

正規分布アプレットについて

船 木 洋 一

1. はじめに

入学時の選抜の多様性から、さまざまな基礎学歴の学生が入学してきています。文系では、数学の基礎学歴のばらつきが目立ちます。そこで数学を使用する教科を学習させるために、工夫が必要とされています。統計学や OR がそうです。そのためか、近年 OR 学会では、文系学生に対する OR 教育が研究対象として取り上げられ、統計学会では、かなり前から、統計教育に関する研究報告が行われています。統計学会での対象は文科系と限定しているわけではありません。パソコンの普及にともない、パソコンを用いた統計教育の研究報告も行われています。統計教育では数値計算を含む実習が欠かせませんが、その際、パソコンは威力を発揮します。

本稿では、さまざまな基礎学歴の学生が自学自習できるようにと作っている、統計教育用 HTML ファイルの、正規分布アプレットについて報告します。パソコンを使った統計教育です。このようなやり方をすると、数表を引くという作業はいらなくなります。これだけパソコンが普及し、これからは普及すると思われます。さらに、統計が使用される現場では、電卓ではなく、コンピュータが使用されるはずで、そのような点から、数表を引く練習や、計算練習を少なくしても、さらには、対象学生によってはなくても良いのでは、と思っています。

統計表を引く作業は分布関数の理解の助けになります。ところが、単純な作業にもかかわらず、そこでつまづく人もいます。面倒を嫌がり、話を聞いて、簡単だ、できる、として実際練習してみないからです。今回の正規分布アプレットには使用に当たって数表を引くほどの面倒はありません。興味を持って使用してもらえるのでは、と思っています。分布関数の理解につながります。

さまざまな確率モデルが、正規分布を使って作られています。そのようなところから正規分布は統計学の中では大事なテーマとなっています。それで、現在製作中の教材でも、正規分布に関する HTML ファイルとそこに載るアプレットは、一番大事なものです。

2. 「正規分布」の内容

正規分布に関する学習では

正規分布の形と確率の特徴

標準正規分布と z 変換

標準正規分布表の使い方

中心極限定理

などが、最小限行われます [4] [6] [8]



本による場合のスケッチ。

正規分布の形は正規確率密度関数の数式

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

を書き、その図を描くことによって示されます。また、図に関する特徴として、 $\mu \pm \sigma, \mu \pm 2\sigma, \mu \pm 3\sigma$ 範囲の確率が示されます。

次に正規分布に関する確率計算のため正規確率変数の z 変換が示されます。 z 変換後の確率変数が、標準正規分布をすることが示されます。そのことを利用して、任意の正規分布のさまざまな確率が、標準正規分布の確率から求められることが示されます。

そこで、正規分布に関する確率計算のために標準正規分布表の引き方が示されます。正規分布のところの実習では、 z 変換と、標準正規分布表の引き方が練習されます。ここでの、標準正規分布表を使用する練習は、ほかの t 分布、 F 分布、 χ^2 分布などの表の使い方の学習にもなります。

正規分布が多用され、それで大事であるという、背景を説明するものとして、中心極限定理があります。これは とは別の、より後の章に有ります。

以上が通常の教科書による、正規分布の最小限の学習内容です（数理統計学の教科書ではちがいます）。これらのことが最小限 HTML ファイルでも学習できるようにします。最後の中心極限定理は別立てで説明します。そのためのアプレットも別立てで作ります。今回の報告では扱いません。

3 . HTML ファイルに書くことの選択とアプレットでさせること

HTML ファイルによる学習はコンピュータによる学習なので、種々の確率計算は直接させることができます。これから、統計分析を行うときには、コンピュータを使用し数表は使用しないでしょう。現在通常行われている教科書による学習と歩調を合わせるために、標準正規分布表の使い方は HTML ファイルの中に有っても良いとは思いましたが、触れないことにしました。（2、 はいいりません。）

瞬時に、任意の平均と標準偏差の分布の図を描くことができます。そこで、アプレットではインターラクティブ性を発揮し、学習者から平均と、標準偏差を入力してもらい、その正規密度関数を描かせることにします。

標準正規分布表を使ってできる、あらゆる計算が、その一つのアプレット上で、できるようにします。視覚的に量がわかるようにします。

一つの正規分布アプレットに盛る項目として

正規密度関数の描画

累積正規分布関数の描画

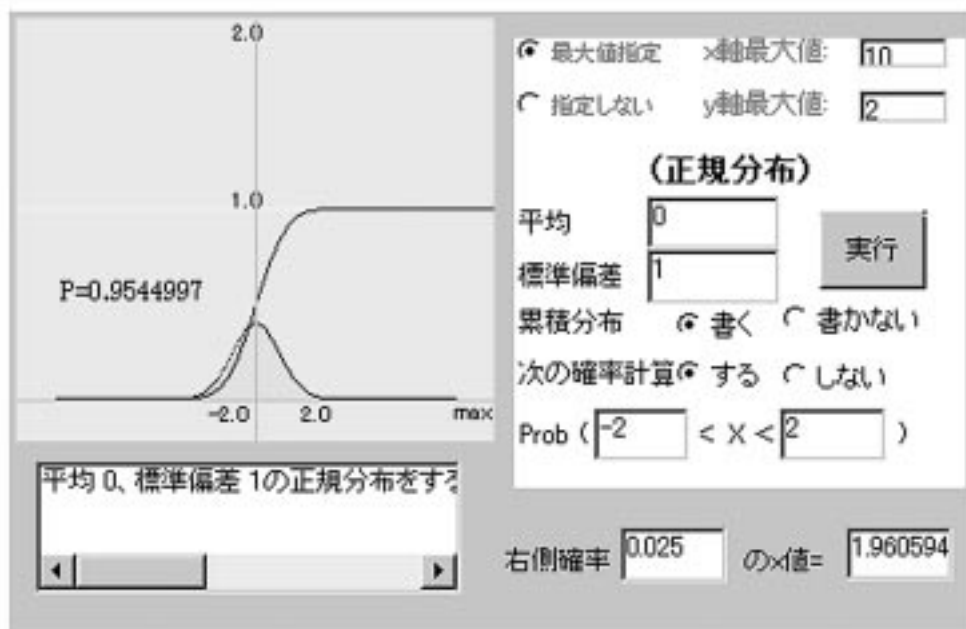
確率の計算

与えられた右側確率に対する x の値

コメントを入れるテキスト窓

を選びました。

4．正規分布アプレットの意図と説明



正規密度関数の描画

学習者が指定した、平均 μ と標準偏差 σ に対する正規密度関数

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

を描きます。学習者が指定した平均と分散を持つ正規分布を描くことにより、「こんな平均だと分布はどこに行くか」、「こんな分散だと分布はどんなに平になるか」など、興味を持った受身でない学習が可能です。平均と、標準偏差に依存して正規分布の位置や大きさが変わります。

図の位置や大きさが変わってもいいように、スケール自動モードと指定モードを選択できるようにしました。いずれの場合も x 座標の中心を 0 と固定しています。スケールを指定するモードでは、学習者が、表示する縦軸と横軸の最大値を選択できます。操作によって、密度関数の狭い部分の拡大図もみられます。全体が描かれなくなりますが、縦軸最大値が

$$\frac{1}{\sigma\sqrt{2\pi}}$$

以下でもかまいません。スケール指定のモードでは、スケールを固定しておくことにより種々の平均と標準偏差をもつ正規分布の相対的な位置関係が学習できます。ここもインタラクティブなど

ここで、遊びの要素も感じられる(?) ところです。スケール自動モードでは、密度関数のグラフがきちんと収まるように、縦軸最大値が2、横軸最大値が $\mu + 3.5$ になるようにします。x座標のスケールが変化します。

統計学では図を描くところから情報を得ます[1][5]。実際のデータでも、またフリーハンドでもありませんが、コンピュータに、パラメータを変えて図を描かせることにより、分布関数に対する学習効果を期待できます。

累積分布関数の描画

確率分布のもう一つの表し方である累積分布

$$F(x) = \int_{-\infty}^x \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] dx$$

も一緒に描画します。0 から 1 に伸びる正規分布の累積分布関数を見ることにより正規分布の確率の特徴を観察できます。一般的な累積分布関数の特徴

$$F(+\infty) = 1$$

$$F(-\infty) = 0$$

$$x > y \quad \text{ならば} \quad F(x) > F(y)$$

など、対応する密度関数とあわせて描かれることにより、視覚的に、意味が理解されます。密度関数の観察には邪魔になることがあるかもしれません。それでラジオボタンで描く描かないを選択できるようにしました。

確率の計算

コンピュータによる学習なので、計算は簡単にできます。密度関数 $f(x)$ の描画だけでなく与えられた範囲 (a,b) の確率

$$\int_a^b f(x) dx$$

を計算させます。緑色で確率を計算した範囲を図に示すようにしてあります。 $\mu \pm \sigma, \mu \pm 2\sigma, \mu \pm 3\sigma$ の範囲だけでなく、 任意の区間の確率を計算させてみるができます。左の境界値に c 、右の境界値に極端に大きな値を指定することにより $x=c$ の右側確率 $1 - F(c)$ を計算できます。逆に左の境界値に極端に大きな値、右の境界値に c を指定することにより、累積確率 $F(c)$ を計算できます。平均0、標準偏差1に対して計算させると標準正規分布表を引く変わりになります。(コンピュータで確率を計算できるので、数表を引く変わりになるということとは変な話ですが。)

z変換

正規分布の確率を求める場合、教科書では標準正規分布表から求めます。そのために平均 μ 標準偏差 σ の数値 x を

$$z = \frac{x - \mu}{\sigma}$$

と z 値に変換します。この式はあっていいのですが、このアプリでは、計算のためにはそれは

必要ありません(2、 に関して)。任意の平均と標準偏差に対して、直接、計算されます。計算機内部では、もちろん標準正規分布から計算されています。これからはこの式を

$$x = \mu + \sigma z$$

と表現し、平均 μ と標準偏差 σ の確率変数と平均 0、標準偏差 1 の確率変数の関係を、初期の段階からこの形で示しておいたほうが良いように思われます。

$$x = \mu + \sigma z$$

の形は回帰分析やランダムウォークなどでも出てきます[2]。この辺は、HTML ファイルの文章で述べます。



一つの使用例、偏差値。偏差値という言葉が用いられているわりには標準的な教科書で説明がありません。それで以下のことを HTML ファイルに書き、アプレットで計算してもらいたいと思っています。

偏差値は平均50標準偏差10の正規分布確率変数の値として考えることができます。ですから、平均、標準偏差入力のところ、平均50標準偏差10とし、確率計算の数値入力のところ、左側境界値に偏差値、右側境界値に十分大きな値、たとえば10000を入れることによって、右側確率を計算させると、近似的に上位何 % 以内にいるかわかります。受験者が多くなく正規分布でうまく表されないうときは、当たりません。

以上を、HTML ファイルの文章に述べます。

与えられた、右側確率に対する x の値を求める

標準正規分布表を用いて求める値の中に、たとえば有意水準 α に対する、 x の値があります。すなわち $1 - F(x) = \alpha$ で、 x をあたえて、この式を満たす x を求めることです。これを行うのが、アプレットのこの部分です。一番右下に配置しました。任意の正規分布に対して、直接計算できます。また、左の図にマウスを持っていきクリックすることにより、マウスのある x 座標に対する標準正規分布の右側確率 $1 - F(x)$ を計算図示します。このときは、縦軸最大値を $\frac{1}{\sqrt{2\pi}}$ 、横軸最大値を 3.5 として、最大に図示しています。何度でもやって見られます。

メッセージを出すこと

テキストエリアを設け、学習者がインプットした座標軸の数値に対するコメントや、確率計算をしないときに、試みることを促すメッセージを出すようにしています。また小さな確率値が $1.4566E-5$ などのように表現されるのでそのような数値が出るケースのときには 1.4566×10^{-5} であることを説明しています。テキストエリアを通して利用者とのコミュニケーションをします。こちらから学習者への働きかけです。

5. まとめ

大学教育で必要とされる基礎知識は、きちんと高校で履修してきてもらうように、あるいはきちんと入学試験で課すようにとされていますが、選抜の多様性は増すことはあってもなくならないでしょう。この正規分布アプレットが、統計学へ関心を持つきっかけになってくれればと思っています。特徴をまとめておきます。

学習者が指定する、任意の平均と標準偏差の正規分布の図を描き確率を計算します。
 正規分布表を使用して行われるあらゆる計算が一つのアプリレットでできます。
 極力、量が、数値だけでなく視覚でとらえられるようにしています。
 学習者がスケールの大きさを変えて見られます。
 状況に応じたメッセージを出し、文による双方向性を持たせています。
 最後ですが、インターネットを通じて学習することも当然できます。
 参考文献の後に、アプリレットのコードを乗せます。簡単なコメントを入れてあります。座標に極端に大きな値が入ると図を描くときに変になるようです。その辺の処理をしました。シマンティックの visual café を利用して書きました[7]。また、暫定的ですが
<http://human.cc.hirosaki-u.ac.jp/keizai/funa3/shoukai2.htm>
 からこのアプリレットがみられます。
 なお、この研究は、科学研究補助金基盤研究 (C)(2) 課題番号 09680307 の補助を受けています。

参考文献

- [1] 畠中 道雄：なるべくグラフを画こう 日本統計学会会報 No.88、1996、1-3
- [2] 船木 洋一：統計教育におけるランダムウォークアプリレットについて 弘前大学経済研究、No.21、1998、181-194
- [3] 古林 隆：統計解析[改訂版](倍風館、1989)
- [4] 宮川 公男：基本統計学[新版](有斐閣、1991)
- [5] 宮川 雅巳：統計教育の今後「工学での統計教育」日本統計学会会報 No.87、1996、7-8
- [6] 森田 優三、久次 智雄：新統計学概論[改訂版](日本評論社、1993)
- [7] Mike Cohn : *Teach Yourself Visual Café 2 in 21 Days* (Sams.net Publishing , 1997)
- [8] Robert V. Hogg and Elliot A. Tanis : *Probability and Statistical Inference Fifth Edition* (Prentice Hall 1997)
- [9] J.W.H.Press, B.P.Flannery, S.A.Teukolsky, W.T.Vetterling : *Numerical Recipes in C* [日本語版] C言語による数値計算のレシピ(技術評論社、1993)

/*

A basic extension of the java.applet.Applet class

*/

```
import java.awt.*;
import java.applet.*;
import symantec.itools.awt.KeyPressManagerPanel;
import symantec.itools.awt.shape.Square;
public class Normalsuper extends Applet
{
    public void init()
    {
        symantec.itools.lang.Context.setApplet(this);
        //(INIT_CONTROLS
        setLayout(null);
        setSize(437,283);
        panel1 = new symantec.itools.awt.KeyPressManagerPanel();
        panel1.setLayout(null);
        panel1.setBounds(228,12,204,204);
        panel1.setBackground(new Color(16777215));
        add(panel1);
```

```

maxGroup = new CheckboxGroup();
maxyesButton = new java.awt.Checkbox(" 最大値指定 ", maxGroup, true);
maxyesButton.setBounds(0,0,72,12);
maxyesButton.setFont(new Font("Dialog", Font.PLAIN, 10));
maxyesButton.setForeground(new Color(16711680));
panel1.add(maxyesButton);
maxnoButton = new java.awt.Checkbox(" 指定しない ", maxGroup, false);
maxnoButton.setBounds(0,24,72,12);
maxnoButton.setFont(new Font("Dialog", Font.PLAIN, 10));
maxnoButton.setForeground(new Color(16711680));
panel1.add(maxnoButton);
horizontal = new java.awt.TextField();
horizontal.setText("10");
horizontal.setBounds(156,0,40,14);
panel1.add(horizontal);
vertical = new java.awt.TextField();
vertical.setText("2");
vertical.setBounds(156,24,40,15);
panel1.add(vertical);
label1 = new java.awt.Label(" 平均 ");
label1.setBounds(0,72,36,24);
panel1.add(label1);
label2 = new java.awt.Label(" 標準偏差 ");
label2.setBounds(0,96,48,24);
panel1.add(label2);
mean = new java.awt.TextField();
mean.setText("0");
mean.setBounds(60,72,60,24);
panel1.add(mean);
sdeviation = new java.awt.TextField();
sdeviation.setText("1");
sdeviation.setBounds(60,96,60,25);
panel1.add(sdeviation);
label3 = new java.awt.Label(" 累積分布 ");
label3.setBounds(0,120,55,19);
panel1.add(label3);
cumulativeGroup = new CheckboxGroup();
cumulativeyesButton = new java.awt.Checkbox(" 書く ", cumulativeGroup, true);
cumulativeyesButton.setBounds(72,120,43,21);
panel1.add(cumulativeyesButton);
cumulativenoButton = new java.awt.Checkbox(" 書かない ", cumulativeGroup, false);
cumulativenoButton.setBounds(120,120,72,16);
panel1.add(cumulativenoButton);
probabilityGroup = new CheckboxGroup();
probabilityyesButton = new java.awt.Checkbox(" する ", probabilityGroup, true);
probabilityyesButton.setBounds(72,144,48,18);
panel1.add(probabilityyesButton);
probabilitynoButton = new java.awt.Checkbox(" しない ", probabilityGroup, false);
probabilitynoButton.setBounds(120,144,60,20);
panel1.add(probabilitynoButton);
lower = new java.awt.TextField();
lower.setText("-2");
lower.setBounds(36,168,43,24);
lower.setBackground(new Color(16777215));
panel1.add(lower);
higher = new java.awt.TextField();
higher.setText("2");
higher.setBounds(120,168,48,24);
higher.setForeground(new Color(0));
panel1.add(higher);
label4 = new java.awt.Label("次の確率計算 ");

```

```

label4.setBounds(0,144,72,16);
panel1.add(label4);
label5 = new java.awt.Label("Prob (");
label5.setBounds(0,168,34,24);
panel1.add(label5);
label6 = new java.awt.Label("< X < ",Label.CENTER);
label6.setBounds(84,168,33,24);
panel1.add(label6);
label7 = new java.awt.Label(")",Label.RIGHT);
label7.setBounds(168,168,12,24);
panel1.add(label7);
label8 = new java.awt.Label("( 正規分布 )",Label.CENTER);
label8.setBounds(24,48,149,24);
label8.setFont(new Font("Dialog", Font.BOLD, 14));
panel1.add(label8);
label9 = new java.awt.Label(" x 軸最大値 :",Label.RIGHT);
label9.setBounds(84,0,60,12);
label9.setForeground(new Color(16711680));
panel1.add(label9);
label10 = new java.awt.Label(" y 軸最大値 :",Label.RIGHT);
label10.setBounds(84,24,60,12);
label10.setForeground(new Color(16711680));
panel1.add(label10);
drawButton = new java.awt.Button();
drawButton.setLabel(" 実行 ");
drawButton.setBounds(138,78,48,36);
drawButton.setBackground(new Color(12632256));
panel1.add(drawButton);
probabilityText = new java.awt.TextArea();
probabilityText.setBounds(12,204,192,60);
add(probabilityText);
label11 = new java.awt.Label(" 右側確率 ");
label11.setBounds(222,240,54,18);
add(label11);
probText = new java.awt.TextField();
probText.setText("0.025");
probText.setBounds(276,234,48,25);
add(probText);
label12 = new java.awt.Label(" の x 値 =");
label12.setBounds(330,240,48,18);
add(label12);
valueText = new java.awt.TextField();
valueText.setEditable(false);
valueText.setText(" ?");
valueText.setBounds(378,234,48,24);
valueText.setBackground(new Color(16776960));
add(valueText);
//}

panel1.setDefaultButton(drawButton);
distribution = new NormalCanvas();
distribution.setBounds(3,3,216,192);
distribution.setBackground(new Color(65535));
set();
add(distribution);
//{(REGISTER_LISTENERS
SymAction lSymAction = new SymAction();
drawButton.addActionListener(lSymAction);
SymMouse aSymMouse = new SymMouse();
probabilitynoButton.addMouseListener(aSymMouse);
probabilityyesButton.addMouseListener(aSymMouse);
maxyesButton.addMouseListener(aSymMouse);

```



```

        maxnoButton.addMouseListener(aSymMouse);
        distribution.addMouseListener(aSymMouse);
        //})
    }
    public void set(){
        int maxflag,cumuflag,probflag;
        // データ読み取り
        yoko=Double.valueOf(horizontal.getText()).doubleValue();
        tate=Double.valueOf(vertical.getText()).doubleValue();
        heikin=Double.valueOf(mean.getText()).doubleValue();
        hensha=Double.valueOf(sdeviation.getText()).doubleValue();
        kagen=Double.valueOf(lower.getText()).doubleValue();
        jougen=Double.valueOf(higher.getText()).doubleValue();
        migi=Double.valueOf(probText.getText()).doubleValue();
        if(maxyesButton.getState()==true)maxflag=1;
        else maxflag=0;
        if(cumulativesButton.getState()==true)cumuflag=1;
        else cumuflag=0;
        if(probabilityyesButton.getState()==true)probflag=1;
        else probflag=0;
        // テキスト表記部分の準備
        val=distribution.atat(kagen,jougen,heikin,hensha);
        xvalue=distribution.vofx(migi)*hensha+heikin;
        meanString=mean.getText();
        sdString=sdeviation.getText();
        lowerString=lower.getText();
        upperString=higher.getText();
        valueString=String.valueOf(val);
        xvalueString=String.valueOf(xvalue);
        message="";
        message0="";
        message1="";
        // 標準偏差は正？
        if(hensha<0) message=" 標準偏差は正です。";
        else
        {
            // 確率計算する？
            if(probflag==1)
            {
                message=" 平均 "+meanString+"、標準偏差 "+
                    sdString+" の正規分布をする確率変数 X が "+lowerString+" と
                    +upperString+" の間の値を取る確率は "+valueString+" です。\\r\\n";
                if(val<=0.001 && val>0.0)
                {
                    message0="( 数値-E) は数値 × 10 の - E 乗の事です。\\r\\n";
                }
            }
            else message=" 確率も計算してみてください。";
            if(jougen<kagen) && (probflag==1) message=" 確率計算の上限と下限を入れ替えてください。\\r\\n";
            // 軸最大は与える？
            if (maxflag != 1)
            {
                yoko=Math.round((3.5*hensha+Math.abs(heikin))*1000)/1000.0;
                sdString=String.valueOf(yoko);
                message1=" 最大値を指定しないときには、y 軸最大値は 2 , x 軸最大値 (今は "+sdString
                +" になっています) は平均値と標準偏差の値によって変わります。\\r\\n";
            }
            if ((maxflag==1)&&(tate<=0.0)&&(yoko>0)) message1=" 正規分布の密度関数は正の値を取ります。y 軸を正の値にして
            ください。\\r\\n";
            if ((maxflag==1)&&(yoko<=0.0)) message1=" x 軸最大値を正の値にしてください。\\r\\n";
        }

        // 図とテキストの表示
        distribution.set(heikin,hensha,kagen,jougen,cumuflag,probflag,yoko,tate,maxflag);
    }

```

```

        probabilityText.setText(message);
        probabilityText.appendText(message0);
        probabilityText.appendText(message1);
        valueText.setText(xvalueString);
    }
    double yoko;
    double tate;
    double heikin;
    double hensu;
    double kagen;
    double jougen;
    double val;
    double xvalue;
    double migi;
    String meanString;
    String sdString;
    String lowerString;
    String upperString;
    String valueString;
    String xvalueString;
    String message,message0,message1;
    NormalCanvas distribution;
    //(DECLARE_CONTROLS
    symantec.itools.awt.KeyPressManagerPanel panel1;
    java.awt.Checkbox maxyesButton;
    CheckboxGroup maxGroup;
    java.awt.Checkbox maxnoButton;
    java.awt.TextField horizontal;
    java.awt.TextField vertical;
    java.awt.Label label1;
    java.awt.Label label2;
    java.awt.TextField mean;
    java.awt.TextField sdeviation;
    java.awt.Label label3;
    java.awt.Checkbox cumulativeyesButton;
    CheckboxGroup cumulativeGroup;
    java.awt.Checkbox cumulativenoButton;
    java.awt.Checkbox probabilityyesButton;
    CheckboxGroup probabilityGroup;
    java.awt.Checkbox probabilitynoButton;
    java.awt.TextField lower;
    java.awt.TextField higher;
    java.awt.Label label4;
    java.awt.Label label5;
    java.awt.Label label6;
    java.awt.Label label7;
    java.awt.Label label8;
    java.awt.Label label9;
    java.awt.Label label10;
    java.awt.Button drawButton;
    java.awt.TextArea probabilityText;
    java.awt.Label label11;
    java.awt.TextField probText;
    java.awt.Label label12;
    java.awt.TextField valueText;
    //})
// イベント処理
class SymAction implements java.awt.event.ActionListener
{
    public void actionPerformed(java.awt.event.ActionEvent event)
    {

```

```

        Object object = event.getSource();
        if (object == drawButton)
            drawButton_ActionPerformed(event);
    }
}
void drawButton_ActionPerformed(java.awt.event.ActionEvent event)
{
    //{{CONNECTION
    set();
    //}}
}
class SymMouse extends java.awt.event.MouseAdapter
{
    public void mousePressed(java.awt.event.MouseEvent event)
    {
        Object object = event.getSource();
        if (object == distribution)
            distribution_MousePressed(event);
    }
    public void mouseClicked(java.awt.event.MouseEvent event)
    {
        Object object = event.getSource();
        if (object == probabilitynoButton)
            probabilitynoButton_MouseClicked(event);
        else if (object == probabilityyesButton)
            probabilityyesButton_MouseClicked(event);
        else if (object == maxyesButton)
            radioButton1_MouseClicked(event);
        else if (object == maxnoButton)
            radioButton2_MouseClicked(event);
    }
}
void probabilitynoButton_MouseClicked(java.awt.event.MouseEvent event)
{
    //{{CONNECTION
    higher.setEnabled(false);
    lower.setEnabled(false);
    //}}
}
void probabilityyesButton_MouseClicked(java.awt.event.MouseEvent event)
{
    //{{CONNECTION
    lower.setEnabled(true);
    higher.setEnabled(true);
    //}}
}
void radioButton1_MouseClicked(java.awt.event.MouseEvent event)
{
    //{{CONNECTION
    horizontal.setEnabled(true);
    vertical.setEnabled(true);
    //}}
}
void radioButton2_MouseClicked(java.awt.event.MouseEvent event)
{
    //{{CONNECTION
    vertical.setEnabled(false);
    horizontal.setEnabled(false);
    //}}
}
void distribution_MousePressed(java.awt.event.MouseEvent event)

```

```

        {
            // to do: code goes here.
            int mousex=10;
            int mousey=10;
            mousex=event.getX();

            mousey=event.getY();
            //(CONNECTION
            distribution.set(0,1,Math.round(1000.0*((double)mousex-
108.0)*7.0/216.0)/1000.0,1000,0,1,3.5,1,1);
            probabilityText.setText(" 標準正規分布。平均 0、標準偏差 1。");
            //}
        }
    }
// 正規確率分布
class NormalCanvas extends Canvas{
    public double mm;
    public double ss;
    public double realxmin;
    public double realxmax;
    public int cflag;
    public int pflag;
    public double tte;
    public double yyko;
    public int mflag;
    public NormalCanvas(){
    }
    public void set(double o,double p,double q,double r,int v,int w,double yko,double tte,int u){
        this.mm=o;
        this.ss=p;
        this.realgmin=q;
        this.realgmax=r;
        this.cflag=v;
        this.pflag=w;
        this.yyko=yko;
        this.tte=tte;
        this.mflag=u;
        repaint();
    }
    // 標準正規確率密度関数の計算ルーチン
    public double norm(double z){
        double zin,zx,z0,t,a;
        zin=z;
        z0=zin/Math.pow(2.0,0.5);
        zx=Math.abs(z0);
        t=1.0/(1.0+0.5*zx);
        a=t*Math.exp(-zx*zx-1.26551223+t*(1.00002368+t*(0.37409196+t*(0.09678418+t*(-
0.18628806+t*(0.27886807+t*(-
-1.13520398+t*(1.48851587+t*(-0.82215223+t*0.17087277)))))))));
        return 0.5d-0.5d*a;
    }
    // 確率に対する x の値
    public double vofx(double pp){
        double aa,bb;
        aa=bb=0;
        aa=-Math.log(4*pp*(1-pp));
        bb=Math.sqrt(aa*(2.06118-5.72622/(aa+11.6406)));
        if(pp>0 && pp<=0.5)return bb;
        else return -bb;
    }
    // 区間の確率計算ルーチン

```

```

public double atai(double sita,double ue, double hei, double hyou){
    double vrealxmin,vrealxmax,vmm,vss,vy;
    vrealxmin=sita;
    vrealxmax=ue;
    vmm=hei;
    vss=hyou;

    if(vrealxmax-vmm>=0)vy=0.5d+norm((vrealxmax-vmm)/vss);
    else vy=0.5d-norm((vrealxmax-vmm)/vss);
    if(vrealxmin-vmm>=0)vy=-0.5d+norm((vrealxmin-vmm)/vss);
    else vy=-0.5d-norm((vrealxmin-vmm)/vss);
    return Math.round(vy*10000000.0)/10000000.0;
}

// 図を描く
public void paint(Graphics g){
    int width=size().width;
    int height=size().height;
    int x,x0,axis,b;
    double y0,y,xtopix,realx,f,xstart,yaxis,h;
    double ytopiy;
// x 軸 y 軸の最大値が 0 以下のときは図をかかない。
    if((yyko<=0.0 || tte<=0.0) && mflag==1){
        g.setColor(Color.cyan);
        g.drawRect(0,0,width,height);
        return;
    }
// 標準偏差が負のときも何もしない。
    if(ss<0.0){
        g.setColor(Color.cyan);
        g.drawRect(0,0,width,height);
        return;
    }
// 軸スケールの決定
    f=1/(ss*Math.sqrt(2*Math.PI));
    h=3.5*ss;
    axis=0;
    yaxis=(double)height-(double)(height/10);
    x0=0;
    ytopiy=yaxis/(2*f);
    xtopix=(double)width/(2.0*h+2.0*Math.abs(mm));
    if(cflag==1)ytopiy=yaxis/2.0d;
    if(mflag==1)ytopiy=yaxis/tte;
    else tte=2.0d;
    if(mflag==1)xtopix=(double)width/(2.0d*yyko);
    else yyko=Math.round((h+Math.abs(mm))*1000)/1000.0;
    xstart=((double)x0-(double)width/(double)2)/xtopix;
    y0=yaxis-ytopiy*f*Math.exp(-(xstart-mm)*(xstart-mm)/(2*ss*ss));
// 座標軸と Y=1 の線の表示
    g.setColor(Color.white);
    y=yaxis-ytopiy;
    g.drawLine(x0,(int)(y),width,(int)(y));
    g.setColor(Color.black);
    g.setFont(new Font("TimesRoman",Font.PLAIN,10));
    g.drawString(String.valueOf(1d),width/2-10,(int)(y));
// 座標軸最大値の表示
    g.setColor(Color.black);
    g.setFont(new Font("TimesRoman",Font.PLAIN,10));
    g.drawString(String.valueOf(tte),width/2-10,10);
    g.drawString("max",width-18,height-10);
// 密度関数表示
    for (x=-width/2;x<width/2;x+=1){
        realx=(double)x/xtopix;

```

```

y=yaxis-ytopiy*f*Math.exp(-(realx-mm)*(realx-mm)/(2*ss*ss));
if(y >=-10000){
    g.setColor(Color.black);
    g.drawLine(x0,(int)y0,x+width/2,(int)y);
}
//( 確率を求める部分の塗りつぶし )
if(pflag==1){
    if(realx>=realxmin && realx<=realxmax){
        g.setColor(Color.green);
        if(y >= -10000) g.drawLine(x+width/2,(int)yaxis,x+width/2,(int)y);
        else g.drawLine(x+width/2,(int)yaxis,x+width/2,0);
    }
}
x0=x+width/2;
y0=y;
}
// 累積分布関数表示
x0=0;
y0=yaxis;
if(cflag==1){
    for (x=-width/2;x<width/2;x+=1){
        realx=(double)x/xtopix;
        if((realx-mm)<=0)y=yaxis-ytopiy*(0.5-norm((realx-mm)/ss));
        else y=yaxis-ytopiy*(0.5+norm((realx-mm)/ss));
        if(x==width/2)y0=y;
        if (y >= -10000){
            g.setColor(Color.blue);
            g.drawLine(x0,(int)y0,x+width/2,(int)y);
        }
    }
x0=x+width/2;
    y0=y;
}
}
// 座標軸を描く
x0=0;
xstart=((double)x0-(double)width/(double)2)/xtopix;
y0=yaxis-ytopiy*f*Math.exp(-(xstart-mm)*(xstart-mm)/(2*ss*ss));
g.setColor(Color.lightGray);
g.drawLine(x0,(int)yaxis,width,(int)yaxis);
g.drawLine(width/2,0,width/2,height);
// 確率計算と表示
if((pflag==1) && (realxmin<=realxmax)){
    g.setColor(Color.black);
    g.setFont(new Font("TimesRoman",Font.PLAIN,10));
    g.drawString(String.valueOf(realxmin),(int)(realxmin*xtopix)+width/2,height-9);
    g.drawString(String.valueOf(realxmax),(int)(realxmax*xtopix)+width/2,height-9);
    //
    y=atai(realxmin,realxmax,mm,ss);
    g.setFont(new Font("TimesRoman",Font.PLAIN,12));
    g.drawString("P="+String.valueOf(y),20,height*2/3);
}
}
}

```