.dat, sim-mask.dat, sim-dead.dat, earea.dat というファイルができる。ここでは earea.dat を以下の計算で利用する。円環の幅は、AGN の距離で 0.1 Mpc の共動距離に対応する角距離とし、$z = 0.3$ の場合 17.3 arcsec となる。角距離と共動距離の関係は、プログラム `cosmic-distance` を利用することで計算できる。

```
$ cosmic-distance -z 0.3 --length 0.1
comoving distance: 1192.175023 [Mpc]
luminocity distance : 1549.827530 [Mpc]
lookback time : 3.399447e+09 [yr]
distance modulus : 40.951417
proper separation distance : 0.000000 [Mpc] (theta=0.000000 [arcsec])
comoving separation distance : 0.000000 [Mpc] (theta=0.000000 [arcsec])
separation angle : 17.301554 [arcsec] (length=0.100000 [Mpc])
comoving volume: 5224894742.355787 [Mpc3] (z=0.300000, omega=1.000000e+00, dz
    =1.000000)
```

[4] この入力ファイルにはコメント行などは書かないようにすること

引数 -z に引き続いて赤方偏移を指定し、--length に引き続きて共動距離を指定して実行すると、上記の結果が得られる。separation angle と書かれた行が、引数で指定された共動距離に対応する角距離であり、0.1 Mpc に対応する角距離は 17.3 秒角であることが分かる。

欠損割合の計算は、領域を細かなメッシュに分割し、メッシュの中の天体数がゼロの領域を欠損領域と判定することによって行っている。CatalogEffectiveArea の出力ファイルである earea.dat は以下のようになっており、2 列目が AGN からの角距離 (arcsec)、3 列目が対応する円環領域の有効面積 (arcmin^2)、3 列目が欠損と判断された領域の割合である。

```
# bin rmid earea badFrac count density maskFrac
# --- arcsec arcmin2 --- --- arcmn-2 ---
   0 8.650 2.47217e-01 0.0000 1 4.04503e+00 0.0000
   1 25.950 7.83539e-01 0.0000 0 0.00000e+00 0.0000
   2 43.250 1.30590e+00 0.0000 2 1.53151e+00 0.0000
   3 60.550 1.82826e+00 0.0000 4 2.18787e+00 0.0000
   4 77.850 2.35062e+00 0.0000 3 1.27626e+00 0.0000
   5 95.150 2.87298e+00 0.0000 3 1.04421e+00 0.0000
   6 112.450 3.39534e+00 0.0000 13 3.82878e+00 0.0000
   7 129.750 3.91770e+00 0.0000 9 2.29727e+00 0.0000
   8 147.050 4.44006e+00 0.0000 11 2.47745e+00 0.0000
   9 164.350 4.96242e+00 0.0000 10 2.01515e+00 0.0000
  10 181.650 5.48478e+00 0.0000 5 9.11614e-01 0.0000
  11 198.950 6.00714e+00 0.0000 12 1.99762e+00 0.0000
  12 216.250 6.52949e+00 0.0000 11 1.68466e+00 0.0000
  13 233.550 7.05185e+00 0.0000 16 2.26891e+00 0.0000
  14 250.850 7.57421e+00 0.0000 11 1.45230e+00 0.0000
  15 268.150 8.09657e+00 0.0000 14 1.72913e+00 0.0000
  16 285.450 8.61893e+00 0.0000 15 1.74035e+00 0.0000
  17 302.750 9.14129e+00 0.0000 14 1.53151e+00 0.0000
  18 320.050 9.66365e+00 0.0000 25 2.58701e+00 0.0000
  19 337.350 1.01860e+01 0.0000 19 1.86530e+00 0.0000
  20 354.650 1.07084e+01 0.0000 25 2.33462e+00 0.0000
  21 371.950 1.12307e+01 0.0000 19 1.69179e+00 0.0000
  22 389.250 1.17531e+01 0.0000 30 2.55252e+00 0.0000
  23 406.550 1.22755e+01 0.0000 31 2.52537e+00 0.0000
  24 423.850 1.27978e+01 0.0000 49 3.82878e+00 0.0000
  25 441.150 1.33202e+01 0.0000 46 3.45341e+00 0.0000
  26 458.450 1.38425e+01 0.0000 32 2.31172e+00 0.0000
  27 475.750 1.43649e+01 0.0000 34 2.36688e+00 0.0000
  28 493.050 1.48872e+01 0.0000 32 2.14949e+00 0.0000
  29 510.350 1.54096e+01 0.0000 39 2.53089e+00 0.0000
  30 527.650 1.59160e+01 0.0010 50 3.14149e+00 0.0000
  31 544.950 1.55329e+01 0.0560 34 2.18890e+00 0.0000
  32 562.250 1.53299e+01 0.0970 26 1.69603e+00 0.0000
  33 579.550 1.48042e+01 0.1540 19 1.28342e+00 0.0000
  34 596.850 1.53723e+01 0.1470 33 2.14672e+00 0.0000
  35 614.150 1.52244e+01 0.1790 39 2.56167e+00 0.0000
  36 631.450 1.46809e+01 0.2300 26 1.77101e+00 0.0000
  37 648.750 1.42996e+01 0.2700 32 2.23783e+00 0.0000
  38 666.050 1.46407e+01 0.2720 39 2.66381e+00 0.0000
  39 683.350 1.42988e+01 0.3070 24 1.67846e+00 0.0000
  40 700.650 1.36665e+01 0.3540 33 2.41466e+00 0.0000
  41 717.950 1.34837e+01 0.3780 26 1.92826e+00 0.0000
```

欠損の判断は数密度で判断しているので、天体数が少ない場合は誤差が大きくなる。また、ボイドのような領域がある場合、そこは欠損領域と判断される場合がある。ボイドであるのか、明るい天体によりマスクされた領域なのかはカタログだけからは判断できず、画像データに戻って確認する必要が

あるが、ここでは、カタログだけから判断する。誤差が大きくても、多数のサンプルで平均化するので、結果に大きな影響を与えることはない。この結果を調べて、相関関数を求める範囲内で欠損領域の割合が一定値以上を越える場合はそのデータは利用しないようにする。CatalogEffectiveArea の結果をグラフ化する root マクロが clustering/samples/plot-earea-sample.C にあり、これをコピーして、

```
root [1] .x plot-earea-sample.C(0.0)
```

と実行すると、欠損領域と判断された領域が青いボックスで示す図が表示される。

課題 2. 課題 1 で取得した UKIDSS データをプロットし、欠損領域があるかを確認すること。また、CatalogEffectiveArea を用いて欠損率を求めてみること。

次に、天体数密度分布を AGN からの射影距離の関数として求める。これは、上記の欠損領域を調べたときに定義したのと同じ円環領域を定義し、それぞれの円環領域における天体数密度を求める。数密度を求める際には、CatalogEffectiveArea で求めた円環領域の有効面積で天体数を割り算して求める。同時に統計誤差についても求める。agn-catalog-density で以上の計算が行える。使いかたは次の例にあるように、-r, -d オプションで AGN の赤経, 赤緯を指定し、--dth で欠損率を求めるのに使った円環の幅と同じ値を指定、--earea で agn-catalog-coverage の出力結果を保存したファイル、-c で UKIDSS 天体の座標ファイル、--mask-rad で AGN 近傍のマスク領域サイズ (agn-catalog-coverage で指定した値と同じ値)、-z で AGN の赤方偏移をそれぞれ指定する。

```
$ agn-catalog-density -r 42.861667 -d 1.252222 --dth 17.30 \
  --earea earea.dat -c radec.dat --mask-rad 4.0 -z 0.3 --mask mask.dat > hist.dat
$ less hist.dat
# agn_ra = 42.861667 [deg]
# agn_dec = 1.252222 [deg]
# agn_z = 0.300000
# agn_dist(Dl) = 1549.827530 [Mpc] (luminosity distance)
# agn_dist(Dc) = 1192.175023 [Mpc] (comoving distance)
# step_dr = 0.099991 [Mpc]
# step_dth = 17.300000 [arcsec]
# log_scale = 0
# step_log = 0.200000 [log10(arcsec)]
# mask_rad = 4.000000 [arcsec]
# min_rad_Mpc = 0.023119
#
# i, r(pc), hist[i], hist_cor[i], area[i](Mpc2), den(Mpc-2), err(Mpc-2), bad_frac
#
0 0.061555 0 0.000000 0.029731 0.000000 0.000000 0.000000 10.650000
1 0.149987 0 0.000000 0.094231 0.000000 0.000000 0.000000 25.950000
2 0.249978 2 2.000000 0.157051 12.734683 9.004781 0.000000 43.250000
3 0.349969 4 4.000000 0.219872 18.192404 9.096202 0.000000 60.550000
4 0.449960 3 3.000000 0.282693 10.612236 6.126977 0.000000 77.850000
5 0.549951 3 3.000000 0.345513 8.682738 5.012981 0.000000 95.150000
```

```
24    ...
```

この数密度分布を誤差棒付きでプロットするには以下のようなマクロを実行すればよい。

```
1  void plot_density()
2  {
3      //
4      //
5      //
6      int ind=0;
7      int num=0, count=0;
8      string buf;
9      double x[1000], y[1000], sig[1000], erx[1000];
10     double dummy;
11
12     ifstream data("hist.dat");
13
14     while (data && getline(data, buf)) {
15         if(buf.at(0) == '#') {
16             continue;
17         }
18         istrstream istr(buf.data());
19         istr>>x[ind]>>count>>dummy>>dummy>>y[ind]>>sig[ind];
20         erx[ind] = 0;
21         ind++;
22         if(count > 0) {
23             num = ind;
24         }
25         if(ind >= 1000) {
26             break;
27         }
28     }
29     data.close();
30     //
31     //
32     //
33     TGraphErrors graph = new TGraphErrors(num, x, y, erx, sig);
34
35     graph->SetLineColor(1);
36     graph->SetLineWidth(1);
37     graph->SetMarkerColor(1);
38     graph->SetMarkerSize(1.);
39     graph->SetMarkerStyle(20);
40
41     graph->SetTitle("");
42     graph->GetXaxis()->SetTitle("distance (Mpc)");
43     graph->GetYaxis()->SetTitle("density (Mpc-3)");
44
45     graph->Draw("AP");
46     graph->Print("density.eps");
47
48 }
```

このマクロは clustering/samples/plot_density.C にある。実行結果は図 8 である。

課題 3. 次の作業を行うスクリプトを作成する。

1. 引数により AGN の座標と赤方偏移、作業ディレクトリ名をスクリプトに変数として与えられること。
2. スクリプト実行開始時に引数で与えられた作業ディレクトリを作成し、そこに移動しそこで検索などの作業が行われること。
3. AGN から半径 0.14 度の範囲の UKIDSS データを取得すること。取得した UKIDSS データのうち、 赤経、赤緯、等級 について抜き出し radecmag.dat というファイルにスペース区切りで天体毎に一行ずつ書き出すこと。
4. 取得した UKIDSS データから欠損率を求め、earea.dat というファイル名で保存すること。
5. UKIDSS データから数密度分布を AGN からの射影距離の関数として求め、hist.dat というファイル名で保存すること。
6. UKIDSS データの天球分布と数密度分布をグラフ化し、png 形式で保存すること。
7. AGN の座標、赤方偏移、観測バンド (今回は K) を summary.dat に出力する。フォーマットは 一行ごとに`<KEYWORD>: <VALUE>` 形式で記述すること。それぞれ、次のキーワードを使用すること：`RA, DEC, REDSHIFT, BAND`。

   課題 1 で検索された AGN のうち、100 個程度について、このスクリプトを実行し、数密度分布を求めるスクリプトを作成し実行する。

課題 3 のスクリプト作成が困難である場合は clusring/samples にある `agn-dataset-sample.sh`, `plot-dist-sample.C`, `plot-earea-sample.C`, `agn-auto-sample.sh`, を利用してもよい。agn-dataset-sample.sh の利用方法は

```
1   $ ./agn-dataset-sample.sh <ra> <dec> <redshift> <workdir>
```

である。agn-auto-sample.sh は引数なしで実行する。ただし、agn の検索で作成されたファイル agn.csv が実行ディレクトリ上に必要である。

5.3 全 AGN の周辺天体数密度分布を足しあわせる

以上のようにして個々の AGN について、その周辺の天体数密度分布を求めることができたので、全 AGN について数密度分布を足し合わせ平均化することで、AGN と銀河の相関関数を求める。足しあわせるに際して、天体分布が平均に比べて極端に偏りがあるデータ、例えば近傍の銀河や銀河団が含まれているデータや、星団があるサンプルは除外する必要がある。また、観測の深さが平均にくらべ極端に浅いような場合は S/N を向上させるために除外する方がよい。

AGN サンプル毎に観測の深さが異なる場合には、足しあわせる際には補正が必要である。観測が深い程、AGN と同じ距離にある銀河の検出可能な数が増え、AGN 周辺にクラスタリングして

いる銀河の検出数が増える。ノイズ成分となる、AGN からは距離が離れていて無相関な前景銀河、背景銀河数の分散 (N) はそれらの総数の平方根 ($\sqrt{N_{bg}}$) で与えられる。したがて、AGN 周辺にクラスタリングしている銀河の S/N 比は向上する。すなわち、数密度分布における AGN 周辺での銀河の超過数は観測の深さに依存する量であり、観測深さが異なる数密度分布をそのまま足し合わせてしまっては、クラスタリングの強さを平均化することにはならない。

深さの違いを考慮してクラスタリングの強さについての平均値を求めるには、AGN と同じ距離における銀河の数密度分布から求められる、平均検出可能数密度 ρ_0 をもとめ、それに対する AGN 周辺での検出銀河の超過数比に対して平均値を求めることが必要である。この超過数比は観測の深さにはよらない量であり、相関関数はこの超過数比から 1 引いた値である。

ρ_0 は光度関数を有効限界等級まで積分することで求めることができる。詳細については文献 [1],[2] を参照のこと。有効限界等級は今回の場合カタログデータのみから求める必要があり、文献 [1] では等級分布のピーク値以上の等級をもつデータについての積算分布を求め、全体の 60% となる等級を有効限界等級の指標としている。ここではその方法に基づいて有効限界等級を求める。

ρ_0 を求めるプログラム `agn-cal-rho0` が用意されている。使い方は `-z` オプションで赤方偏移、`-b` オプションで観測波長域 (B,V,R,I,i,z,K のいずれか)、`--m-min`, `-m` オプションでそれぞれ積分範囲の下限と上限 (観測等級) を指定して実行する。

```
$ agn-cal-rho0 -z 1.8 -b R --m-min 20 -m 24
# distance modulus = 45.672126
# R band 6500
# wavelength at source: 2321.428571
# M0: -20.452772
# alpha: -1.250000
rho= 3.662206e-05
```

出力される ρ_0 の単位は Mpc^{-3}、仮定されて宇宙論パラメータは $\Omega_M = 0.3$, $\Omega_\lambda = 0.7$, $H_0 = 70\mathrm{km\ s^{-1}\ Mpc^{-1}}$ である。

課題 3 により求めた数密度分布を足しあわせる際に、B_{QG} や ρ_0, χ^2, σ_{max} を計算しその値でセレクションを行う。足し合わせを行うスクリプト `agn-density-add.sh` が用意されている。`agn-auto-sample.sh` で作成される結果ファイル、または課題 3 で作成された結果ファイルを利用する。本スクリプトは以下のように実行する。

```
$ agn-density-add.sh -l list.dat
```

`list.dat` には AGN 毎に作成されたディレクトリ名一行毎に書かれたファイルである。例えば、ディレクトリを `agn_001, agn_002, agn_003, ...`, のような名前で作成した場合は、

```
$ ls -d agn_??? > list.dat
```

により作成できる。実行すると、`hist_add.dat` というファイルに平均化された数密度分布が出力される。また、このファイルの最後の方の行には以下のような数値が記述されている。

```
...
# sum ave num
# AVE_DEN_PK: 0.191938 0.00228497 84
# AVE_DEN_10: 0.20321 0.00241917 84
# AVE_DEN_60: 0.271808 0.00323582 84
# AVE_DEN_80: 0.314744 0.00374696 84
```

`AVE_DEN_PK, AVE_DEN_10, AVE_DEN_60, AVE_DEN_80` とあるのは、それぞれ限界等級として、等級分布のピークとなる等級、ピーク以上のデータのうち等級が小さい方から 10% , 60%, 80% を

含む等級をとった場合の平均銀河数密度である。AVE_DEN_60 という行の 0.00323582 という数値を、足しあわせれた AGN サンプルに対する平均銀河数密度 ρ_0 として利用する。この値は相関距離を求める際に利用する。

課題 4. 課題 3 で求められた数密度分布を agn-density-add.sh を用いて足し合わせ平均化する。

5.4 相関距離を求める

次に足しあわされた数密度分布をプロットし、フィッティングをして相関距離を求める。フィッティングを行う root のマクロは以下のようになる。

```
1  void fit_density(double rho0)
2  {
3      double x[1000], y[1000], sig[1000], erx[1000];
4      double dummy;
5      double xmax;
6      ifstream data("hist_add.dat");
7      int ind=0;
8      int num=0, count=0;
9      string buf;
10     //
11     //
12     //
13     while (data && getline(data, buf)) {
14         if(buf.at(0) == '#') {
15             continue;
16         }
17         istrstream istr(buf.data());
18         istr>>x[ind]>>count>>dummy>>dummy>>y[ind]>>sig[ind];
19         erx[ind] = 0;
20         ind++;
21         if(count > 0) {
22             num = ind;
23             xmax = x[ind];
24         }
25         if(ind >= 1000) {
26             break;
27         }
28     }
29     data.close();
30
31     //
32     //
33     //
34     TF1 *func = new TF1("func", func_cc, 0.01, 10, 3);
35     func->SetParLimits(0, 1, 120.0);
36     func->SetParLimits(1, 5.0, 1000.0);
37     func->FixParameter(0, 10);
```

```
38      func->FixParameter(2, rho0);
39      func->SetLineColor(1);
40      func->SetLineWidth(1);
41      func->SetParameters(10.0, 0.0, rho0);
42
43      //
44      //
45      //
46      graph = new TGraphErrors(num, x, y, erx, sig);
47      graph->Fit(func, "", "", 0.0, xmax);
48
49      graph->SetLineColor(1);
50      graph->SetLineWidth(1);
51      graph->SetMarkerColor(1);
52      graph->SetMarkerSize(1.);
53      graph->SetMarkerStyle(20);
54
55      graph->SetTitle("");
56      graph->GetXaxis()->SetTitle("distance (Mpc)");
57      graph->GetYaxis()->SetTitle("density (Mpc-3)");
58
59      graph->Draw("AP");
60  }
61
62
63  Double_t func_cc(Double_t *x, Double_t *par)
64  {
65     // 相関関数を定義
66  }
```

以下が実行結果である。フィットした結果のプロットは図 9 のようになる。

```
1   root [1] .x fit-density.C(0.00323582, 5.0);
2   num = 50
3    FCN=34.2209 FROM MIGRAD STATUS=CONVERGED 126 CALLS 127 TOTAL
4                     EDM=2.79302e-08 STRATEGY= 1 ERROR MATRIX ACCURATE
5     EXT PARAMETER STEP FIRST
6     NO. NAME VALUE ERROR SIZE DERIVATIVE
7      1 p0 1.22942e+01 8.42140e-01 3.96316e-05 1.71903e-02
8      2 p1 1.17599e+01 7.38129e-02 1.48277e-06 4.04600e-01
9      3 p2 3.23582e-03 fixed
```

この結果より相関距離は $12.29 \pm 0.84\ h^{-1}\mathrm{Mpc}$ であることが分かる。ただし、この誤差は統計誤差のみであり、相関関数の導出に際して仮定された条件に関数系統誤差は別に評価する方法がある。詳細は [1, 2] を参照すること。

課題 5. 課題 4 で求めた平均化された数密度分布をプロットする。また、相関関数フィッティングによって相関距離を求める。

参考文献

[1] Shirasaki, Y. et al., 2011, PASJ, **63**, 469S

[2] Komiya, Y. et al., 2013, Apj, **775**, 43

[3] シリーズ現代の天文学 第 3 巻, 宇宙論 II – 宇宙の進化二間瀬敏史, 池内了, 千葉柾司編, 日本評論社, 2007 年

[4] Croom, S. M. et al., 2005, MNRAS, 356, 415

[5] Hickox, R., C. et al., 2009, Apj, 696, 891

[6] Ross, N. P. et al., 2009, ApJ, 697, 1634

[7] Krumpe, M., Miyaji, T., and Coil, A. L., 2010, ApJ, 713, 558

[8] Hickox, R. C. et al. 2011, Apj, 731, 117

[9] Mountrichas, G. and Georgakakis, A., 2011, arXiv:1110.5910

[10] Jenkins A.R., et al. (for the Virgo consortium), 1998, ApJ, 499, 20

[11] Smith, R. E. et al., 2003, MNRAS, 341, 1311

[12] Sheth, R. K., Mo, H. J. and Tormen, G., 2001, MNRAS, 323, 1

[13] van den Bosch F. C., 2002, MNRAS, 331, 98

[14] Veron-Cetty M.-P. and Veron P., 2010, Astron. Astrophys., 518, A10

[15] Sheth, R. K. et al. 2001, MNRAS, 323, 1

[16] Koutoulidis, L. et al. 2013, MNRAS, 428, 1382

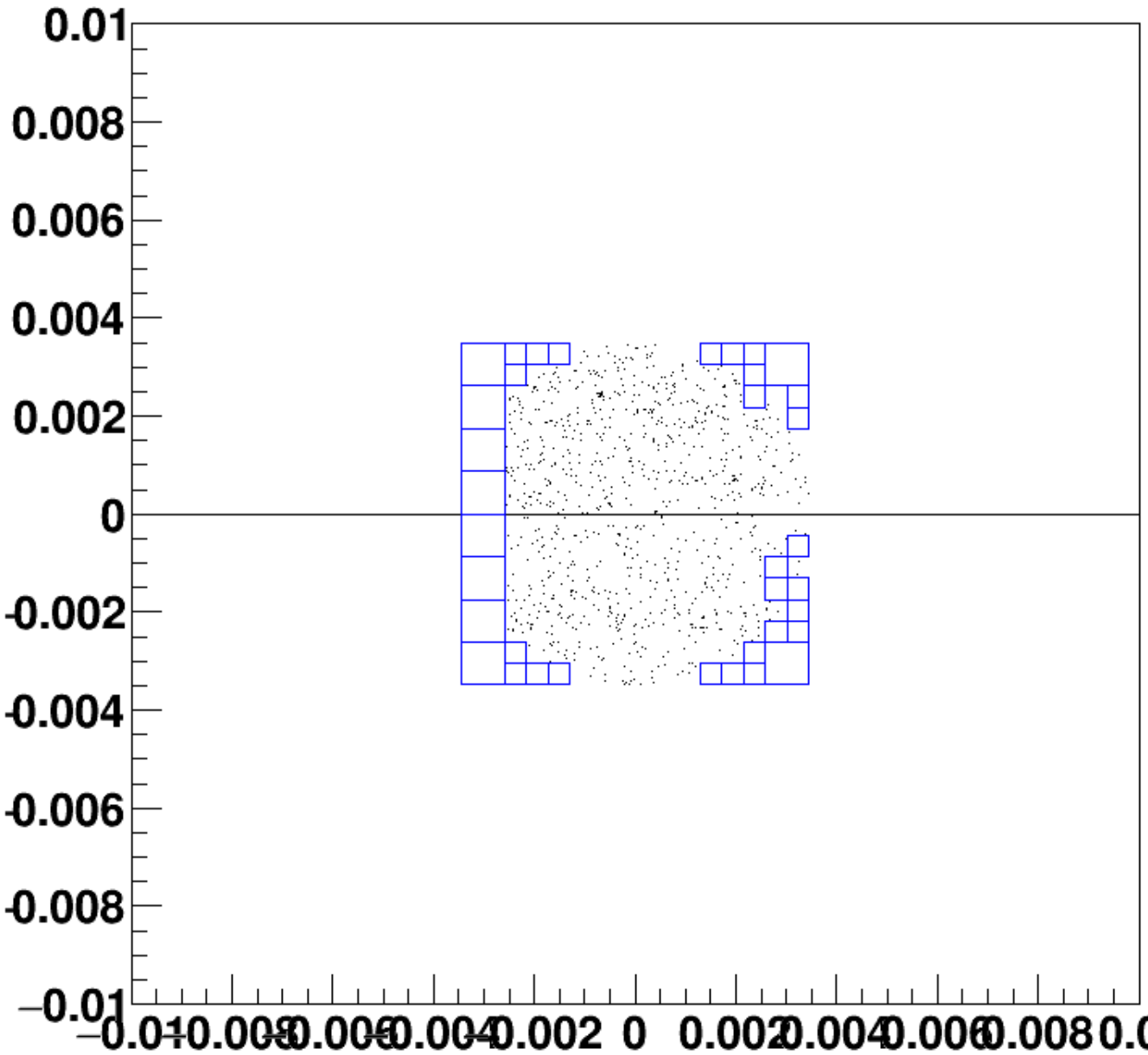
0.01
0.008
0.006
0.004
0.002
0
-0.002
-0.004
-0.006
-0.008
-0.01
-0.01
-0.008
-0.006
-0.004
-0.002
0
0.002
0.004
0.006
0.008
0.01

図 7: データ欠損領域。

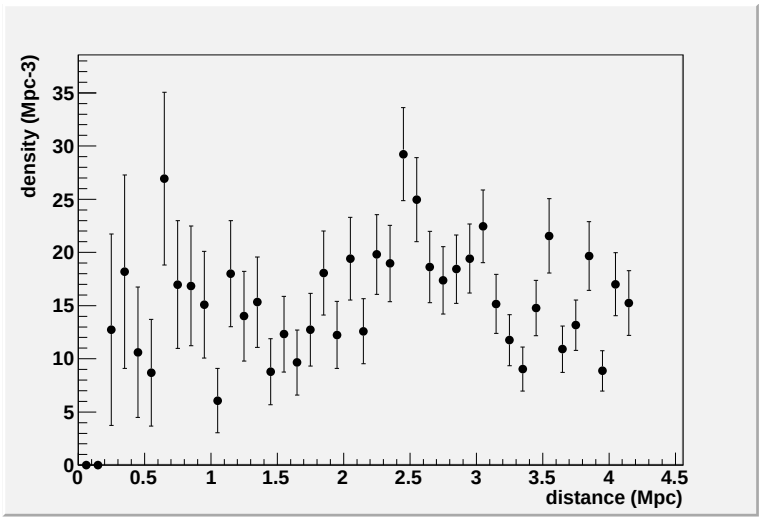

図 8: AGN からの射影距離に対する銀河数密度分布

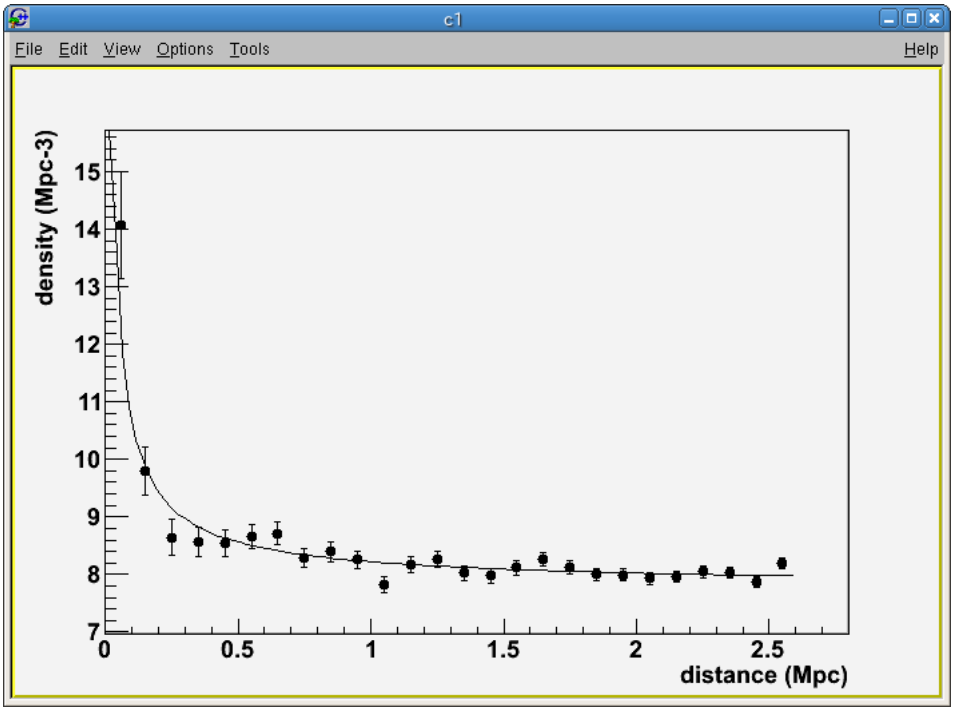

図 9: