このとき、遺伝子Aの頻度の分布の
累積密度関数は
不完全ベータ関数となる

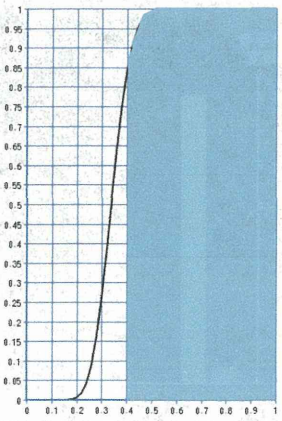
$$I_x(1+r,1+n-r)=\frac{\int_0^x t^r(1-t)^{n-r}dt}{\int_0^1 t^r(1-t)^{n-r}dt}$$

右に、一般人口60人を調べて、
20人が遺伝子変異Aを有する場合の
不完全ベータ関数を例示する。

上記の「一般人口でn人を調べ、r人が
遺伝子Aを有する」場合、一般人口中の
頻度が例えば0.4以上である確率Pは

$$P=1-I_{0.4}(1+r,1+n-r)$$

で得られる。



比較対象その1：ESP (Exome Sequencing Project) から
6500人分のEA (European American) とAA (African American)の
SNPサマリをダウンロードして比較

NHLBI Exome Sequencing Project (ESP)
Exome Variant Server

Home | Data Browser | Data Usage and Release | How to Use | What's New | Contact and FAQ | Downloads

The following downloadable files contain SNPs and coverage data for the ESP 6500 exomes (chromosomes 1-22, and X).

ESP6500.snps.txt.tar.gz (SNPs and annotations in space-delimited text format)
ESP6500.snps.vcf.tar.gz (SNPs and annotations in VCF format - same data as ESP6500.snps.txt.gz)
ESP6500.coverage.all_sites.txt.tar.gz (sequencing coverage data for every site in our targets)
ESP6500.coverage.seq_blocks.txt.tar.gz (sequencing coverage summary data for every sequencing block in our targets)

Chromosome	Position	Ref	Alt	Info		ESP EA			ESP AA		
1	14618	T	C	(nexus)SNP	rs79134172	WASHSP	4300	0	0	2203	0	0
1	14653	C	T	(nexus)SNP	rs62635297	WASHSP	4300	0	0	2203	0	0
1	14677	G	A	(nexus)SNP	rs112391608	WASHSP	4300	0	0	2203	0	0
1	14698	A	C	(nexus)SNP		WASHSP	4300	0	0	2203	0	56
1	14703	TC		(nexus)DEL		WASHSP	4300	0	0	2203	0	56
1	14706	C	T	(nexus)SNP		WASHSP	4300	0	0	2203	0	56
1	14975	CG	CA	(nexus)MNP		WASHSP	4300	0	0	2203	0	56
1	14975	C	T	(nexus)SNP	rs11264571;rs11803341	WASHSP	4300	0	0	0	2203	0
1	14976	G	A	(nexus)SNP	rs71252251;rs77117975	WASHSP	4300	0	0	0	2203	0
1	15829	G	A	(nexus)SNP		WASHSP	4300	0	0	2203	0	56
1	16957	G	T	(nexus)SNP		WASHSP	4300	0	0	2203	0	56
1	69322	T	A	SNV	.	OR4F5	4300	0	0	2203	0	0	56	0
1	69428	T	G	SNV	rs140739101	OR4F5	3203	129	92	1098	12	1	55

Pa：健常日本人の遺伝子異常保有頻度が3%以上である確率
Pb：疾患群の遺伝子異常保有頻度が、5%以上である確率
Pc：健常日本人の遺伝子異常保有頻度が、欧米人（EA、AA）の2倍以上である確率
Pd：健常日本人の遺伝子異常保有頻度が、中国人（CHB）の2倍以上である確率

疾患群サンプル、健常日本人サンプル（1000人ゲノム）、
健常中国人サンプル（1000人ゲノム）は、サンプル数が少ないので
既述のベータ分布などを用いて、頻度を確率学的に算出

欧米人(EA,AA)はサンプル数が多いので、絶対値で算出する。

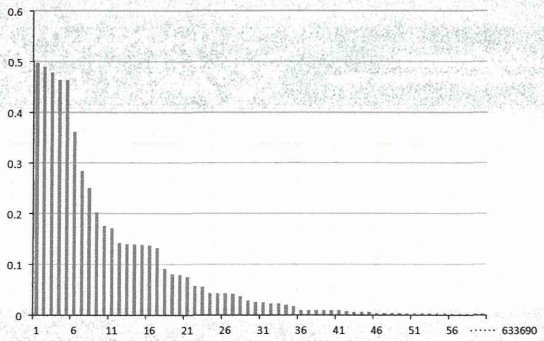
$$P = Pa \times Pb \times Pc \times Pd$$

を尤度として、このPの値が大きい方から順に検討していく

遺伝形式は、優性タイプが有力であり、
hg19のreferenceではないalternativeのSNP、Indelが有力である。
（reference側が疾患原因SNPということもあり得るが）

P	sumP	proportion	sum proportion chr	reference	alternative	type	rs#	gene
0.496970352	4.71489148	0.105415641	0.105415641	22 A	AC	INDEL	rs74910017	RIBC2
0.487101359		0.103322264	0.208737904	4 TTTTG	T	INDEL	rs140019361	DCHS2
0.469316679		0.099549839	0.308287743	12 T	G	SNV	.	FAM186A
0.386942964		0.08207701	0.390364753	11 TTGCTGCTGCT	T	INDEL	rs141671766	MAML2
0.342793265		0.072712127	0.46307688	11 C	CTGCCCTACT	INDEL	rs11268490	DNHD1
0.179907724		0.038161407	0.501238287	12 T	G	SNV	rs12297653	FAM186A
0.176200242		0.037374989	0.538613276	7 AT	A	INDEL	rs148691355	VWDE
0.173640818		0.036832093	0.575445369	8 CCTG	C	INDEL	rs5881390	MAP3K4
0.171714706		0.036423532	0.611868901	12 A	G	SNV	rs10876022	FAM186A
0.140107036		0.029719022	0.641587924	7 T	TC	INDEL	rs34061445	KCP
0.139262869		0.02953996	0.671127884	12 T	A	SNV	.	FAM186A
0.13917386		0.02952108	0.700648964	18 A	ATCT	INDEL	rs34960489	GREB1L
0.138096306		0.029292513	0.729941477	11 CA	C	INDEL	.	MUC2
0.132340942		0.028071705	0.758013182	19 GGCTCCTTCA	G	INDEL	rs67168196	FAM71E2
0.129976442		0.027570156	0.785583338	11 T	TAG	INDEL	rs113897236	RP11-397M16.4
0.123161477		0.026124589	0.811707927	16 G	C	SNV	rs7199961	ZNF469
0.110918341		0.023527617	0.835235544	12 C	A	SNV	rs11114486	PTPRQ
0.060285191		0.012787487	0.848023031	3 T	C	SNV	rs9833423	C3orf77
0.060152559		0.012759354	0.860782385	12 T	A	SNV	.	GXYLT1
0.059194087		0.012556046	0.873338431	5 TTCA	T	INDEL	rs10583854	ANKRD31
0.048069268		0.01020053	0.883538961	7 G	A	SNV	rs56353590	MUC12
0.040175338		0.008521854	0.892060815	12 C	CGCCGCTGC	INDEL	rs61519723	NOOR2
0.029849425		0.006331557	0.898392372	7 T	C	SNV	rs1059002	VWDE
0.029749879		0.006310442	0.904702814	8 G	A	SNV	.	RP1L1
0.029410492		0.006238452	0.910941267	12 G	A	SNV	rs34782671	FAM186A
0.029134178		0.006179841	0.917121108	8 T	A	SNV	rs13253053	RP1L1
0.029097349		0.00617203	0.923293138	12 T	G	SNV	.	FAM186A
0.025399806		0.005387719	0.928680857	21 GA	G	INDEL	rs35359062	AP000244.1
0.024523062		0.005201747	0.933882605	12 TTGC	T	INDEL	rs3835029	ZNF384
0.02442221		0.005180035	0.93906298	7 G	C	SNV	rs56357244	MUC12
0.023163919		0.004913451	0.94397641	21 A	G	SNV	rs9992755	AP000244.1

疾患原因遺伝子の発症形式が優性で、かつAlternativeが原因とした場合の、尤度



上位30程度で、全体の95%を占める
・・・この中に95%の確率で原因遺伝子変異が存在する

今後の課題
・・・mapping（得られたバラバラのPCR productをreference genomeに
当てはめていく作業）におけるエラーの除去、
・・・次世代シークエンサー、およびデータ解析上のエラーの除去
・・・いかに「最後の1つ」に落とし込んで行くか
・・・PCRを実際に全てのサンプルに行い、SNPやIndelの存在を確認する

P	sumP	proportion	sum proportion chr	reference	alternative	type	rs#	gene
0.496970352	4.71489148	0.105415641	0.105415641	22 A	AC	INDEL	rs74910017	RIBC2
0.487101359		0.103322264	0.208737904	4 TTTTG	T	INDEL	rs140019361	DCHS2
0.469316679		0.099549839	0.308287743	12 T	G	SNV	.	FAM186A
0.386942964		0.08207701	0.390364753	11 TTGCTGCTGCT	T	INDEL	rs141671766	MAML2
0.342793265		0.072712127	0.46307688	11 C	CTGCCCTACT	INDEL	rs11268490	DNHD1
0.179907724		0.038161407	0.501238287	12 T	G	SNV	rs12297653	FAM186A
0.176200242		0.037374989	0.538613276	7 AT	A	INDEL	rs148691355	VWDE
0.173640818		0.036832093	0.575445369	8 CCTG	C	INDEL	rs5881390	MAP3K4
0.171714706		0.036423532	0.611868901	12 A	G	SNV	rs10876022	FAM186A
0.140107036		0.029719022	0.641587924	7 T	TC	INDEL	rs34061445	KCP
0.139262869		0.02953996	0.671127884	12 T	A	SNV	.	FAM186A
0.13917386		0.02952108	0.700648964	18 A	ATCT	INDEL	rs34960489	GREB1L
0.138096306		0.029292513	0.729941477	11 CA	C	INDEL	.	MUC2
0.132340942		0.028071705	0.758013182	19 GGCTCCTTCA	G	INDEL	rs67168196	FAM71E2
0.129976442		0.027570156	0.785583338	11 T	TAG	INDEL	rs113897236	RP11-397M16.4
0.123161477		0.026124589	0.811707927	16 G	C	SNV	rs7199961	ZNF469
0.110918341		0.023527617	0.835235544	12 C	A	SNV	rs11114486	PTPRQ
0.060285191		0.012787487	0.848023031	3 T	C	SNV	rs9833423	C3orf77

遺伝学的解析プログラム集

各民族での多型頻度のベイズ理論による計算

一般人口で n 人を調べ，r 人が遺伝子 A を有することが分かったとき，一般人口中の遺伝子 A 頻度を表す分布の確率密度関数 $f(x)$ は，

$$f(x) = Be_x(1+r, 1+n-r)$$

になる．ここで $Be_x(\alpha,\beta)$ は（第一種）ベータ分布である． Excel には関数があり，
$Be_x(1+r, 1+n-r)$ = BETA.DIST(x, 1+r, 1+n-r, FALSE)

なお，

$$Be_x(\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} \quad B(\alpha,\beta)$$

分母の $B(\alpha,\beta)$ はベータ関数であり，

$$B(\alpha,\beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt$$

遺伝子 A の頻度の分布の累積密度関数は $I_x(\alpha,\beta)$ 正則化された不完全ベータ関数となる

$$I_x(\alpha,\beta) = \frac{\int_0^x t^{\alpha-1}(1-t)^{\beta-1}dt}{\int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt}$$

$I_x(\alpha,\beta)$ も excel に組み込み関数がある．

$I_x(\alpha,\beta)$ = BETA.DIST(x, a, β , TRUE)

上記の「一般人口で n 人を調べ，r 人が遺伝子 A を有する」場合，一般人口中の頻度が 0.05 以上である確率 P は，

P = 1 - BETA.DIST(0.05, 1+r, 1+n-r, TRUE)

となる．

優性遺伝モデルでは Homo+Hetero を遺伝子 A 保有者，劣性遺伝子モデルでは Homo のみを遺伝子 A 保有者と読み替える．これを，reference，alternative に関して行う．

ESP 集団はサンプル数が大きいので，ESP 集団との比較は上記の式を直接使用する．例えば，European American での遺伝子 A の頻度が a であり，日本人 n 人を調べ，r 人が遺伝子 A を有することが分かったとき，日本人での遺伝子頻度が European American の 10 倍である確率 P は，

P = 1 - BETA.DIST(10a, 1+r, 1+n-r, TRUE)

で計算できる．

サンプル数の少ないもの同士，例えばCHBとJPTでの比較が必要な場合は，以下のようなやり方が必要かもしれない．

CHBでの遺伝子Aの頻度がAであり，CHB N人を調べ，R人が遺伝子Aを持っていた．JPTでの遺伝子Aの頻度がaであり，JPT n人を調べ，r人が遺伝子Aを持っていた．ここで，日本人での遺伝子A頻度がCHBでの遺伝子A頻度の10倍以上の確率は，$0 \le x \le 1$の区間をΔxに細分化して（例えば0.01の100区間に分けて）考える．

$(x, x+\Delta x)$の区間のCHBの確率は

$$Be_x(1+R, 1+N-R)\Delta x$$

または，

$$I_{x+\Delta x}(1+R, 1+N-R) - I_x(1+R, 1+N-R)$$

と書ける．この場合JPTの確率が10倍以上となる確率は，

$$1 - I_{10x}(1+r, 1+n-r)$$

よって，$(x, x+\Delta x)$の区間でJPTがCHBの10倍以上となる確率は，

$$\left(I_{x+\Delta x}(1+R, 1+N-R) - I_x(1+R, 1+N-R)\right)\left((1 - I_{10x}(1+r, 1+n-r)\right)$$

これを，$0 \le x \le 1$（実際はxは0.1以上を取れないので$0 \le x \le 0.1$）の範囲で足し合わせる．積分が難しいので，実際に加えてしまう方が簡単であろう．

Ruby による program

```
"likelihood.rb"
#!/usr/bin/env ruby
#!/usr/local/bin/ruby
$: << File.dirname( __FILE__ ) + "/sub"
require "Math2"
class Math2Class < Object
  include Math2
end

math2Class = Math2Class.new

Ref_Dominant = 0; Ref_Recessive = 1; Alt_Dominant = 2; Alt_Recessive = 3

CLCCEU = 21; CLCJPT = 27; CLCCHB = 33; CLCIreTar = 45; CLCIPF = 54; CLCDOC = 63
modelAr = [ "Ref_Dominant", "Ref_Recessive", "Alt_Dominant", "Alt_Recessive" ]
targetAr = []; targetAr[ CLCIreTar ] = "CLCIreTar"; targetAr[ CLCIPF ] = "CLCIPF"; targetAr[ CLCDOC ] = "CLCDOC"

Model = Alt_Dominant
Target = CLCIreTar

JapaneseFrequencyThreshold = 0.05
FoldOfTargetGeneToWestern = 10.0
FoldOfTargetGeneToChinese = 1.0

argvFile = ARGV.shift
io = open( argvFile, "r" )
buf = io.read
buf.gsub!( /\r/, "\n" )
if buf[ -1, 1 ] != "\n" then buf = buf + "\n" end
io.close
io = open( argvFile, "w" )
io.write buf
io.close

ioWorld = open( "#{ File.dirname( __FILE__ ) }/#{ File.basename( argvFile, ".*") }_compWorld_#{ targetAr[ Target
] }_freqInJPN_#{ JapaneseFrequencyThreshold }_foldToCEU_#{ FoldOfTargetGeneToWestern }_foldToCHB_#{
FoldOfTargetGeneToChinese }.txt", "w" )
ioLog = open( "#{ File.dirname( __FILE__ ) }/#{ File.basename( argvFile, ".*") }_Log_compWorld_#{ targetAr[ Target
] }_freqInJPN_#{ JapaneseFrequencyThreshold }_foldToCEU_#{ FoldOfTargetGeneToWestern }_foldToCHB_#{
FoldOfTargetGeneToChinese }.txt", "w" )
ioLog.puts "Chromosome\tPosition\tRef\tAlt\tInfo\t\t\t\tCLCInfo\t\t\t\tp_jptDisFreqGTTreshold\tp_jptDisFreqCEU\
tp_jptDisFreqCHB\tp_compWorld"
io = open( argvFile, "r" )
```

```
line = io.gets
while line = io.gets
 ar = line.chomp!.split( "\t" )
 if ar == [] then next end
 case ar[ 0 ]
 when "X"
  ar[ 0 ] = "23"
 when "Y"
  ar[ 0 ] = "24"
 when "M"
  ar[ 0 ] = "25"
 end
 ceuAll = ar[ CLCCEU ].to_f + ar[ CLCCEU + 1 ].to_f + ar[ CLCCEU + 2 ].to_f
 chbAll = ar[ CLCCHB ].to_f + ar[ CLCCHB + 1 ].to_f + ar[ CLCCHB + 2 ].to_f
 jptAll = ar[ CLCJPT ].to_f + ar[ CLCJPT + 1 ].to_f + ar[ CLCJPT + 2 ].to_f
 targetAll = ar[ Target ].to_f + ar[ Target + 1 ].to_f + ar[ Target + 2 ].to_f
 case Model
 when Ref_Dominant
  ceuDis = ar[ CLCCEU ].to_f + ar[ CLCCEU + 1 ].to_f
  chbDis = ar[ CLCCHB ].to_f + ar[ CLCCHB + 1 ].to_f
  jptDis = ar[ CLCJPT ].to_f + ar[ CLCJPT + 1 ].to_f
  targetDis = ar[ Target ].to_f + ar[ Target + 1 ].to_f
 when Ref_Recessive
  ceuDis = ar[ CLCCEU ].to_f
  chbDis = ar[ CLCCHB ].to_f
  jptDis = ar[ CLCJPT ].to_f
  targetDis = ar[ Target ].to_f
 when Alt_Dominant
  ceuDis = ar[ CLCCEU + 1 ].to_f + ar[ CLCCEU + 2 ].to_f
  chbDis = ar[ CLCCHB + 1 ].to_f + ar[ CLCCHB + 2 ].to_f
  jptDis = ar[ CLCJPT + 1 ].to_f + ar[ CLCJPT + 2 ].to_f
  targetDis = ar[ Target + 1 ].to_f + ar[ Target + 2 ].to_f
 when Alt_Recessive
  ceuDis = ar[ CLCCEU + 2 ].to_f
  chbDis = ar[ CLCCHB + 2 ].to_f
  jptDis = ar[ CLCJPT + 2 ].to_f
  targetDis = ar[ Target + 2 ].to_f
 end
 #Japanese frequency > JapaneseFrequencyThreshold
 p_jptDisFreqGTTreshold = 1.0 - math2Class.reg_incomplete_beta( JapaneseFrequencyThreshold, 1 + jptDis, 1 +
jptAll - jptDis )
 #Japanese frequency > FoldOfTargetGeneToCEU
#  p_jptDisFreqCEU = 1.0 - math2Class.reg_incomplete_beta( [ FoldOfTargetGeneToWestern * ceuDis.to_f / ceuAll.
to_f, 1.0 ].min, 1 + jptDis, 1 + jptAll - jptDis )
 p_jptDisFreqCEU = math2Class.product_reg_incomplete_beta( FoldOfTargetGeneToWestern, 1 + ceuDis, 1 + ceuAll
```

```
- ceuDis, 1 + jptDis, 1 + jptAll - jptDis )
  #Difference between CHB and JPT
  p_jptDisFreqCHB = math2Class.product_reg_incomplete_beta( FoldOfTargetGeneToChinese, 1 + chbDis, 1 + chbAll
- chbDis, 1 + jptDis, 1 + jptAll - jptDis )
#   p_jptDisFreqCHB = 1.0 - math2Class.reg_incomplete_beta( [ FoldOfTargetGeneToChinese * chbDis.to_f / chbDis.
to_f, 1.0 ].min, 1 + jptDis, 1 + jptAll - jptDis )
  #p p_CHB_JPT = math2Class.product_reg_incomplete_beta( 1.0, 1 + 36, 1 + 72, 1 + 36, 1 + 72 )
  #p p_CHB_JPT = math2Class.product_reg_incomplete_beta( 1.0, 1 + jptDis, 1 + jptAll - jptDis, 1 + jptDis, 1 + jptAll -
jptDis )
  #p [chbDis,jptDis ]###
  #p_level3
  p_compWorld = p_jptDisFreqGTTreshold * p_jptDisFreqCEU * p_jptDisFreqCHB
  ###
  ioWorld.puts "#{ p_compWorld }"
  ioLog.puts "#{ ar[ 0 ] }\t#{ ar[ 1 ] }\t#{ ar[ 2 ] }\t#{ ar[ 3 ] }\t#{ ar[ 4 ] }\t#{ ar[ 5 ] }\t#{ ar[ 6 ] }\t#{ ar[ 7 ] }\t#{ ar[ 8
] }\t#{ ar[ 9 ] }\t#{ ar[ 10 ] }\t#{ ar[ 11 ] }\t#{ p_jptDisFreqGTTreshold }\t#{ p_jptDisFreqCEU }\t#{ p_jptDisFreqCHB
}\t#{ p_compWorld }"
end
ioWorld.close
ioLog.close
io.close
```

C による extension

```
"math2.h"
void Init_Math2( void );
static VALUE wrap_product_reg_incomplete_beta( VALUE self, VALUE r_factor, VALUE r_a1, VALUE r_b1, VALUE r_
a2, VALUE r_b2 );
static VALUE wrap_reg_incomplete_beta( VALUE self, VALUE r_x, VALUE r_a, VALUE r_b );
static double product_reg_incomplete_beta_c( double factor, double a1, double b1, double a2, double b2 );
static double reg_incomplete_beta_c( double x, double a, double b );
static double beta( double x, double y );
static double my_gamma( double x );
static double loggamma( double x );

"math2.c"
#include "ruby.h"
#include "math.h"
#include "stdio.h"
#include <math2.h>

#define PI     3.14159265358979324  /* pi */
#define LOG_2PI 1.83787706640934548  /* log2 / pi */
#define N      8

#define B0  1              /* Bernoulli number */
```

```
#define B1  (-1.0 / 2.0)
#define B2  ( 1.0 / 6.0)
#define B4  (-1.0 / 30.0)
#define B6  ( 1.0 / 42.0)
#define B8  (-1.0 / 30.0)
#define B10 ( 5.0 / 66.0)
#define B12 (-691.0 / 2730.0)
#define B14 ( 7.0 / 6.0)
#define B16 (-3617.0 / 510.0)

static double loggamma( double x ){  /* logGamma */
        double v, w;

        v = 1.0;
        while ( x < N ) { v *= x;  x++; }
        w = 1 / ( x * x );
        return ( ( ( ( ( ( ( B16 / ( 16 * 15 ) )  * w + ( B14 / ( 14 * 13 ) ) ) * w
                + ( B12 / ( 12 * 11 ) ) ) * w + ( B10 / ( 10 *  9 ) ) ) * w
                + ( B8  / (  8 *  7 ) ) ) * w + ( B6  / (  6 *  5 ) ) ) * w
                + ( B4  / (  4 *  3 ) ) ) * w + ( B2  / (  2 *  1 ) ) ) / x
                + 0.5 * LOG_2PI - log( v ) - x + ( x - 0.5 ) * log( x );
}

static double my_gamma( double x ){  /* gamma */
        if( x < 0 ) {
                return PI / ( sin( PI * x ) * exp( loggamma( 1 - x ) ) );
        }
        return exp( loggamma( x ) );
}

static double beta( double x, double y ){  /* beta */
        return exp( loggamma( x ) + loggamma( y ) - loggamma( x + y ) );
}

static double reg_incomplete_beta_c( double x, double a, double b ){
        int k;
        double p1, q1, p2, q2, d, previous;

        if ( a <= 0 ) return HUGE_VAL;
        if ( b <= 0 ) {
                if ( x <  1 ) return 0;
                if ( x == 1 ) return 1;
                /* else */  return HUGE_VAL;
        }
        if ( x > ( a + 1 ) / ( a + b + 2 ) )
```

```
                return 1 - reg_incomplete_beta_c( 1 - x, b, a );
        if ( x <= 0 )  return 0;
        p1 = 0;  q1 = 1;
        p2 = exp( a * log( x ) + b * log( 1 - x )
                + loggamma( a + b ) - loggamma( a ) - loggamma( b ) ) / a;
        q2 = 1;
        for ( k = 0; k < 200; ) {
                previous = p2;
                d = - ( a + k ) * ( a + b + k ) * x
                        / ( ( a + 2 * k ) * ( a + 2 * k + 1 ) );
                p1 = p1 * d + p2;  q1 = q1 * d + q2;
                k++;
                d = k * ( b - k ) * x / ( ( a + 2 * k - 1 ) * ( a + 2 * k ) );
                p2 = p2 * d + p1;  q2 = q2 * d + q1;
                if ( q2 == 0 ) {
                        p2 = HUGE_VAL;  continue;
                }
                p1 /= q2;  q1 /= q2;  p2 /= q2;  q2 = 1;
                if ( p2 == previous ) return p2;
        }
        printf( "regulated incomplete beta function does not converge.\n" );
        return p2;
}

static double product_reg_incomplete_beta_c( double factor, double a1, double b1, double a2, double b2 ) {
        double x = 0.0, dif = 0.01, p = 0.0;
        while( ( x * factor <= 1.0 ) && ( x + dif <= 1.0 ) ){
                p = p + ( reg_incomplete_beta_c( x + dif, a1, b1 ) - reg_incomplete_beta_c( x, a1, b1 ) ) * ( 1.0 - reg_
incomplete_beta_c( x * factor, a2, b2 ) );
                x += dif;
        }
        return p;
}

static VALUE wrap_reg_incomplete_beta( VALUE self, VALUE r_x, VALUE r_a, VALUE r_b ){
        return rb_float_new( reg_incomplete_beta_c( NUM2DBL( r_x ), NUM2DBL( r_a ), NUM2DBL( r_b ) ) );
}

static VALUE wrap_product_reg_incomplete_beta( VALUE self, VALUE r_factor, VALUE r_a1, VALUE r_b1, VALUE r_
a2, VALUE r_b2 ) {
        return rb_float_new( product_reg_incomplete_beta_c( NUM2DBL( r_factor ), NUM2DBL( r_a1 ), NUM2DBL(
r_b1 ), NUM2DBL( r_a2 ), NUM2DBL( r_b2 ) ) );
}

void Init_Math2( void ) {
```

```
    volatile VALUE module;
    module = rb_define_module( "Math2" );

    rb_define_method( module, "reg_incomplete_beta", wrap_reg_incomplete_beta, 3 );
    rb_define_method( module, "product_reg_incomplete_beta", wrap_product_reg_incomplete_beta, 5 );
}
```

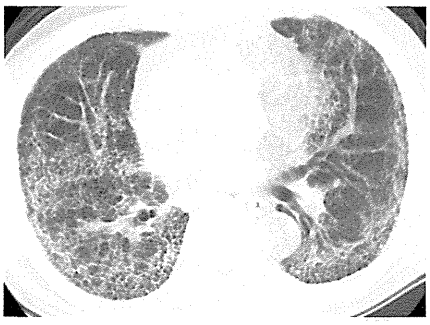
集積症例画像集

特発性肺線維症急性増悪

掲載したものは収集症例の一部である．DAD 様の所見がある場合は，
IPF を広めに診断している

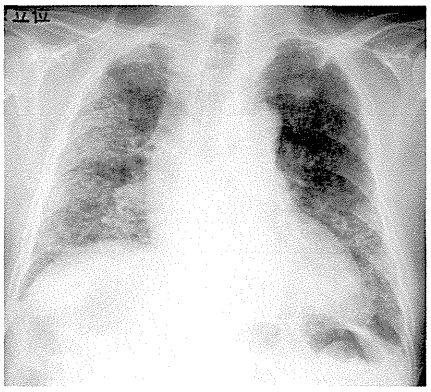
男性　　IPF 急性増悪

増悪前CT画像　　増悪時CT画像　　増悪後CT画像



2007/12/22

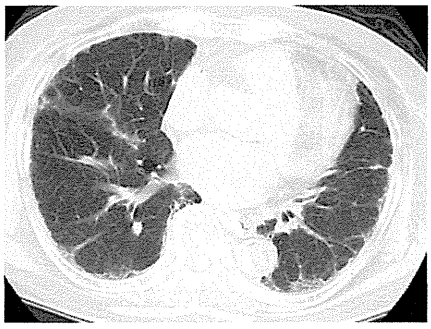
増悪前XP　　増悪時XP　　増悪後XP



2007/12/21

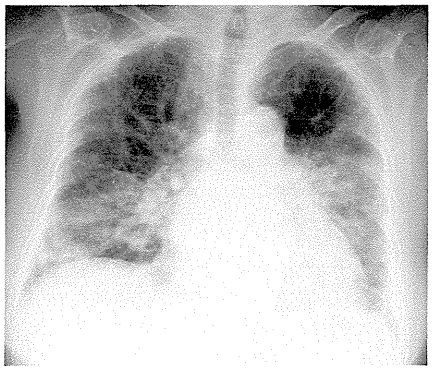
女性　　IPF 急性増悪

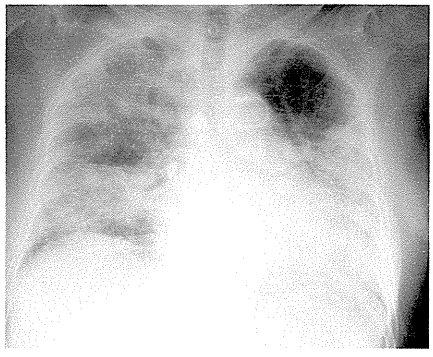
増悪前CT画像　　増悪時CT画像　　増悪後CT画像



2008/06/11　　2008/12/08

増悪前XP　　増悪時XP　　増悪後XP



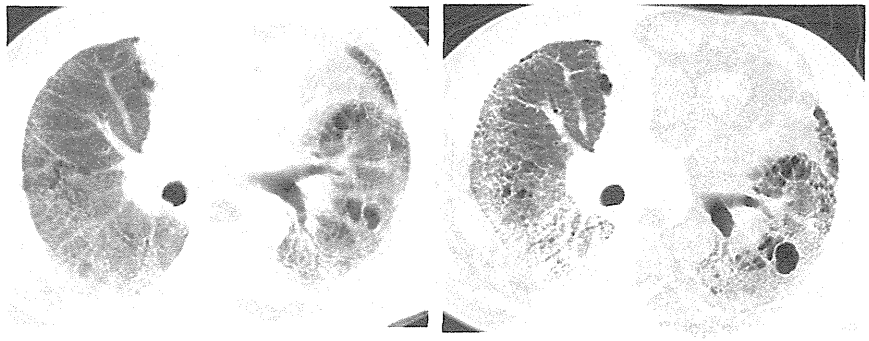


2008/12/08　　2009/01/06

139

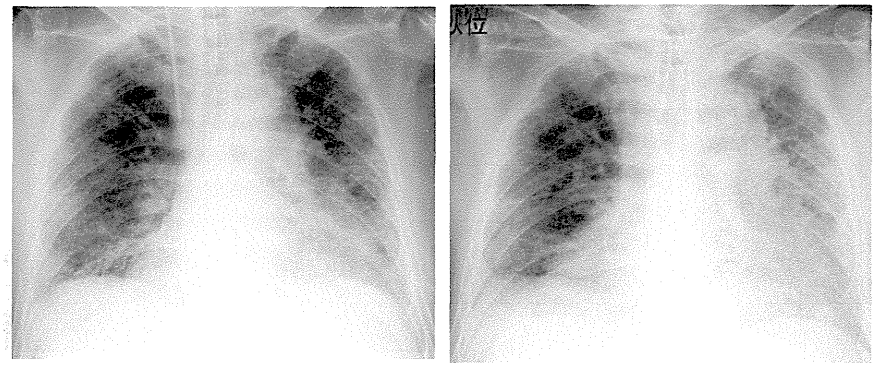
男性　　IPF 急性増悪

増悪前CT画像　　増悪時CT画像　　増悪後CT画像



2009/01/27　　2009/03/05

増悪前XP　　増悪時XP　　増悪後XP



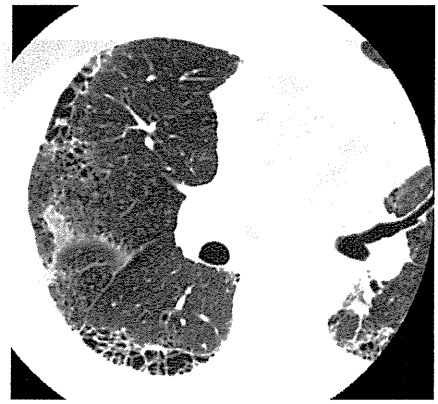
2009/01/26　　2009/03/06

153

男性

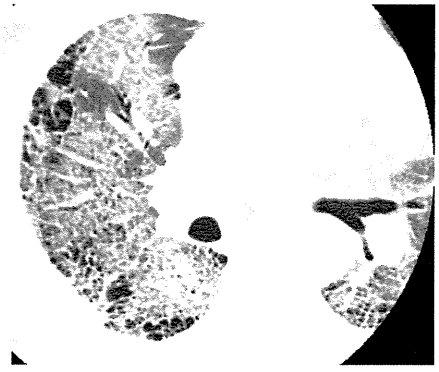
IPF 急性増悪

増悪前CT画像



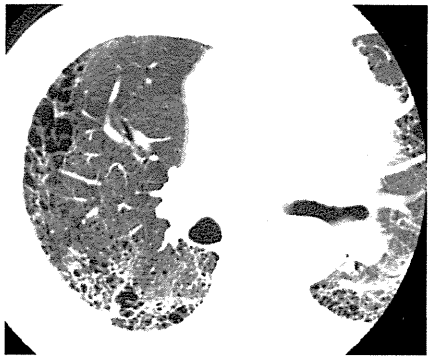
2008/09/01

増悪時CT画像



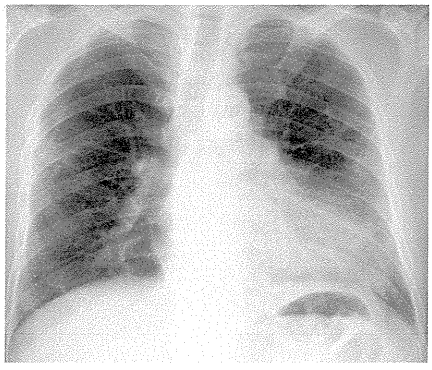
2009/04/20

増悪後CT画像



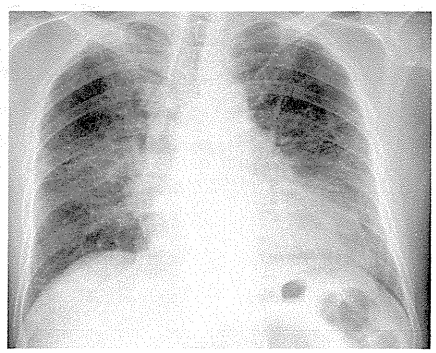
2009/05/15

増悪前XP



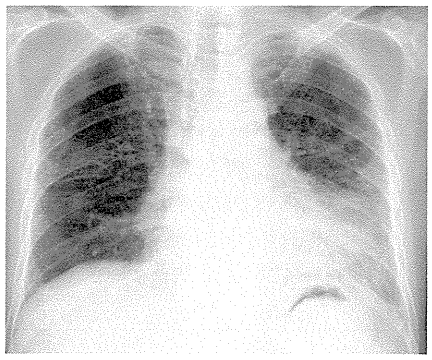
2008/07/09

増悪時XP



2009/04/20

増悪後XP



2009/05/13

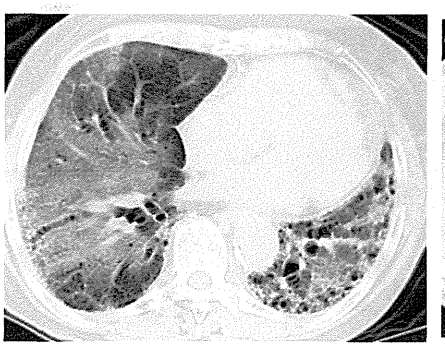
:性

IPF 急性増悪

増悪前CT画像

2009/02/16

増悪時CT画像

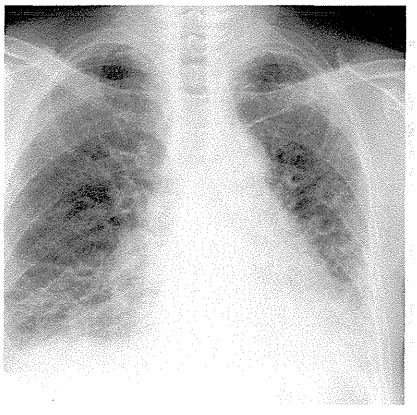


2009/04/22

増悪後CT画像

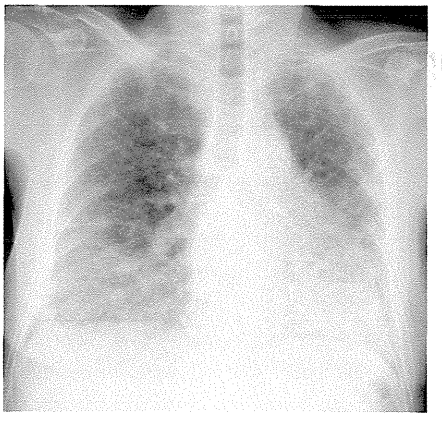
2009/06/22

増悪前XP



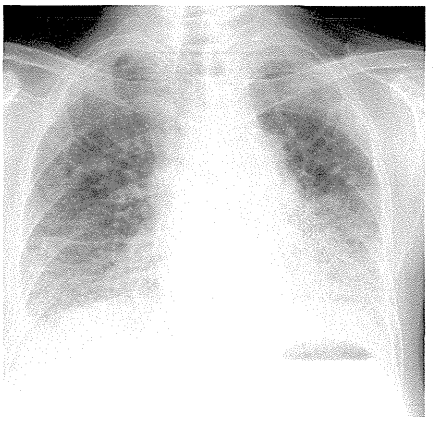
2009/01/30

増悪時XP



2009/04/22

増悪後XP

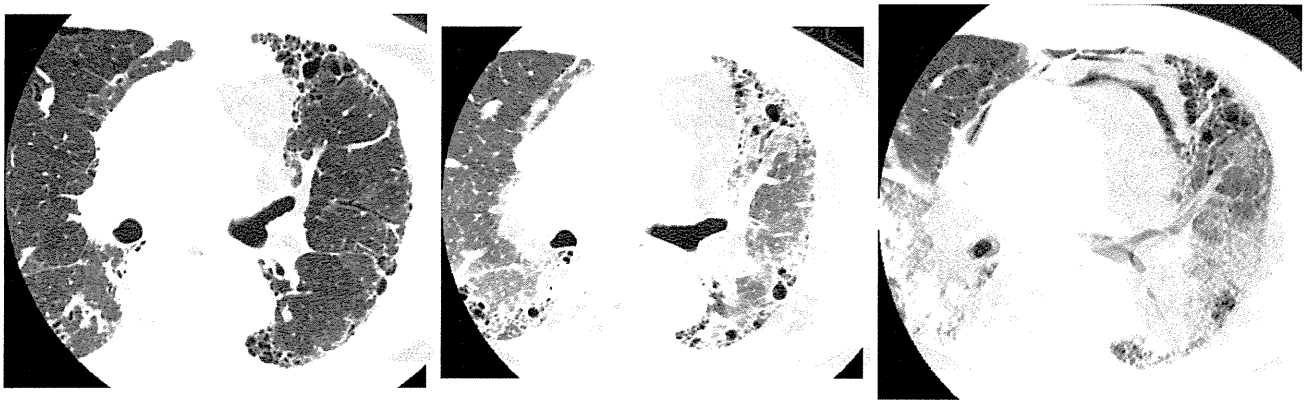


2009/06/18

169

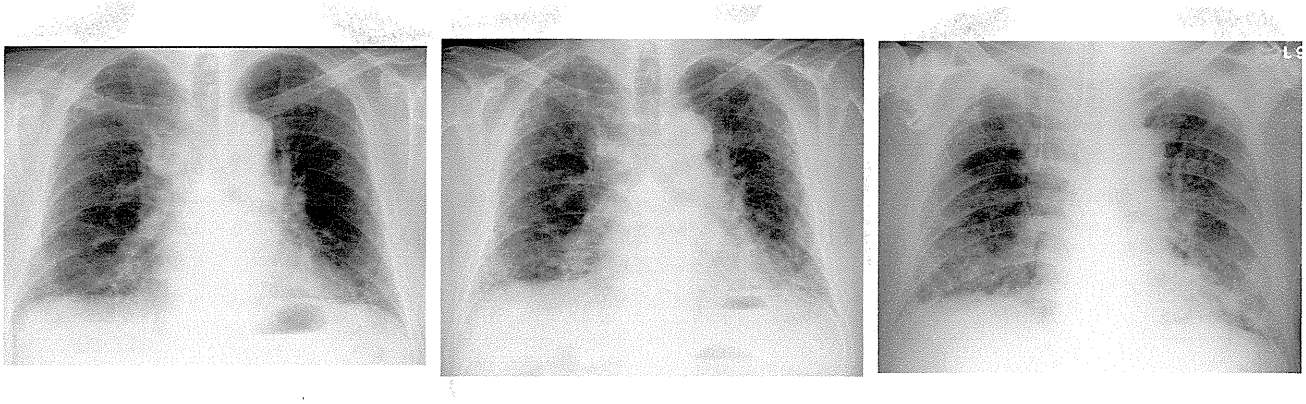
男性　　IPF 急性増悪

増悪前CT画像　　増悪時CT画像　　増悪後CT画像



2009/03/18　　2009/05/11　　2009/06/12

増悪前XP　　増悪時XP　　増悪後XP

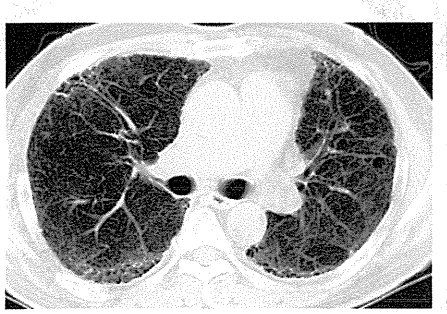
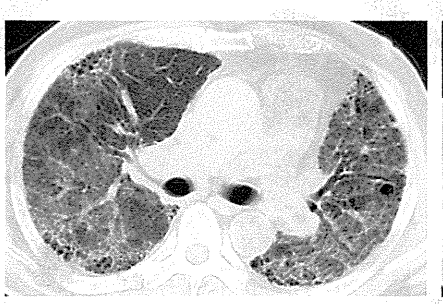
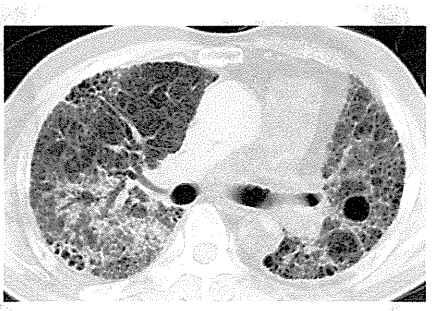


2009/03/18　　2009/05/11　　2009/06/12

171

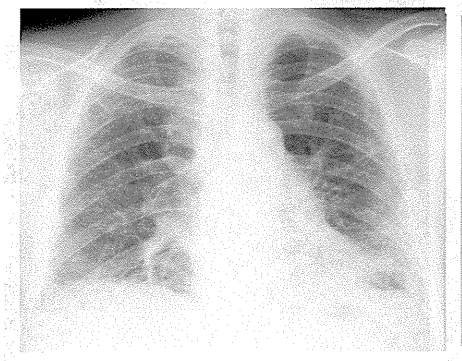
男性　　IPF 急性増悪

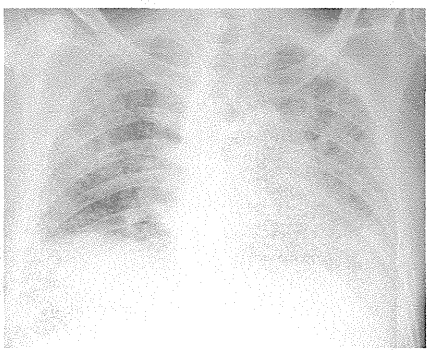
増悪前CT画像　　増悪時CT画像　　増悪後CT画像

2009/02/25　　2009/06/13　　2009/06/23

増悪前XP　　増悪時XP　　増悪後XP



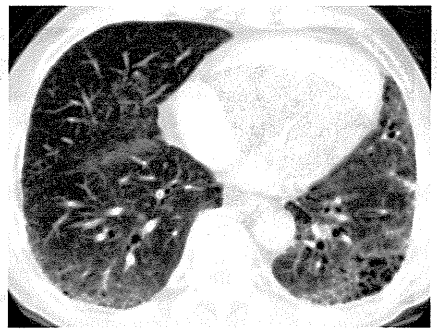


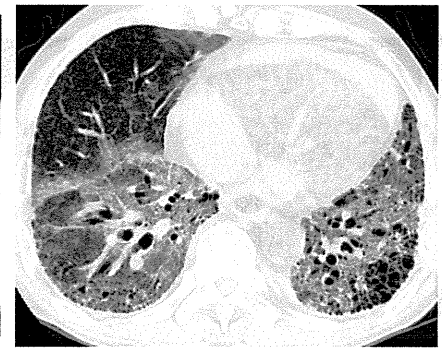
2009/02/25　　2009/06/13　　2009/06/24

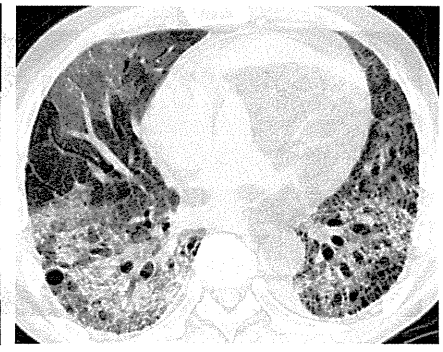
183

男性　　IPF 急性増悪

増悪前CT画像　　増悪時CT画像　　増悪後CT画像

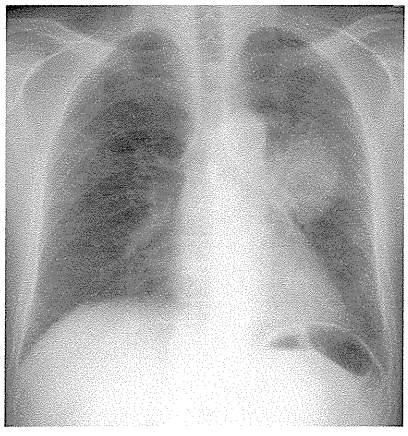


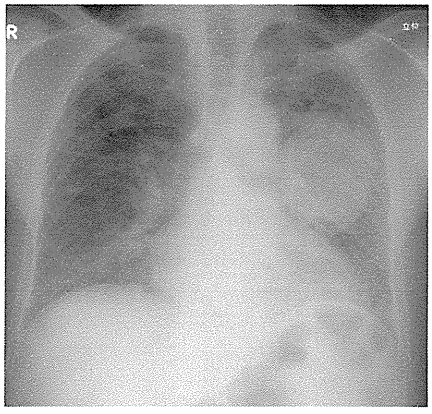


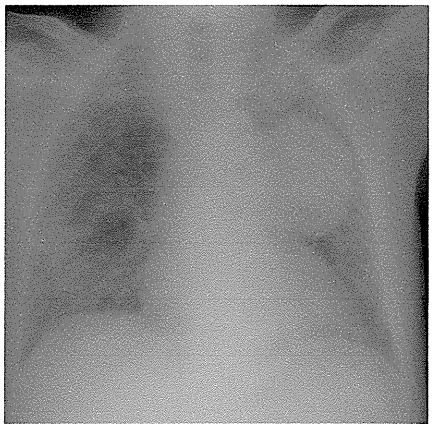


2009/06/08　　2009/06/28　　2009/07/06

増悪前XP　　増悪時XP　　増悪後XP







2009/06/08　　2009/06/28　　2009/07/06