バイオインフォマティクス

藤　博幸

目次

1. 多重比較

1-1. 統計検定 - 何故帰無仮説を棄却するのか –

統計検定では、まず**帰無仮説(null hypothesis)**を設定し、その仮説のもとで、観測されたデータが観察される確率(p-value)が計算される。この確率が、あらかじめ設定された有意水準よりも小さければ、帰無仮説を棄却し、**対立仮説(alternative hypothesis)**を採択(?)する。なぜ、このような面倒な手続きを踏むのだろうか？例えば、いくつか仮説を用意して、その中で、与えられたデータに対して最も高い確率を占めるものを選ぶという方法を取らないのは何故だろうか？多重比較の話をする前に、まずこのような手続きをとる必要があることの意味を考えよう。

例: コイントス

100 回のコイントスの試行を行い、表が 70 回、裏が 30 回出たとしよう。

このコインは、公正なコインだろうか？

帰無仮説: 裏も表も 0.5 の確率で出現

有意水準を 1%として検定してみる。

この仮説のもとでの、上記コイントスの結果が得られる確率(p-value)は二項確率で計算できる。

$$\sum_{i=70}^{100} \frac{100!}{i!\,(100-i)!}\left(\frac{1}{2}\right)^{i}\left(\frac{1}{2}\right)^{100-i}$$

```
sum(dbinom(70:100, 100, 0.5))
[1] 3.92507e-05
```

```
x <- 0.5
p <- 0.0
for (i in 70:100) {
  p <- p + gamma(101)/(gamma(i+1)*gamma(100-i+1))*x^100
}
print(p)
3.92507e-05
```

有意水準 1%とすると、p-値はそれよりも小さいので、コインが公正であるという帰無仮説は棄却される。

1-1-1. 反証から棄却へ

科学的命題の例として、アインシュタインの一般相対性理論を考える。この理論によれば、重力のポテンシャルによりスペクトルが赤方偏移することが予測される。もし、赤方偏移が観測されなければ、この理論は間違っているということになる。赤方偏移が観測されれば、とりあえずその理論を採択するという形をとる。科学的命題とは、このように経験によって間違っていることが証明される可能性（反証可能性）がなければならない。反証可能性は科学と疑似科学を区別する基準となるものである。

明日は雨が降るか降らないかだ

この命題は反証できないし、新しい情報をもたらさない。

渋川春海（安井算哲, 1639-1715）は江戸時代の暦学者である。当時、日本は 862 年に唐から伝わった宣明暦が使われていたが、かなりの誤差が生じており、日蝕、月蝕を正しく予測できなくなっていた。渋川春海は天体観測の結果に基づき、元の授時暦への改暦を提案した。しかし、授時暦による日触予測に失敗により、改暦は頓挫した。この予測の失敗は上記の反証に相当する。その後の研究で、中国の暦をそのまま採用しても、中国と日本の里差により誤差が出ることから新たな大和暦を提案し、後に貞亨暦として採択され、改暦された。渋川春海を主人公とした冲方丁の小説「天地明察」の中で、渋川春海の改暦事業への取り組みの様子を知ることができる。

次の命題を考えよう。

夕焼けの翌朝は 90%の確率で晴れである。

このような形の仮説を、**統計的仮説**と呼ぶ。
この命題は反証できない。経験データを積み重ねても、夕焼けの翌朝が晴れである頻度が 50%のことも 100%のことも起こりうる。それは、上記の統計的仮説が誤りであることにはならない。
そこで、反証の代わりに提案されたのが統計的仮説の棄却である。
ある仮説のもとで、観測データが実現される確率(p-値)を求めた時に、それが小さな値であれば、その仮説は観測データと両立しないものとして棄却する。
棄却(rejection)は、**反証(refutation)**ではない。反証は仮説が正しくないことを証明することであるのに対し、棄却は統計的仮説が正しいことを疑うのに十分な強い証拠をもとに、その仮説の受容を拒否することである。そのため、仮説が正しい可能性が残されている。この仮説が正しい可能性を制御するのに導入されるのが有意水準である。有意水準を 1%と設定した時に、それが意味することは何であろうか？次にこの有意水準の意味を考えよう。

参考文献

天地明察　冲方丁著　角川書店　(2014)

科学哲学　ドミニック・ルクール著　(沢崎、竹中、三宅訳)　白水社 (2005)

理系人に役立つ科学哲学　森田邦久著　化学同人 (2010)

統計的証拠とその解釈　細谷雄三著 牧野書店 (1995)

1-1-2.　第一種の過誤

帰無仮説が真の時に、それを棄却する誤りを第一種の過誤と呼ぶ。この**第一種の過誤**を犯す確率を制御するのが、**有意水準(significance level)**である。有意水準は α で表されることが多く、一般には α =0.01 や α =0.05 に設定されることが多い。

次の例を考えてみよう。

対応のない場合の 2 標本の差の検定を考える。対応がないというのは、2 群が独立であることを意味する。2 群の平均値に差があるかを検討するのに用いられ、*t*-検定が用いられる。帰無仮説は、平均値に差がないと設定される。また *t*-検定では、母集団が正規分布に従うことが仮定される。

例えば、正常細胞とがん細胞で、遺伝子 X の発現量を 10 回ずつ計測したものとする。正常細胞とがん細胞で、遺伝子 x の発現量に差があるかどうかを見る場合に使われるのが *t*-検定である。

第一種の過誤における有意水準の意味を調べるために次の解析を行ってみよう。

まず、A 群のデータを、平均 0, 標準偏差 1 の正規分布に従う乱数を 100 個取り出して作成する。同様に、B 群のデータを、平均 0, 標準偏差 1 の正規分布に従う乱数を 100 個取り出して作成する。A 群も B 群も同じ正規分布に従っているので、帰無仮説が成立しており、両者には差がないはずである。この 2 群の平均値の差の検定を有意水準 1%で実施する。これを 1000 回繰り返した時、何回棄却（第 1 種の過誤）されるかを調べよう。

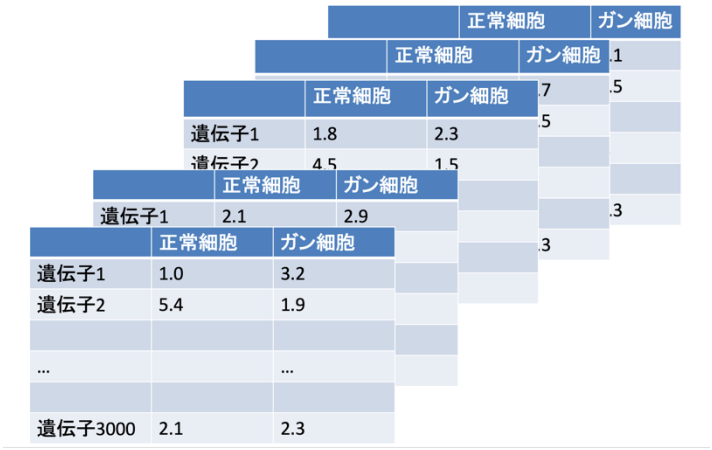
```
n <- 0
for (i in 1:1000) {
   A <- rnorm(100, 0, 1)
   B <- rnorm(100, 0, 1)
   if (t.test(A, B)$p.value < 0.01)  n <- n + 1
 }
print(n)
```

何回か実行してみよう。*n* は 10 前後の値として得られる。1000 回のうちの 10 回ということは、第 1 種の過誤が 1 %程度生じているということである。つまり、帰無仮説が成立している時に、帰無仮説が棄却される危険性が有意水準である 1 %に抑えられている。

有意水準を 5%にしても、同様の結果が得られることを確認しよう。

1-2.　多重比較の問題点

近年の次世代シークエンサの発達により、組織や細胞レベルので、数千の遺伝子の発現量を一度に計測することができるようになってきている。このような解析をトランスクリプトーム解析と呼ぶ。ヒトの正常細胞とがん細胞についてトランスクリプトーム解析をそれぞれ 10 回ごと実施したとする。その結果、3000 個の遺伝子それぞれについて、正常細胞、がん細胞における遺伝子の発現量が 10 個ずつ得られることになる。正常細胞とがん細胞を比較した時に、がん細胞で発現量が正常細胞よりも増加している、あるいは減少している遺伝子が検出できれば、がんに関連する遺伝子の候補を得ることができる。この時に、各遺伝子の 10 個の発現量は正規分布に従うものと仮定し、両細胞で遺伝子の発現量に差はないという帰無仮説のもとで、遺伝子ごとに *t*-検定を繰り返すのは処理として適切ではない。3000 個の遺伝子について *t*-検定を繰り返すという状況は、1-2 で見た仮想的な *t*-検定の繰り返しに対応しており、第 1 種の過誤を生じる可能性が出てくるからである。この問題は、**多重比較**あるいは**多重検定**と呼ばれている。



多重比較は、上記のゲノムワイドな発現解析ばかりでなく、一つの実験のセッションの中で複数回の検定を繰り返す時には必須の統計処理となってきている。

1-3. Bonferroni 法

1-3-1. family wise error rate

検定を繰り返せば繰り返すほど、偶然棄却される帰無仮説が増える。複数回繰り返した検定全体において帰無仮説が棄却される可能性を、**family wise error rate** とよぶ。

1 回の検定で偶然棄却される確率　　$1 - 0.95 = 0.05$

2 回の検定で、1 回でも偶然棄却される確率　　$1 - (0.95)^2 = 0.0975$

3 回の検定で、1 回でも偶然棄却される確率　　$1 - (0.95)^3 = 0.142625$

20 回の検定で、1 回でも偶然棄却される確率　　$1 - (0.95)^{20} = 0.6415141$

100 回の検定で、1 回でも偶然棄却される確率　　$1 - (0.95)^{100} = 0.9940795$

http://www.med.osaka-u.ac.jp/pub/kid/clinicaljournalclub1.html　より

Familywise error rate を調整する方法として、(1)F統計量や t統計量等の**統計量**に基づいた方法と　(2)それらの統計量から算出された ***p*値のみ**を操作する方法がある。

(1) 統計量を用いた方法(1)としては、F 統計量を用いた **Fisher's least significant difference (Fisher's LSD)法**、 t 統計量を用いた **Tukey's honestly significant difference (Tukey's HSD)法**、 t 統計量を用いて control 群と非コントロール群の比較のみを行う **Dunnet 法**等が開発された。これらは、分散分析などで使用されている

(2) 統計量ではなく、p値を調整する方法(2)としては、**Bonferroni 法**や **Holm 法**が挙げられる。これらの方法は、統計量に依存しないため、どのような検定に対しても利用でき、汎用性が高い。

http://www.med.osaka-u.ac.jp/pub/kid/clinicaljournalclub1.html　より

ここでは後者の方法について解説

1-3-2. Bonferroni 法の考え方

$$\text{Family Wise Error Rate} = 1-(1-\alpha)^m$$

$$= 1-\sum_{i=0}^{m}\frac{m!}{i!(m-i)!}(-\alpha)^m$$

$$\approx 1-(1-m\alpha) = m\alpha$$

有意水準 α は 0 より大きく 1 より小さいので二次以上の項は小さくなるので 0 で近似し、$i=0$ と 1 の項のみを使う

先に見たように FWER は、m回検定を繰り返した時に、少なくとも 1 回偶然棄却される確率を表す。

そこで、α を α/mに置き換えておくと、FWER を α にすることができる。

https://www.slideshare.net/yuifu/fdr-kashiwar-3　より

1-3-3.　Bonferroni 法の手続き

検定総数が Nの場合、それぞれの検定の有意水準を α から α/Nに変更する方法が、Bonferroni 法である。検定総数が 20 ならば、20 個の検定全てにおいて、有意水準を 0.05/20 = 0.0025 に変更する。

遺伝子が 10000 個あり、正常細胞とガン細胞で発現差を検定したい時、通常の有意水準を 1%とすると、$0.01/10000=1\text{x}10^{-6}$ に変更しておく。

(1) 各遺伝子の p-value が $1\text{x}10^{-6}$ より小さい時有意とする

(2) あるいは、各遺伝子の発現差の p-value を 10000 倍しておいて、それが 0.01 より

小さい場合を有意とする。

http://www.med.osaka-u.ac.jp/pub/kid/clinicaljournalclub1.html

https://www.slideshare.net/yuifu/fdr-kashiwar-3　より

具体例で考えよう。

正常細胞とガン細胞を比較し、8 つの遺伝子について、その発現差の *p*-value が *t* 検定によって次のように得られているとする。

	遺伝子 1	**遺伝子 2**	**遺伝子 3**	**遺伝子 4**	**遺伝子 5**	**遺伝子 6**	**遺伝子 7**	**遺伝子 8**
p-value	0.45	0.05	0.016	0.0005	0.019	0.009	0.03	0.091

この結果を R を使って Bonferroni 補正で検定してみよう。

１％有意水準で検定を実施する

```
pv <- c(0.45, 0.05, 0.016, 0.0005, 0.019, 0.0014, 0.03, 0.091)
```

補正なしだと、遺伝子 4, 遺伝子 6 の発現に有意差がある.

(1) で述べた方法と同じやり方

```
tv1 <- pv < 0.01/8
tv1
[1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

(2)で述べた方法と同じやり方

```
pv2 <- pv*8
tv2 <- pv2 < 0.01
tv2
[1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

いずれも同じ結果を与える。

遺伝子 4 のみが有意となる。

R の p.adjust 関数でも Bonferoni 補正を行える。

```
p.adjust(pv, method="bonferroni", n=length(pv))
[1] 1.000 0.400 0.128 0.004 0.152 0.0112 0.240 0.728
```

```
p.adjust(pv, method="bonferroni", n=length(pv)) < 0.01
[1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

1-4. Benjamini-Hochberg 法

1-4-1. FWER から FDR へ

FP (帰無仮説が正しいのに、あやまって帰無仮説を棄却すること；正常細胞とガン細胞で、遺伝子 A の発現に差がないのに、棄却してしまうこと）を抑えようとしているが、
そのため TP (帰無仮説が間違っていて、帰無仮説が正しく棄却されること；正常細胞とガン細胞で、遺伝子 A の発現に差があり、帰無仮説が棄却されること）であっても棄却されにくい。すなわち、
Bonferroni 法などの FWER は、保守的すぎて、発現に差のある遺伝子を検出しにくい。

そこで、FP が混じっても良いので、TP を増やす手法が考えられた。この時の指標となるのが、False Discovery Rate (FDR)である。すなわち、FP がどれだけ含まれているか(FDR)を推定して、それを新たな基準とする。この基準は、有意水準の p-値と区別するため、**q-値**とよぶ。

FDR = FP / (FP + TP)
TP = true positive:対立仮説が正しく、帰無仮説が棄却
FP = false positive:帰無仮説が正しいのに、帰無仮説が棄却

1-4-2. Benjamini-Hochberg 法の原理

$p_1 < p_2 < \cdots < p_i \cdots < p_m$ とする
i 番目(p_i)を検討する。
α を false positive を含む割合とする.
$m \times \alpha$ は、m 回の検定中 false positive の期待回数
i / m は, m 回の検定中 i 個の占める割合
すると, $m \times \alpha \times i / m = \alpha \times i$ は、i 回の検定の中で false positive を起こす回数(=棄却されたのに本当は帰無仮説が正しいもの)の期待値と見なせる。
FP の上限として $\alpha \times i$ を考える。

今、p_i 以下の p-value で棄却する。
すると positive となる検定が i 個ある。
この i 個には true positive も false positive も含まれているとすると FP + TP = i
FDR = FP/(FP + TP) = FP/i $\leq$ $\alpha \times i / i = \alpha$
このようにして FDR を α 以下になるよう制御できる。

$p_i <= \alpha \times i / m$ を満たす時、$p_1 \sim p_i$ を有意とする。

この式を書き直すと

$p_i \times m / i <= \alpha$ を満たす時、$p_1 \sim p_i$を有意とする。

$q_i = p_i \times m / i$　を p-value に対して、**q-value** とよぶ。

1-4-3.　Benjamini-Hochberg 法の手続き

m 回の多重検定の場合

(1)　p-value を昇順に並べる（$p_1 < p_2 < \cdots < p_m$）

(2)　$i = m$ とする

(3)　$p_i <= \alpha \times i / m$ を満たす時、$p_1 \sim p_i$を有意とする。

そうでなければ、i を　$i - 1$ にして，上の条件

を確認する。

※ $i = 1$ になっても、条件を満たさない場合は有意なものはないとする。

https://www.slideshare.net/antiplastics/dna-21259335　より

5 つの遺伝子について正常細胞とがん細胞の発現差の p-値が以下のようになったとする。

	***p*-value**
Gene 1	0.21
Gene 2	0.001
Gene 3	0.1
Gene 4	0.06
Gene 5	0.005

まず、これを p-値でソートする。

	***p*-value**
Gene 2	0.001
Gene 5	0.005
Gene 4	0.06
Gene 3	0.1
Gene 1	0.21

上に書いた手続きに従い、p-値を q-値に変換する。

	***p*-value**		***q*-value**
Gene 2	0.001	0.001 x (5/1)	0.005
Gene 5	0.005	0.005 x (5/2)	0.0125
Gene 4	0.06	0.06 x (5/3)	0.1

Gene 3	0.1	0.1 x (5/4)	0.125
Gene 1	0.21	0.21 x 5/5	0.21

FDR の閾値を 0.05 として *p*-value の大きいものから順番に検討していくと、Gene 2 と Gene 5 の発現差が有意であることがわかる。

1-4-4. R で計算してみよう

Gene1~Gene5 の *p*-value をベクトルとして表現

```
pv <- c(0.21, 0.001, 0.1, 0.06, 0.005)
```

pv を昇順にソート

```
spv <- sort(pv)
spv
[1] 0.001 0.005 0.060 0.100 0.210
```

q-value を記憶させる空ベクトルを作成

```
qv <- c()
```

for 文で q-value に変換

```
for (i in 1:length(spv))  qv <- c(qv, spv[i]*length(spv)/i)
qv
[1] 0.0050 0.0125 0.1000 0.1250 0.2100
```

上で説明した計算結果と一致することを確認

次に p.adjust 関数を使って計算してみる。

```
p.adjust(pv, method="BH", n=length(pv))
[1] 0.2100 0.0050 0.1250 0.1000 0.0125

p.adjust(pv, method="BH", n=length(pv)) < 0.05
[1] FALSE  TRUE FALSE FALSE  TRUE
```

p.adjust 関数を使うと、元の遺伝子の並び順のままで検定結果を得ることができる。

1-5. Storey 法

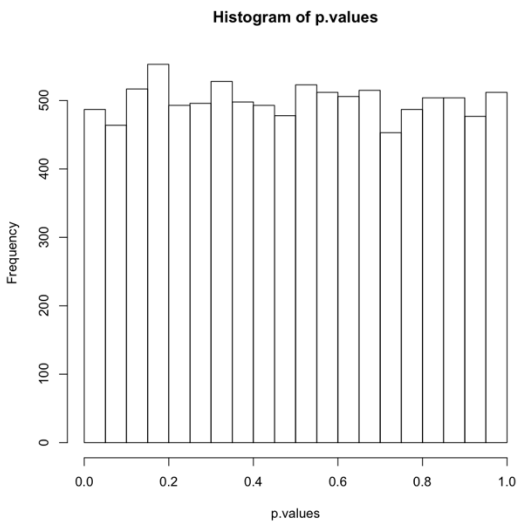
1-5-1. Benjamini-Hochberg 法の問題点

Benjamini-Hochberg 法のキモは、*q*-value の計算を、

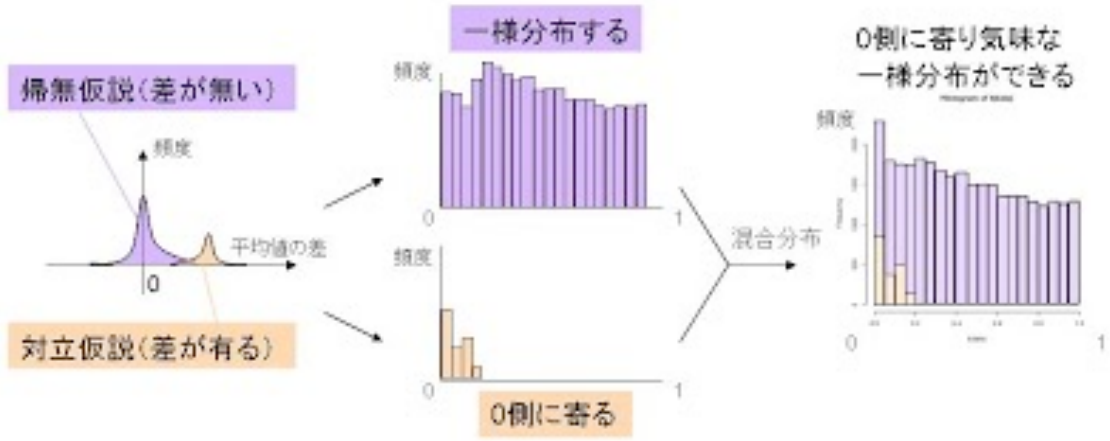
$q_i = p_i \times m / i$

としている点にあるが、これは p_iを基準とした時、その時の false positive の数が $p_i \times m$ と仮定していることになる。言い換えると、p_i が一様分布していることを仮定している。帰無仮説が全て正しい場合(ex. 10000 個全ての遺伝子で正常細胞とガン細胞における発現量に差はない)、p-value は一様分布に従う。

```
pvalue <- c()
for (i in 1:10000) {
  pvalue <- c(pvalue, t.test(rnorm(10,0,1), rnorm(10,0,1))$p.value)
}
hist(pvalue)
```



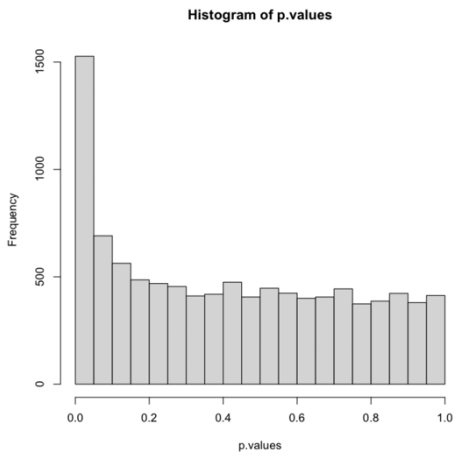
対立仮説が正しい場合の検定が混ざっていると、p-value が低いところの密度が高くなる。



https://sites.google.com/site/scriptofbioinformatics/maikuroarei-guan-xi/fdr-zhi-yu-r

より

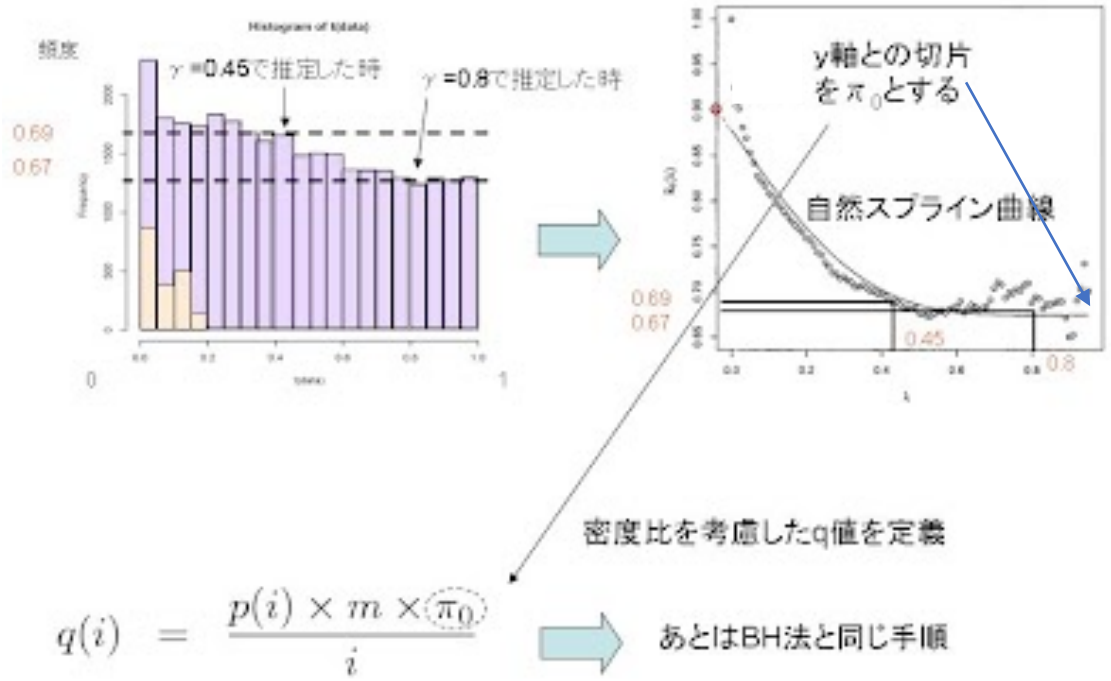
```
N <- 8000
M <- 10000 -N
p.values <- c()
for (i in 1:N) {
  p.values <- c(p.values,t.test(rnorm(10,0,1), rnorm(10,0,1))$p.value)
}
for (i in 1:M) {
  p.values <- c(p.values, t.test(rnorm(10,0,1), rnorm(10,1,1))$p.value)
}
hist(p.values)
```



帰無仮説が正しい場合と対立仮説が正しい場合が混ざり込んでいることを考慮して多重比較の補正を行う方法が開発されている。Storey 法はそのような方法の一つであり、R で利用することができる。

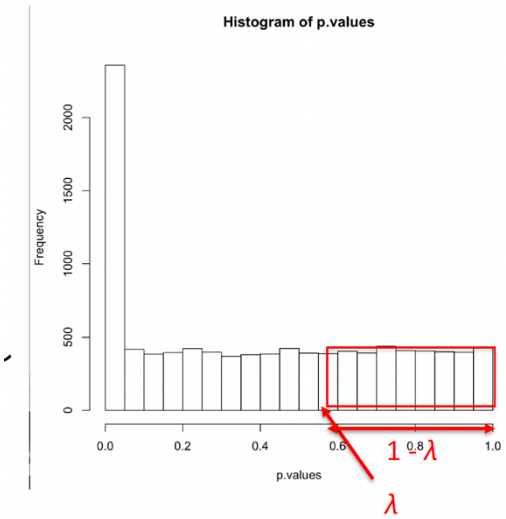
1-5-2. Storey 法

今、複数回の検定の中で、帰無仮説が正しい場合の p-値の分布と、対立仮説が正しい場合の p-値の分布が各々 π_0：1 - π_0の比で混ざっているとする。Storey 法では、このπ_0が最初に推定される。



https://sites.google.com/site/scriptofbioinformatics/maikuroarei-guan-xi/fdr-zhi-yu-r

より　（ただし、π_0の推定の図に間違いがあるので注意）



まず、π_0の推定の仕方を見てみよう。

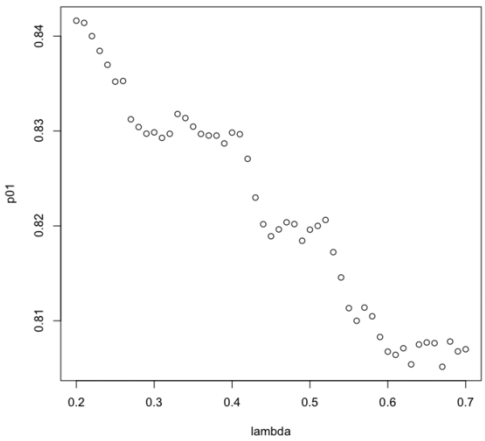
λというパラメータを考え、これを 0.0~1.0 の範囲で動かす。

各λに対して、π_0を次のように推定する。また、この推定値を$\pi_0(\lambda)$と表す。

$$\pi_0(\lambda) = \left(\lambda\text{より大きな}p-\text{value の数}\right)/m(1-\lambda)$$

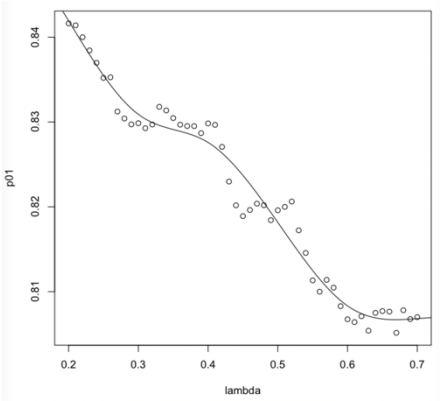
このようにして得られたλに対して、$\pi_0(\lambda)$をプロットする（上図参照）。このプロットを自然スプライン関数でフィッティングして、λ=1に外挿した時の$\pi_0(\lambda)$をπ_0として使う。

先に作成した 8000:2000 の割合で、帰無仮説が正しい場合と、そうでない場合が混じっている例で、実際にこの推定を試してみよう。




```
lambda <- seq(0.2,0.7,0.01)
p01 <-c()
for (i in lambda) {
   p01 <- c(p01,
length(p.values[p.values>i])/((1-i)*(N+M)))
}
plot(lambda,p01)
```

このプロットを**自然スプライン曲線**で近似する。**スプライン**とは、与えられた点を通る曲線を、区分的に多項式で近似する方法である。R にはスプラインを行う関数やパッケージが複数用意されている。ここではパッケージ splines を使って自然スプラインを行ってみよう。




```
library(splines)
plot(lambda, p01, xlim=c(0.1,0.8), ylim=c(0.79,0.85))
df <- data.frame(lambda=lambda, p01=p01)
fml <- lm(p01 ~ ns(lambda, df=5), data=df)
plot(df)
lmd <- seq(0.1,1.0,length.out=200)
lines(lmd, predict(fml, data.frame(lambda=lmd)))
```

このスプラインでプロットが近似できていることがわかる。
このスプライン関数で lambda を 1 に外挿した時の、p01 を求めてみる。

```
lmd <- seq(0.1,1.0,length.out=200)
extp <- predict(fml, data.frame(lambda=lmd))
print(extp[200])
```

これにより、lambda が 1 の時の p01 は 0.8086553 と推測される。今回、1 万回の検定の中で、帰無仮説が正しい検定が 8000 回行われているので、この推定値はほぼ正しい値を与えている。
このようにして推定された π_0 を Benjamini-Hochberg 法の q-value に乗じて検定を行うが、それ以外の手続きは Banjamini=Hochberg 法とほぼ同じである。

以下は、この後に使用する qvalue パッケージの関数を元に検定の手続きを作成したものである。

```
m <- length(p.values)
o <- order(p.values, decreasing=TRUE)
ro <- order(o)
i <- length(p.values):1
qvals <- extp[200]*pmin(1, cummin(p.values[o]*m/i))[ro]
```

最後の行、“qvals <- extp[200]*pmin(1, cummin(p.values[o]*m/i))[ro]”について解説する。

pminは、minの並列版である。

```
pmin(c(0.9, 3.0, 1.0, 6.8), c(1.0, 0.8, 4.3, 1.0))
```

の出力は、

```
[1] 0.9 0.8 1.0 1.0
```

となる。また、

```
pmin(1.0, c(1.0, 0.8, 4.3, 1.0))
```

の出力は、

```
[1] 1.0 0.8 1.0 1.0
```

となる。

cumminは、ある部位までの最小値を返す。

```
cummin(c(0.8, 1.0, 6.0, 0.7, 0.9, 0.6, 0.1, 0.8, 0.6))
[1] 0.8 0.8 0.8 0.7 0.7 0.6 0.1 0.1 0.1
```

※ pmaxやcummaxもある

p.values[o]は、p.valuesを降順に並べ直している。

最後の[ro]は元の順番に戻す処理である。

この処理で本当にFDRが制御できているかを調べる。

0.05を閾値とする場合を考えよう。以下の数値は、上記のp-valueの発生に乱数を使っているため、人によって異なっている。

Positive = 帰無仮説を棄却した総数は以下のようにして求められる。

```
length(qvals[qvals<0.05])
[1] 205
```

false positiveの数は以下のようにして求められる。

```
q1 <- qvals[1:N]
length(q1[q1<0.05])
```

```
[1] 11
```

True positiveの数は次のようにして求められる。

```
q2 <- qvals[(N+1):(N+M)]
length(q2[q2 < 0.05])
[1] 194
```

FDRは、FP/(FP+TP)なので次のように計算される。

11/205

```
[1] 0.05365854
```

これは設定した 0.05 に近い値になっている。

Benjamini-Hochberg 法では、*p*-value のアナロジーとして小文字の *q*-value が使われているが、Storey 法では、大文字の Q を使った *Q*-value が使われている。

次に、R のパッケージ qvalue を使って、Storey 法を行ってみよう。

1-5-3. パッケージ qvalue

パッケージ qvalue は、CRAN でなく BioConductor から配布されている。インストールの際に、注意すること。

```
library(qvalue)
x <- qvalue(p.values)
y <- x$qvalues

length(y[y<0.05])
```

これにより positive が 207 個あることがわかる。

a <- y[1:8000]

b <- y[8001:10000]

length(a[a<0.05])

これにより false positive が 11 個あることがわかる。

length(b[b<0.05])

この操作から、true positve は 196 個あることがわかる。

すると FDR は、

11/207 で 0.0531401 となり、0.05 あたりで FDR が抑えられていることがわかる。先に書いたように乱数を使っているので、数値は人によって異なっている。また、パッケージを使わなかった場合との違いは、スプラインの部分の計算が異なるためと思われる。実際、π_0 の推定値が異なっている。

```
str(x)
List of 8
```

```
 $ call     : language qvalue(p = p.values)
 $ pi0      : num 0.804
 $ qvalues  : num [1:10000] 0.757 0.785 0.797 0.746 0.664 ...
 $ pvalues  : num [1:10000] 0.77 0.894 0.96 0.728 0.464 ...
 $ lfdr     : num [1:10000] 1 1 1 1 0.963 ...
 $ pi0.lambda: num [1:19] 0.892 0.865 0.849 0.842 0.835 ...
 $ lambda   : num [1:19] 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5 ...
 $ pi0.smooth: num [1:19] 0.871 0.863 0.856 0.849 0.842 ...
 - attr(*, "class")= chr "qvalue"
```

これから、$\pi 0$ の推定値は x$pi0 で得られるが、この値は 0.8039532 となり、上の推定値と微妙に異なっていた。

関数 qvalue のコードは、library を呼び出した後に、コンソールに qvalue とタイプすると見ることができる。上の qvalue の手続きは、このコードを元に作成した。

```
function (p, fdr.level = NULL, pfdr = FALSE, lfdr.out = TRUE,  pi0 = NULL, ...)
{
    p_in <- qvals_out <- lfdr_out <- p
    rm_na <- !is.na(p)
    p <- p[rm_na]
    if (min(p) < 0 || max(p) > 1) {
        stop("p-values not in valid range [0, 1].")
    }
    else if (!is.null(fdr.level) && (fdr.level <= 0 || fdr.level >
        1)) {
        stop("'fdr.level' must be in (0, 1].")
    }
    if (is.null(pi0)) {
        pi0s <- pi0est(p, ...)
    }
    else {
        if (pi0 > 0 && pi0 <= 1) {
            pi0s = list()
            pi0s$pi0 = pi0
        }
        else {
            stop("pi0 is not (0,1]")
        }
```

```
    }
    m <- length(p)
    i <- m:1L
    o <- order(p, decreasing = TRUE)
    ro <- order(o)
    if (pfdr) {
        qvals <- pi0s$pi0 * pmin(1, cummin(p[o] * m/(i * (1 -
            (1 - p[o])^m))))[ro]
    }
    else {
        qvals <- pi0s$pi0 * pmin(1, cummin(p[o] * m/i))[ro]
    }
    qvals_out[rm_na] <- qvals
    if (lfdr.out) {
        lfdr <- lfdr(p = p, pi0 = pi0s$pi0, ...)
        lfdr_out[rm_na] <- lfdr
    }
    else {
        lfdr_out <- NULL
    }
    if (!is.null(fdr.level)) {
        retval <- list(call = match.call(), pi0 = pi0s$pi0, qvalues = qvals_out,
            pvalues = p_in, lfdr = lfdr_out, fdr.level = fdr.level,
            significant = (qvals <= fdr.level), pi0.lambda = pi0s$pi0.lambda,
            lambda = pi0s$lambda, pi0.smooth = pi0s$pi0.smooth)
    }
    else {
        retval <- list(call = match.call(), pi0 = pi0s$pi0, qvalues = qvals_out,
            pvalues = p_in, lfdr = lfdr_out, pi0.lambda = pi0s$pi0.lambda,
            lambda = pi0s$lambda, pi0.smooth = pi0s$pi0.smooth)
    }
    class(retval) <- "qvalue"
    return(retval)
}
<bytecode: 0x7ffc7750b518>
<environment: namespace:qvalue>
```

2.　ネットワーク解析

2-1.　ネットワークデータ

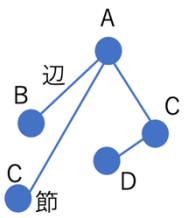
2-1-1. 階層的システムとしての生体システム

システムとは、ある特定の入力に対して、決まった出力を与えるものである。例えば、細胞というシステムを考えたときに、それに対して、薬剤 A を与えると形状が丸まり、薬剤 B を与えるとペプチド X が分泌されるというように、特定の入力（刺激）」が与えられると、決まった応答を返すことを意味する。システムは、複数の要素で構成されており、それらの要素の相互作用による集団的特性として、上記の入力-応答が実現される。細胞システムを考えると、その要素は水、DNA, RNA, 脂質、タンパク質などの分子である。しかし、そのような分子を寄せ集めただけでは、細胞というシステムは実現されない。それらの分子が相互作用し、集団的特性を発揮できるように配置される必要がある。このように、システムを考える際には、そのシステムを構成する**要素**と、それらの**相互作用**の両方を理解する必要がある。

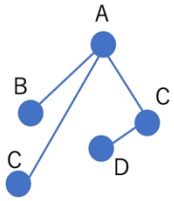
生体システムを考える際の、もう一つのポイントは**階層性**である。生体システムの最小単位をどこに取るかは人によって異なるが、細胞システムを最小単位とすることが多いと思う。細胞はそれ自体がシステムであるが、それらの細胞が集まり、相互作用することで、組織や臓器というシステムが構成される。これは空間的なスケールが一つ上がったシステムを考えることを意味する。組織や臓器といったシステムは相互作用し、個体というシステムを構成する。個体は、互いに相互作用することで個体群というシステムを形成する。ヒトの場合は、社会システムが形成される。また、異なる種の個体群が相互作用し合うことで生態系というシステムが形成される。空間的な階層が異なるシステムを扱う際にも、何が要素で、それらがどのように相互作用しているかを理解する必要がある。

ここでは分子レベルの相互作用ネットワーク、特にタンパク質の相互作用ネットワークを取り扱う。タンパク質の機能は、**生化学的機能（biochemical function）**と**生物学的機能(biological function)**に大別される。生化学的機能とは、酵素活性やリガンド結合能など、そのタンパク質自身の属性としての機能である。生化学的機能はタンパク質自身の属性であるので、そのアミノ酸配列や立体構造などから蚊一隻していくことができる。一方、生物学的機能とは、代謝パスウェイやシグナル伝達経路などを介した高次の生命現象（行動、記憶など）との関わりにおける機能を意味する。すなわち、生物学的機能は、そのタンパク質単体としての機能ではなく、複数のタンパク質との相互作用に基づく機能である。そのため、生物学的機能を解析するには、一つのタンパク質に注目するのではなく、それがどのようなタンパク質と、どのように相互作用するかを知る必要がある。このような相互作用を解析するための情報技術を**ネットワーク解析**と呼ぶ。

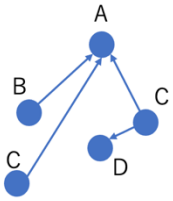
2-1-2. ネットワーク解析の基本



相互作用ネットワークは、数学的には**グラフ**と呼ばれる。グラフは、**節**(node or vertex)と**辺**(edge)から構成される。2-1-1 の用語と対応させると、節はシステムの要素を表し、辺は要素間の相互作用を表すことになる。グラフは、**有向グラフ**と**無向グラフ**に大別される。有向グラフとは、辺が矢印で表されるもので、相互作用に方向性がある場合に使われる。例えば、遺伝子の発現



無向グラフ



有向グラフ

制御ネットワークは、遺伝子を要素として構成され、ある遺伝子 X が別の遺伝子 Y の発現を制御している場合に、X から Y に矢印を引く形で表現される。これに対して、タンパク質間相互作用には、そのような方向性がないので、辺に向きのない無向グラフで表現される。本講義では、タンパク質間相互作用を例として取り上げるため、無向グラフを用いる。

グラフは図として表現されるが、コンピュータで扱うためにはテキストとしての表現が必要になる。代表的なグラフのテキスト表現として**隣接行列**と**辺リスト**がある。今、5 つの要素 A, B、C、D、E からなるネットワークを考える。A-B, A-C, B-E, C-D, C-E の相互作用があるものとする。このネットワークを、各行、各列が A, B, C, D, E に対応する行列で次のように表現する。

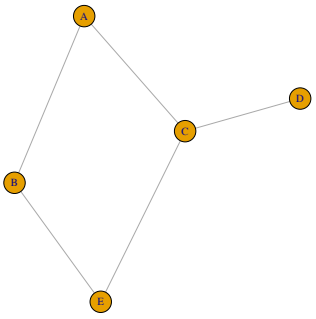
	A	B	C	D	E
A	0	1	1	0	0
B	1	0	0	0	1
C	1	0	0	1	1
D	0	0	1	0	0
E	0	1	1	0	0

この行列では、相互作用している要素に対応するところには 1、相互作用しない要素の間には 0 が置かれている。このような行列を**隣接行列**と呼ぶ。

これに対して**辺リスト**は左図のように表現される。辺リストで表現されたグラフは、R のパッケージ igraph を使って可視化や解析を行える。

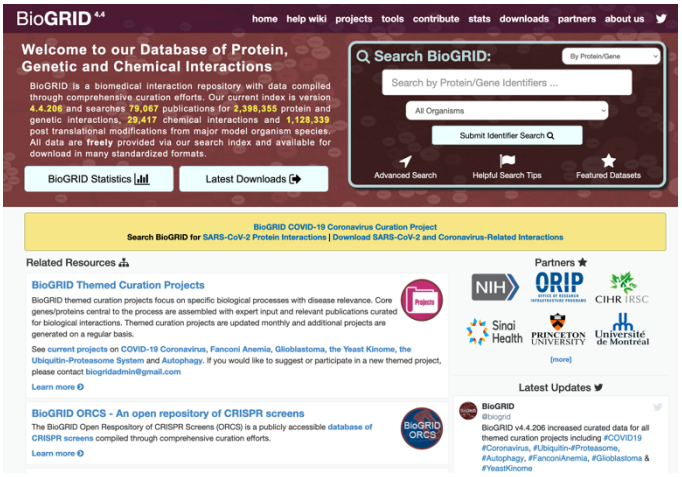
A	B
A	C
B	E
C	D
C	E

```
data <- data.frame(ia=c("A","A","B","C","C"), ib=c("B","C","E","D","E"))
library(igraph)
g <- graph.data.frame(data, directed=F)
plot(g)
```



上記処理で左のようなネットワーク図を作成できる。graph.data.frame は、igraph の関数であり、データフレーム data を、igraph のオブジェクト g に変換する関数である。変換されたオブジェクトを引数とした plot で可視化できる。これ以外にも覚えておくべき基礎的な事柄はあるが、本講義に必要な事柄のみにとどめる。

いくつかタンパク質間相互作用データベースが作成され、公開されているが、ここでは、その代表的なデータベースの一つである BioGRID から抽出したデータを使うので、BioGRID について簡単に紹介する。

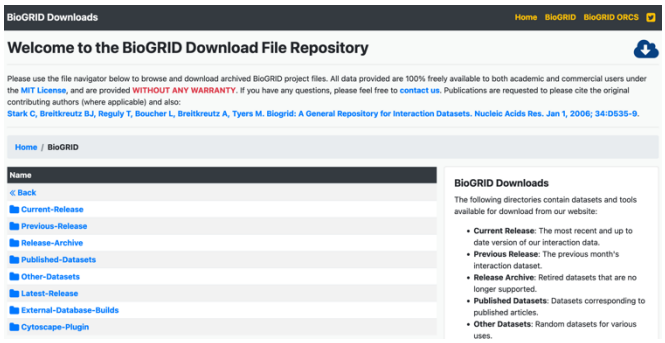


BioGRID HP

次の URL あるいは BioGRID で検索すれば、BioGRID の HP に移動できる。

https://thebiogrid.org/

BioGRID では、相互作用は二項関係で記述されており、相互作用するペアごとに、gene symbol, 由来する生物種、実験手法などの注釈を 1 行にまとめる形で記述されている。収集されている実験手法は大きく、physical な手法(Affinity Capture-MS、Affinity Capture-RNA、Protein-RNA、Two-hybrid など)と genetic な手法（Dosage Growth Defect、Negative Genetic、Phenotypic Enrichment、Synthetic Rescue など）に分かれている。2022 年 2 月 12 日現在のバージョンは version 4.4.206 で、様々な生物から得られた 2,312,698 ペアの相互作用データが含まれている。上で述べたように、BioGRID はタンパク質間相互作用のデータとして設計されたものであるが、現在のデータベースにはタンパク質-RNA 相互作用のデータも含まれている。このページのトップの Current-Release をクリックすると、BioGRID の最新のダウンロードファイルのリストが表示されたページに移動できる。ここに表示されたリストをクリックすると、そのファイルをダウンロードできる。拡張子の zip は圧縮形式を表しており、コマンド gunzip で解凍できる。ここでは、２つのファイルについて説明しておく。



BioGRIDのダウンロードページ

最新リリースのリスト

BIOGRID-IDENTIFIERS-4.4.206.tab.zip を展開すると BIOGRID-IDENTIFIERS-4.4.206.tab.txt が得られる。このファイルには、BioGRID の ID コードと他のデータベースでの ID の対応関係が記述されている。このファイルに記述されている ID の変換を介して相互作用以外の情報を入手することができる。これについては、三章の GO エンリッチメント解析のところで改めて説明する。

BIOGRID-ALL-4.4.206.tab3.zip を展開すると、BIOGRID-ALL-4.4.206.tab3.txt が得られる。このファイルには相互作用のペアのデータが格納されている。上で述べたように 1 行に一つの相互作用が記述されている。 1 行目にこのファイルのヘッダー行があり、各列についてのタイトルが書かれている。
各項目はタブで区切られている。

#BioGRID Interaction ID Entrez Gene Interactor A Entrez Gene Interactor B BioGRID ID Interactor A BioGRID ID Interactor B Systematic Name Interactor A Systematic Name Interactor B Official Symbol Interactor A Official Symbol Interactor B Synonyms Interactor A Synonyms Interactor B Experimental System Experimental System Type Author Publication Source Organism ID Interactor A Organism ID Interactor B Throughput Score Modification Qualifications Tags Source Database SWISS-PROT Accessions Interactor A TREMBL Accessions Interactor A REFSEQ Accessions Interactor A SWISS-PROT Accessions Interactor B TREMBL Accessions Interactor B REFSEQ Accessions Interactor B Ontology Term IDs Ontology Term Names Ontology Term Categories Ontology Term Qualifier IDs Ontology Term Qualifier Names Ontology Term Types Organism Name Interactor A Organism Name Interactor B

今回、この講義で利用する項目や関係する項目を赤で示している。BioGRID ID Interactor A と B には、相互作用するタンパク質のペアのそれぞれに BioGRID から与えられた identifier を記されている。BioGRID ID Interactor A と B は、異なる実験により同じペアが同定された場合は、違う行に同じタンパク質ペアが記述されている場合がある。また、A と B の ID が交換されて B A となっている場合がある。統計を取る際には注意する必要がある。Official Symbol Interactor A と B は、相互作用ペア A と B のそれぞれの公式の gene symbol を表している。Experimental System は、上述の種々の実験のどれがその相互作用の同定に用いられたかを表しており、Experimental System type は Physical か Genetic の 2 種類のいずれかが書かれている。Organism Name Interactor A と B は、タンパク質 A、B が由来する生物の種名が書かれている。生物種が A、B それぞれに割り振られているのは、例えば病原菌やウイルス由来のタンパク質と、宿主由来のタンパク質が相互作用している場合に、異なる種名が A, B に割り当てられるためである。

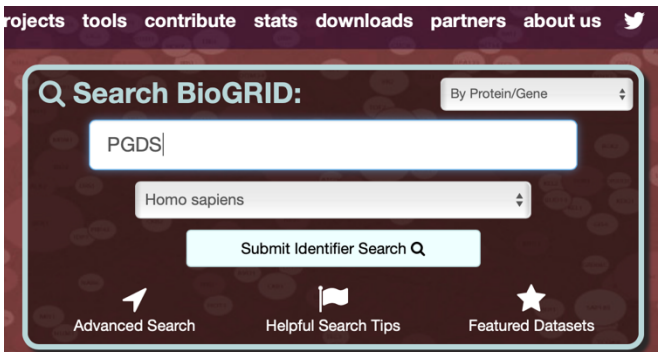
BioGRID については、次の引用文献を参照すること。

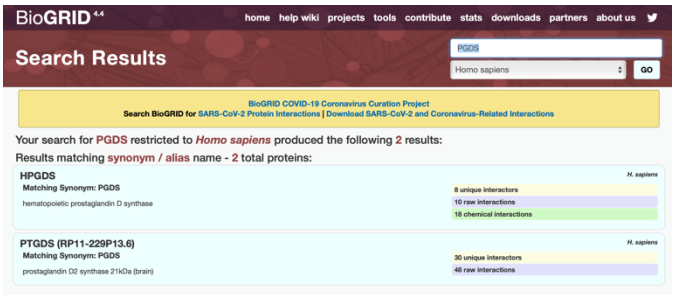
Strak *et al.* (2006) BioGRID: a general repository for interaction datasets. *Nucl. Acids Res.* **34**(Database issue):D535-9.

Oughtred *et al.* (2020) The BioGRID database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions. *Protein Sci.* **30**, 187-200.

このタンパク質間相互作用のデータ全てを取り扱うのは大変なので、今回はヒトのタンパク質間相互作用のうち、リポカリン型 prostaglandin D_2 合成酵素(以下 PGDS)を中心とした相互作用を取り出したデータを作成した。PGDS は、シクロオキシゲナーゼによって生成された prostaglandin H_2 を基質として prostaglandin D_2 を生成する酵素である。prostaglandin D_2 はヒトやマウスの脳内で生成される主要なプロスタグランジンであり、痛覚緩和、体温低下、睡眠誘発などの生理活性を持つ。PGDS は、細胞内の小胞体のルーメンサイドに局在して、はたらいている。PGDS は、リポカリンファミリーと呼ばれるタンパク質ファミリーに属している。リポカリンファミリーは、疎水性低分子の輸送に関わる非酵素タンパク質のファミリーであり、PGDS はそこから酵素として進化してきたと考えられている。また、PGDS は基質である prostaglandin H_2 が存在しない脳脊髄液中に分泌されており、輸送タンパク質としても機能していると考えられている。

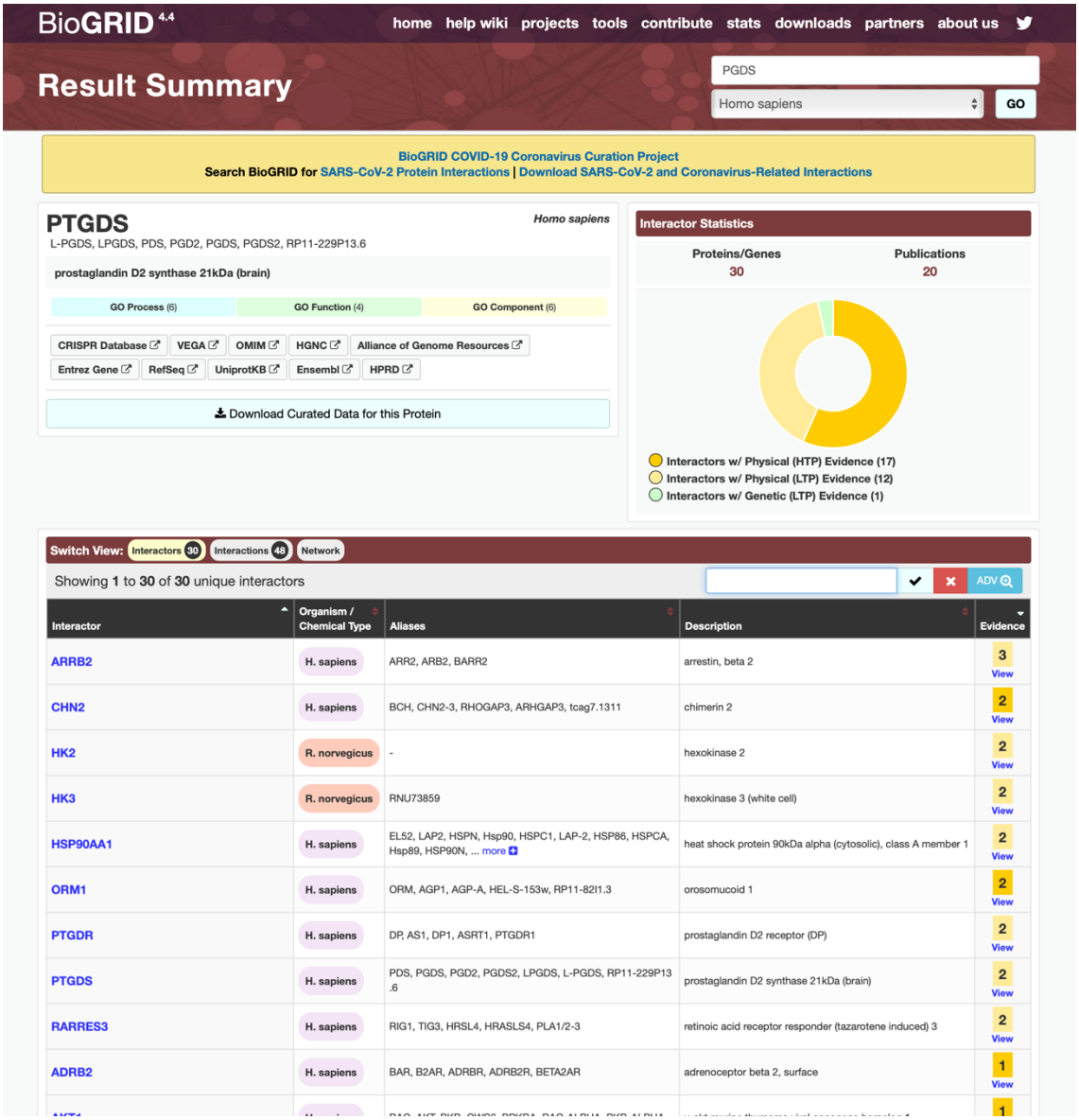
Toh *et al.* (1996) Glutathione-independent prostaglandin D synthase as a lead molecule for designing new functional proteins. *Protein Eng. 9*, 1067-1082.





BioGRID の HP の上部の検索機能で PGDS と相互作用するタンパク質を探索してみる。検索する生物種を Homo sapiens として検索すると 2 つ検索結果が出てくる。上段の HPGDS は hematopoietic prostaglandin D_2 synthase で、活性は上で述べたものと同じだが、glutathione S transferase ファミリーに属している、下段の PTGDS が今回調べようとしているリポカリン型 PGDS である。PTGDS をクリックすると、相互作用するタンパク質の一覧が表示される。

しかし、一覧が得られただけでは、それがどのようなネットワークを構築し、どのように相互作用しているのかはわからない。ダウンロードしたファイルから PGDS に関連する相互作用情報を抜き出したファイルを準備したので、それを使ってどのようにネットワーク解析を行うのかを説明していく。



ファイル humanPGDSPPI2.txt は、BioGROD ID の**辺リスト**の形で、PGDS とそれと相互作用しているタンパク質が記されている。このファイルに含まれるタンパク質の数は 23y 個、相互作用の数は 181 ペアである。以下、humanPGDSPPI2.txt の先頭の 5 行を記す。PGDS の BioGRID ID は 111702 である。

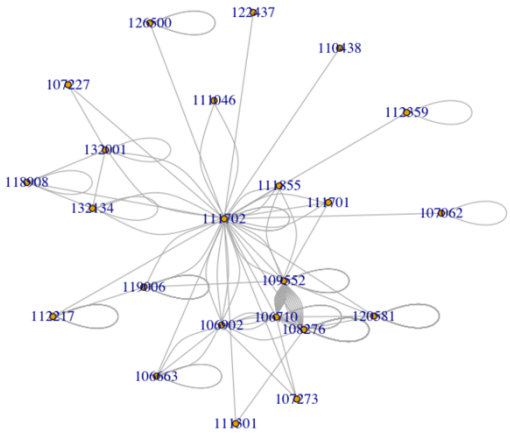
```
108276  108276
109552  106710
109552  109552
106663  106663
106710  109552
```

2-2. 可視化

可視化は 2-1-2 でも試したが、“humanPGDSPPI2.txt”を使って調べよう。R を立ち上げ、setwd 関数で“humanPGDSPPI2.txt”のあるディレクトリに移動した後に、以下の処理を行う。igraph のオブジェクトとしてネットワークを作成し、それを plot を使って可視化する。

2-2-1. ループの除去

```
library(igraph)
data <- read.table("humanPGDSPPI2.txt", header=F)
g <- graph.data.frame(data, directed=F)
plot(g, vertex.size=3)
```

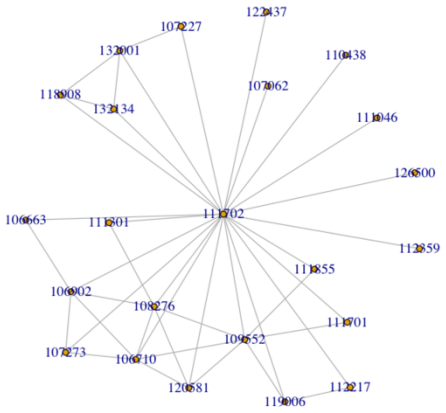


read.table は、辺リストをデータフレームとして読み込んでいる。しかし、igraph パッケージは、通常のデータフレームは利用できないため、graph.data.frame で igraph で利用できる形に変換している。Plot 中の vertex.size は、ノードのサイズを指定している。

自身との相互作用、また複数の同じ相互作用の記述によりループが形成されているので、このループを除去してみよう。

```
plot(simplify(g), vertex.size=3)
```

simplify によって、ループが消えているのがわかる。しかし、ネットワークのレイアウトが変わってしまっている。



2-2-2. レイアウトの記憶

igraph では描画するたびに、ネットワークのレイアウトが変わってしまう。同じレイアウトでネットワークを作成する方法は二つ紹介しておく。

最初の方法は、layout.auto 関数を使う方法である。

```
coords= layout.auto(g)
```

によって、レイアウトを記憶しておけば、何回以下の操作をやっても同じ構成で描画される。ただし、試行錯誤で適切なレイアウトを見つける必要がある。

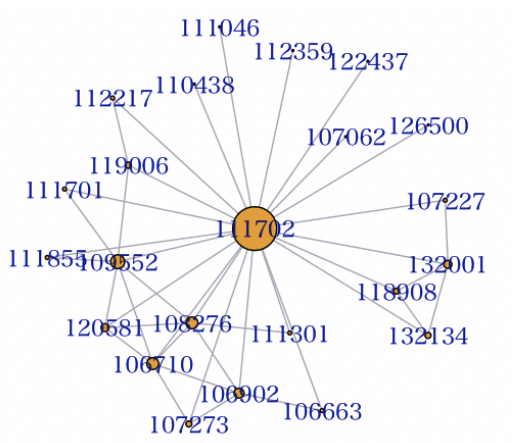
```
plot(simplify(g), vertex.size=3, layout=coords)
```

もう一つは set.seed 関数を使う方法である。ネットワークを描画する前に、
set.seed(12345) を実行し、その後に plot(simplify(g), vertex.size=3)で描画する。12345 には意味はない。色々な数字を試してみて、描画として都合の良いレイアウトになる数字を選択する必要がある。

```
set.seed(12345)
plot(simplify(g), vertex.size=3)
set.seed(12345)
plot(simplify(g), vertex.size=3)
```

何度描画しても、同じレイアウトになるのがわかる。

2-2-3. 節のサイズ



節のサイズをベクトルとして与えて変更してみる。次の章で説明する次数（節が持つ辺の数＝節がいくつの他の節とリンクしているか）に比例するように節の大きさを変更してみる。
各節の次数は igraph パッケージの degree 関数で計算できる。次数そのままだとノードのサイズが大きく過ぎるものがあるので、適当な数字を乗じて大きさを調節する。ここでは 0.3 をかけている。

```
dg <- degree(simplify(g))
print(dg)
```

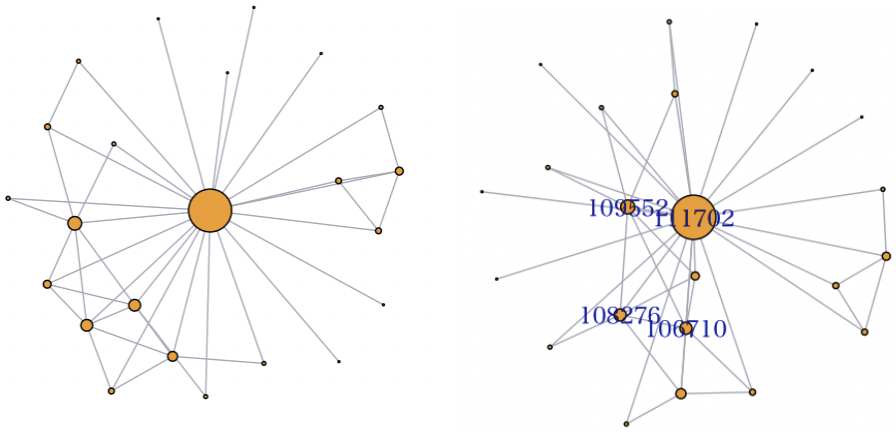
```
108276 109552 106663 106710 107062 111855 107227 120581 106902 119006 112359 112217 118908
     6      7      2      6      1      2      2      4      5      3      1      2      3
110438 111702 107273 111701 111046 126500 132001 132134 111301 122437
     1     22      3      2      1      1      4      3      2      1
```

```
plot(simplify(g), vertex.size=dg)
```

2-2-4. 節のラベルの非表示、変更

節のラベルとして BioGRID ID が使われている。plot 関数の中で、vertex.label=NA とすると、ラベルを非表示にできる。

```
plot(simplify(g), vertex.size=degree(simplify(g)), vertex.label=NA)
```



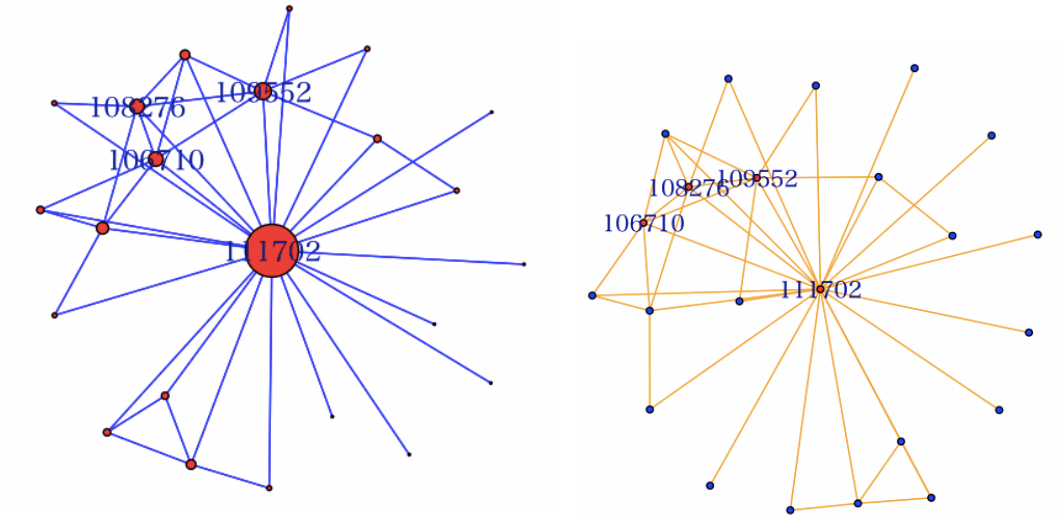
ラベルをベクトルで与えることで、特定の節のラベルのみ表示させることができる。
ここでは 5 より大きな字数を持つ節のラベルのみ表示させている。

```
lab <- c()
for (i in 1:length(dg)) {
        if (dg[i] > 5) {
                lab <- c(lab, names(dg[i]))
        } else {
                lab <- c(lab, "")
        }
}
plot(simplify(g), vertex.size=degree(simplify(g)), vertex.label=lab)
```

2-2-5. 節のおよび辺の色の変更

plot 関数で、　edge.color, vetex.color を指定することで色を変更できる。ここでは、節を赤に、辺を青に変更してみる。

```
plot(simplify(g), vertex.size=degree(simplify(g)), vertex.label=lab,
vertex.color="red", edge.color="blue")
```



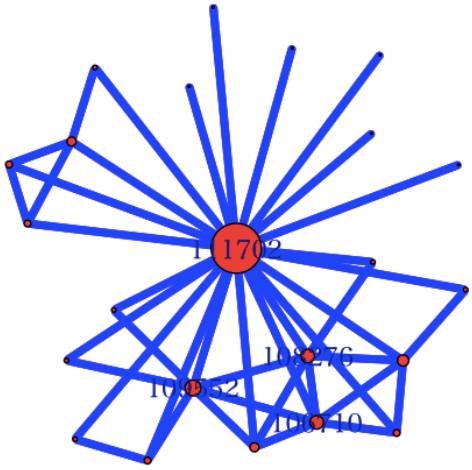
ベクトルを使って、10 より大きな次数を持つ節を赤、それ以外の節を青に設定できる。また、edge.color の設定で、辺をオレンジに変更することができる。

```
cols <- c()
for (i in 1:length(dg)) {
        if (dg[i] > 5) {
                cols <- c(cols, "red")
        } else {
                cols <- c(cols, "blue")
        }
}
plot(simplify(g), vertex.size=3, vertex.label=lab, vertex.color=cols,
 edge.color="orange")
```

2-2-6. 辺の幅の変更

辺の幅は、edge.width の設定を変えることで変更できる。

```
plot(simplify(g), vertex.size=degree(simplify(g)), vertex.label=lab,
vertex.color="red", edge.color="blue", edge.width=5)
```



ここでは、ネットワークの可視化に必要最低限の知識を紹介した。igraph によるネットワークの可視化については、いくつかのウェブサイトでの詳しい解説やネットワーク解析についてのテキストがある。さらに学びたい人はそれらを参照して勉強してもらいたい。

http://www.nemotos.net/igraph-tutorial/NetSciX_2016_Workshop_ja.html

https://sites.google.com/site/kztakemoto/r-seminar-on-igraph---supplementary-information

鈴井　努（2017）R で学データサイエンス 8　ネットワーク分析 第 2 版　　共立出版

2-3.　中心性解析

中心性とは、ネットワークを構成する各節に対して求められる指標であり、その節が、そのネットワークの中で、どの程度中心的な位置を占めているかを表している。中心性が高い節は、そのネットワークの中で重要な役割を担うことが多いことが知られている。中心性にはいくつか異なる定義があり、そのいくつかは igraph パッケージの中で使用できる。

2-3-1. 次数中心性（degree centrality）

2-2-3 で説明した次数に基づく中心性である。可能な最大次数は（節の総数 − 1）であることから、節の次数を（節の総数 − 1）で割ったものが、次数中心性として用いられる。

節 i の次数中心性 $= \frac{\text{節 } i \text{ の次数}}{\text{節の総数} - 1}$

```
dg <- degree(simplify(g))
dc <- dg/(length(dg)-1)
print(dc)
```

```
    108276     109552     106663     106710     107062     111855     107227     120581
0.27272727 0.31818182 0.09090909 0.27272727 0.04545455 0.09090909 0.09090909 0.18181818
    106902     119006     112359     112217     118908     110438     111702     107273
0.22727273 0.13636364 0.04545455 0.09090909 0.13636364 0.04545455 1.00000000 0.13636364
    111701     111046     126500     132001     132134     111301    122437
0.09090909 0.04545455 0.04545455 0.18181818 0.13636364 0.09090909 0.04545455
```

固有ベクトル中心性、PageRank 中心性、パワー中心性など、リンクしている節の中心性を考慮する形で次数中心性を修飾した中心性がいくつか考案されている。上記の 3 つは、igraph パッケージの中に関数として含まれている。

次数に関連する話題として、ネットワーク解析を行う上で重要な概念を説明しておく。

ハブとは、非常に多くの相互作用相手を持つ節のことを指す。生体ネットワークは、少数のハブと、多くの次数の低い節より構成されている。より正確には、次数の対数 x と、その次数を持つ節の出現頻度の対数 y に、$y \propto -k \cdot x$ の関係が成立している。このような性質を持つネットワークを**スケールフリーネットワーク**と呼ぶ。スケールフリー性は、生体ネットワークばかりでなく、インターネットなどでも見られる性質で、ランダムな節の破壊に対して頑健であることが知られている。

今回のデータを使って、スケールフリー性の確認がどのように行われるかを見ていこう。

```
dg <- degree(simplify(g))
hist(dg, breaks=20)
```

これにより次数の出現度数のヒストグラムが得られる。
このヒストグラムん情報を変数 dgh に格納する。

```
dgh <- hist(dg, breaks=20)
```

dgh の中にどのようなデータが含まれているかを str で確認する。

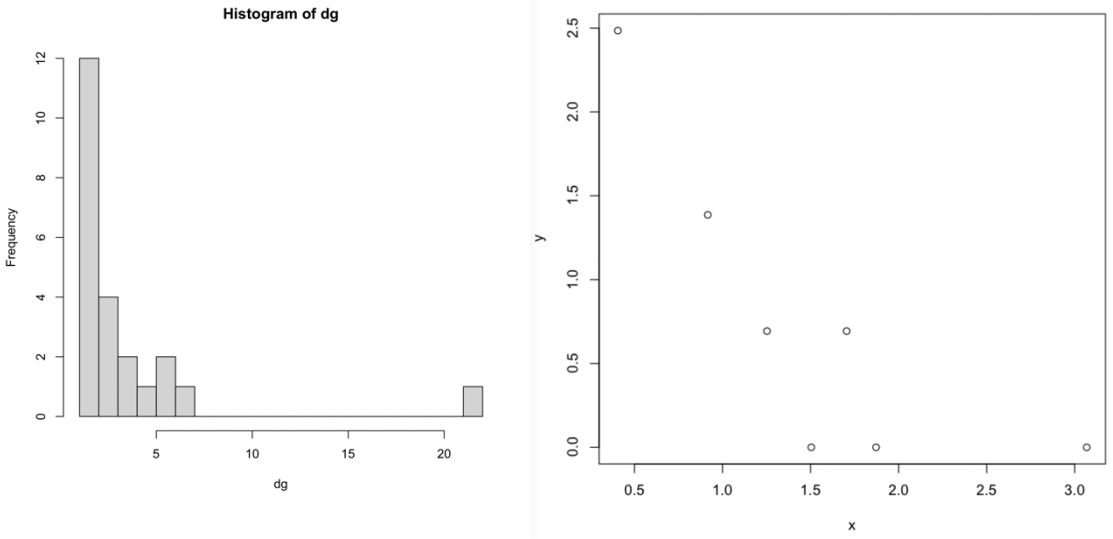
```
str(dgh)
List of 6
 $ breaks  : int [1:22] 1 2 3 4 5 6 7 8 9 10 ...
 $ counts  : int [1:21] 12 4 2 1 2 1 0 0 0 0 ...
 $ density : num [1:21] 0.5217 0.1739 0.087 0.0435 0.087 ...
 $ mids    : num [1:21] 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5 10.5 ...
 $ xname   : chr "dg"
 $ equidist: logi TRUE
```

```
 - attr(*, "class")= chr "histogram"
```

これにより、counts が各次数の出現度数、mids がビンの中央値を表すので、それぞれの対数をとる。

```
x <- log(dgh$mids)
y <- log(dgh$counts)
plot(x, y)
```

次数が高いところで少しばらけるが、プロットが直線的であるように見える。このネットワークは、サイズは小さいが、スケールフリー性を持っていると考えられる。



2-3-2. 近接中心性（closeness centrality）

ネットワーク中の 2 つの節の最短距離を考える。ここで、距離とは 2 つの節を結ぶ最短経路の中の辺の数を表す。ある節の**近接中心性**は、その節から他の節への最短距離の平均値の逆数と定義される。その節がネットワークの中心に位置するほど、他の節への平均最短距離は小さくなると考えられるので、その逆数をとって、中心的であるほど大きな値を取るようにしてある。

$$\text{節 } i \text{ の近接中心性} = \frac{\text{節の総数} - 1}{\text{節 } i \text{ からの他の節への最短距離の合計}}$$

igraph パッケージでは節間の最短距離は `distances` 関数で求めることができる。

```
distances(g)
```

	108276	109552	106663	106710	107062	111855	107227	120581	106902	119006	112359	112217	118908	110438	111702	107273	111701	111046	126500	132001	132134	111301	122437
108276	0	1	2	1	2	2	2	1	1	2	2	2	2	2	1	2	2	2	2	2	2	1	2
109552	1	0	2	1	2	1	2	1	2	1	2	2	2	2	1	2	1	2	2	2	2	2	2
106663	2	2	0	2	2	2	2	2	1	2	2	2	2	2	1	2	2	2	2	2	2	2	2
106710	1	1	2	0	2	2	2	1	1	2	2	2	2	2	1	1	2	2	2	2	2	2	2
107062	2	2	2	2	0	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2
111855	2	1	2	2	2	0	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2
107227	2	2	2	2	2	2	0	2	2	2	2	2	2	2	1	2	2	2	2	1	2	2	2
120581	1	1	2	1	2	2	2	0	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2
106902	1	2	1	1	2	2	2	2	0	2	2	2	2	2	1	1	2	2	2	2	2	2	2
119006	2	1	2	2	2	2	2	2	2	0	2	1	2	2	1	2	2	2	2	2	2	2	2
112359	2	2	2	2	2	2	2	2	2	2	0	2	2	2	1	2	2	2	2	2	2	2	2
112217	2	2	2	2	2	2	2	2	2	1	2	0	2	2	1	2	2	2	2	2	2	2	2
118908	2	2	2	2	2	2	2	2	2	2	2	2	0	2	1	2	2	2	2	1	1	2	2
110438	2	2	2	2	2	2	2	2	2	2	2	2	2	0	1	2	2	2	2	2	2	2	2
111702	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
107273	2	2	2	1	2	2	2	2	1	2	2	2	2	2	1	0	2	2	2	2	2	2	2
111701	2	1	2	2	2	2	2	2	2	2	2	2	2	2	1	2	0	2	2	2	2	2	2
111046	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	0	2	2	2	2	2
126500	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	0	2	2	2	2
132001	2	2	2	2	2	2	1	2	2	2	2	2	1	2	1	2	2	2	2	0	1	2	2
132134	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1	2	2	2	2	1	0	2	2
111301	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	0	2
122437	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	0

節間の距離行列が求まる。そこで、各節の近接中心性は次のように求めることができる。

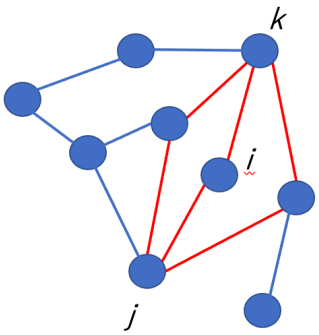
```
1/apply(distances(g),1, sum)/(length(distances(g)[,1])-1)
```

```
     108276      109552      106663      106710      107062      111855      107227      120581      106902      119006      112359
0.001196172 0.001228501 0.001082251 0.001196172 0.001057082 0.001082251 0.001082251 0.001136364 0.001165501 0.001108647 0.001057082
     112217      118908      110438      111702      107273      111701      111046      126500      132001      132134      111301
0.001082251 0.001108647 0.001057082 0.002066116 0.001108647 0.001082251 0.001057082 0.001057082 0.001136364 0.001108647 0.001082251
     122437
0.001057082
```

ここで apply を使って各行についての和をとっている。apply で平均をとらなかったのは、要素数ではなく、(要素数-1)で割りたかったからである。

類似する中心性として、**離心中心性**がある。離心中心性は、ある節から他の節への距離の最大値の距離として定義される。

2-3-3. 媒介中心性（betweenness centrality）



ある節 i の媒介中心性を考えるために、i と異なる 2 つの節 j, k を考える。j と k の最短経路の数を $g_{j,k}$ とする。また、その中で i を通る経路の数を $g_{j,k}(i)$ とする。

左の例では、$g_{j,k}= 3$, $g_{j,k}(i)=1$ となる。これを i を含まない、全ての節のペアについて和をとったものが節 i の媒介中心性となる。

$$\sum_{\{j,\ k\ |\ j,\ k\neq i\}} \frac{g_{j,k}(i)}{g_{j,k}}$$

R には媒介中心性を求めるための関数 betweenness がある

```
betweenness(g)
```

```
    108276     109552     106663     106710     107062     111855     107227     120581     106902     119006     112359
 2.7699554  6.6549152  0.0000000  2.8855072  0.0000000  0.0000000  0.0000000  0.0000000  2.8635531  0.2500000  0.0000000
    112217     118908     110438     111702     107273     111701     111046     126500     132001     132134     111301
 0.0000000  0.0000000  0.0000000 194.7427357  0.0000000  0.0000000  0.0000000  0.0000000  0.8333333  0.0000000  0.0000000
    122437
 0.0000000
```

３種類の中心性を説明したが、これらの関係を見てみよう。

```
dg <- degree(simplify(g))
dc <- dg/(length(dg)-1)
dc <- cbind(dc, 1/apply(distances(g),1, sum)/(length(distances(g)[,1])-1))
dc <- cbind(dc, betweenness(g))
colnames(dc) <- c("degree", "closeness", "betweenness")
cor(dc)
```

```
             degree closeness betweenness
degree      1.0000000 0.9847955   0.9269797
closeness   0.9847955 1.0000000   0.9780240
betweenness 0.9269797 0.9780240   1.0000000
```

今回のネットワークでは、３種の中心性は、いずれのペアにおいても高い相関係数を示しており、類似することがわかる。次に各中心性を降順に並べ、上位 5 個を取り出してみる。

位数中心性の場合：

```
dg <- degree(simplify(g))
dc <- dg/(length(dg)-1)
dc[order(dc, decreasing=T)][1:5]
[1] 194.742736   6.654915   2.885507   2.863553   2.769955
names(dc)[order(dc, decreasing=T)][1:5]
[1] "111702" "109552" "108276" "106710" "106902"
```

近接中心性の場合：

```
cc <- 1/apply(distances(g),1, sum)/(length(distances(g)[,1])-1)
cc[order(cc, decreasing=T)][1:5]
    111702      109552      108276      106710      106902
0.002066116 0.001228501 0.001196172 0.001196172 0.001165501
```

媒介中心性の場合：

```
bc <- betweenness(g)
bc[order(bc, decreasing=T)][1:5]
   111702     109552     106710     106902     108276
194.742736   6.654915   2.885507   2.863553   2.769955
```

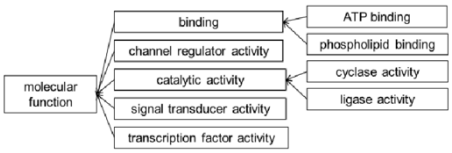
上位 5 個は、位数中心性と近接中心性は共通しているが、媒介中心性はやや異なっている。また、今回 PGDS と相互作用するタンパク質でネットワークを作成したので、いずれの中心性においても PGDS (BioGRID ID = 111702)が最大の中心性を持っている。

3. GO エンリッチメント解析

3-1. GO (Gene Ontology)

オントロジー(ontology)は、分野によって異なる意味で使用されるが、情報科学の分野では「知識」を、対象としている世界における知識を、その知識を構成する「概念」を明示的に表現するとともに、その「概念の特性」や「概念間の関係」として表現したものである。抽象的な表現でわかりにくい表現だが、以下の遺伝子オントロジー(gene ontology)の説明で具体的に説明する。

遺伝子オントロジーでは、**遺伝子の機能**の注釈を、分**子機能(molecular function)、細胞要素（cellular component）、生物学的プロセス(biological process)**の３つに分類した上で、それらに関連する用語を収集し、それの用語の統一、分類し、またそれらの間の相互関係の情報を提供してい



る。下図は高井(2017)からとったGO内の分子機能に属している概念の関係性の例である。２つの概念は、"is-a"あるいは"part-of"などで関係づけられる。例えば"cyclase activity" + "is-a" + "catalytic activity" などのように、関係付けを意味する矢印の部分に"is-a"などのその関係性の意味が割り当てられる。また、この２つの概念をつなぐ経路の辺の数を使って概念間の距離とすることで、概念の近さを表現できる。例えば、phopholipid bindingとcyclase activityをつなぐ最短経路内の辺の数は4であるが、phspholipid bindingとbindingの距離は1であり、後者の方が概念として近いことがわかる。これらの概念の直感的な近さと一致している。相対的に包括的な概念が「親」となり、より細かい概念が「子」となっている。この概念のことを、GO termと呼ぶ。GO termには、番号(GO ID)が割り振られている。

桝屋、溝口 (2014) 遺伝学オントロジー *人工知能学会論文誌* **26,** 311-327.
高井 (2017) 個別化医療・個別化予防を推進するオントロジー *CICSJ Bulletin* **35,** 174-179.

3-2. エンリッチメント解析と超幾何分布

遺伝子機能がGOと整理されているおかげ、コンピュータによって遺伝子機能の解析を行うことができる。その代表的な解析方法が**エンリッチメント解析**と呼ばれるものである。1-2で、例として正常細胞とがん細胞で発現量に差がある遺伝子の選択を挙げた。今、多重比較の補正のもとで、30個の遺伝子ががん細胞で有意に発現量が増加していたとしよう。これらの遺伝子の機能に共通性が見られれば、その機能ががんと関わるのではないかと推測することができる。この時に使われるのはGO termである。30個の遺伝子が持つGO termを考える。今、この30個の遺伝子の内、GO term の一つとしてtyrosine kinase signaling pathwayが25個の遺伝子から得られたとする。がん細胞で発現量が増加した遺伝子のセットにおけるこのGO termの観察数、ヒトの遺伝子セット全体の中でtyrosine kinase pathwayというGO termが出現する割合から考えて、有意に多いと言えるだろうか？これに答えるための解析がエンリッチメント解析である。

今、ヒトの遺伝子の全数をNとする。この中で、tyrosine kinase pathwayというGO termを持つものがM個であったとする。N個の遺伝子から、何らかの基準（上の例ではがん細胞で発現量が有意に増加している）で選択された遺伝子の数をn個、その中でtyrosine kinase pathwayというGO termを持つものがm個であったとする。この**確率質量**は次のように超幾何分布を使って計算できる。

$$P(n,m,N,M)=\frac{\binom{M}{m}\binom{N-M}{n-m}}{\binom{N}{n}}$$

分子の$\binom{M}{m}$は、GO termであるtyrosine kinase pathwayを持つM個の遺伝子から、m個がサンプルされる組み合わせの数である。分子の$\binom{N-M}{n-m}$は、tyorosine kinase pathwayというGO termを

持たない $N-M$ 個から、その GO term を持たない $n-m$ 個がサンプルされる組み合わせの数である。分母の $\binom{N}{n}$ は、全遺伝子 N から n 個の遺伝子を取り出す組み合わせの数である。上の例では、$n = 30$, $m = 25$ である。この帰無仮説は、ヒトの全遺伝子 N 個中、当該の GO term を持つ遺伝子が M 個見出されるという比率に従って、n 個のサンプル中 m 個の遺伝子に該当の GO term が見出されるという同義置換率とおである。これから p-値を次のように計算される。

$$p\text{-値} = \sum_{i=m}^{\min(n,M)} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}}$$

これが設定された有意水準よりも小さければ、帰無仮説が棄却され、その GO term は有意にエンリッチされていると判断される。しかし、ここで一章で述べた多重比較の問題が出てくる。選択された n 個の遺伝子の持つ GO term が k 種類あったすると、k 回上記の検定を繰り返すことになるのでそのための多重比較の補正が必要となる。

法隆、林 (2015) エクソンレベルでのエンリッチメント解析におけるエクソン数に起因するバイアスの補正　*計量生物学* **36,** 63–84.

3-3.　R を使ったエンリッチメント解析

GO エンリッチメント解析を行えるサイトやツールがいくつか存在するが、今回は R で利用できる gprofiler2 を用いて解析してみる。今回は、先のネットワーク解析に用いた PGDS と相互作用するタンパク質の集合について、どのような GO term がエンリッチされているかを調べる。ネットワーク解析に用いた humanPGDSPPI2.txt は辺リストの形であり、また BioGRID ID で記述されている。BioGRID ID は gprofiler2 には認識されないので、BIOGRID-IDENTIFIERS-4.4.206.tab.txt の異なるデータベースの間の ID の対応関係を見て、gene symbol のリストの形に変換したファイルを"gsPGDS.list"を準備した。これを読み込んで解析を実施する。

```
gslist <- read.table("gsPGDS.list",header=F)
gslist$V1
```

```
 [1] "ADRB2"    "AKT1"     "ARRB2"    "BCAT2"    "CACNA1A"  "CAPN1"    "EGFR"     "HSP90AA1" "MLH1"     "ORM1"
[11] "PIGH"     "PTGDR"    "PTGDS"    "RARRES3"  "ATXN1"    "SHBG"     "CARD10"   "UBQLN2"   "FBXW7"    "ZNF747"
[21] "RNF183"   "CYSRT1"   "KRTAP12-3"
```

次にパッケージ gprofiler2 を読み込む。

```
library(gprofiler2)
```

もし、ここでエラーが出るようであれば、gprofiler2 がまだインストールされていないので、CRAN からインストールしておく必要がある。Gprofiler2 の関数 gost を使ってエンリッチメント解析を行う。gost の使い方については、次の URL の HP に記述されている。

https://rdrr.io/cran/gprofiler2/man/gost.html

```
data <- gost(as.vector(gslist$V1), organism="hsapiens",
correction_method="fdr")
```

第一引数は query で、エンリッチメントを確認したい遺伝子のセットである。文字列のベクトルとして与えてある。今回は、official gene symbol で遺伝子を指定している。Official gene symbol とは、NCBI で定義されている遺伝子名であり、現在も更新されていて変更されることがある。

```
as.vector(gslist$V1)
```

```
 [1] "ADRB2"    "AKT1"     "ARRB2"    "BCAT2"    "CACNA1A"  "CAPN1"    "EGFR"     "HSP90AA1" "MLH1"     "ORM1"     "PIGH"
"PTGDR"    "PTGDS"    "RARRES3"  "ATXN1"    "SHBG"     "CARD10"   "UBQLN2"   "FBXW7"    "ZNF747"   "RNF183"   "CYSRT1"
"KRTAP12-3"
```

第二引数では生物種を指定しており、今回は Homo sapiens が指定されている。
第三引数は、多重比較の補正の方法の指定である。今回、FDR を指定しているが、これにより Benjamini-Hochberg 法で補正が行われる。gost では補正法として、そのほかに Bonferroni 法、sSCS 法（gost の開発者のオリジナルの方法）が選択できる。

計算結果は変数 data に格納されているが、多くの情報が含まれているので、p_value(補正済み)、term ID, term の３つのみを取り出し、p_value で昇順に並べ直した。ここでは、p_value が 0.001 より小さいもの 7 つのみを表示している。Term ID としてここでは GO term だけを示しているが、gprofiler２は、他のデータベースの注釈 term についてもエンリッチ解析が行われている。term name を見ながら、このタンパク質のセットにエンリッチされている用語を見てみると、signaling pathway 関係の term が 9 つ中、6 つを占めており、このネットワークにはシグナリング関係の機能がエンリッチされていることがわかる。NO はシグナル分子として働く。G protein-coupled receptor は、シグナルで変換に関わる受容体である。リン酸化による翻訳後修飾は、細胞内で変換されたシグナルの代表的なものである。Signaling pathway というそのままの term が 2 つ含まれている。prostaglandin D_2 は, 細胞外に分泌され、G protein-coupled receptor に結合して、近傍の細胞にシグナルを伝える（パラクライン）。このため、相互作用しているタンパク質もシグナル伝達に関与するものが多いのかもしれない。

```
data$result[,c("p_value", "term_id",
"term_name")][order(data$result["p_value"]),]
```

```
      p_value         term_id                                                  term_name
668 0.0000179141       GO:0030235                                nitric-oxide synthase regulator activity
```

75	0.0006350478	GO:0022401	negative adaptation of signaling pathway
76	0.0006350478	GO:0023058	adaptation of signaling pathway
77	0.0006350478	GO:0002029	desensitization of G protein-coupled receptor signaling pathway
78	0.0006350478	GO:0002032	desensitization of G protein-coupled receptor signaling pathway by arrestin
79	0.0006350478	GO:0051054	positive regulation of DNA metabolic process
80	0.0009637258	GO:0009894	regulation of catabolic process
81	0.0009637258	GO:0031331	positive regulation of cellular catabolic process
82	0.0009637258	GO:0033138	positive regulation of peptidyl-serine phosphorylation

4．終わりに

第一章では、多重比較の説明をした。これは、ゲノムワイドな解析ばかりでなく、通常の実験でも一つのセッションの中で検定を繰り返す操作が含まれる場合には要求される操作であるので、ぜひ理解してもらいたい。また、今回、タンパク質相互作用を題材としてネットワーク解析とＧＯエンリッチメント解析について解説したが、他の生物学的問題にも応用可能であるので、原理およびツールの使い方を理解してもらいたい。