

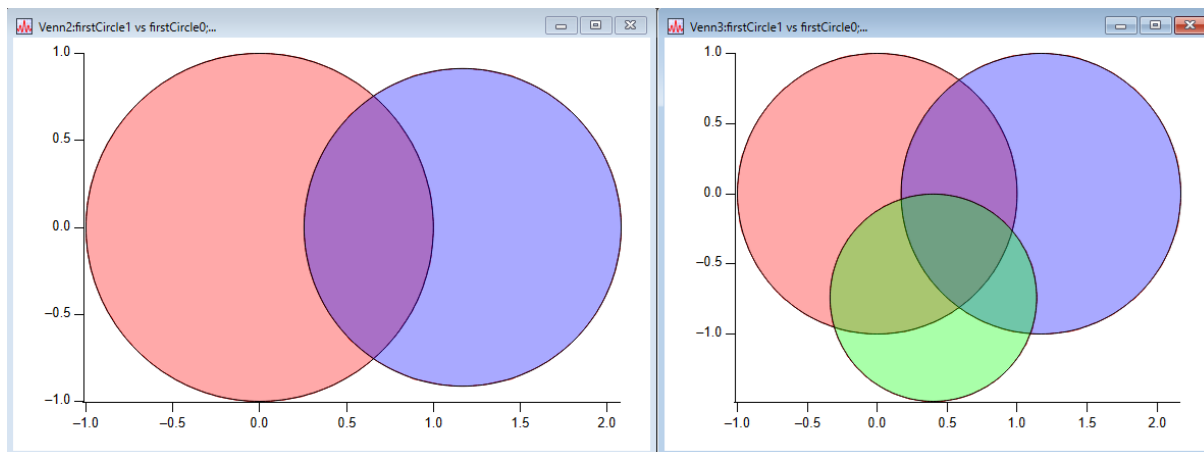
CONTENTS

サンプルの Experiment – Venn Demo	2
デモの操作手順	2
プロセスの内容.....	2

サンプルの Experiment – Venn Demo

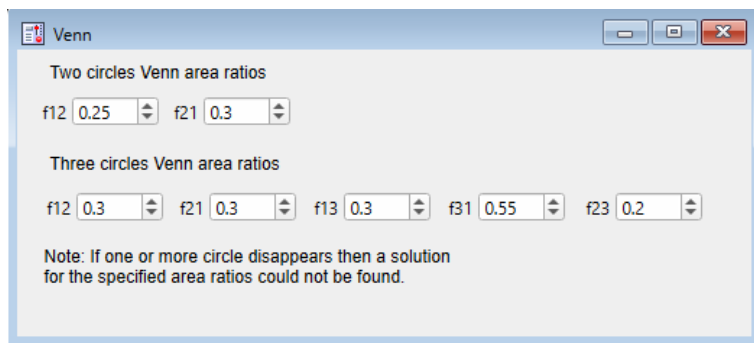
メニュー File → Example Experiments → Graphing Techniques → Venn Diagram Demo

この Experiment では、ベン図の作成方法を説明します。



デモの操作手順

デモ Experiment を開くと次のコントロールパネルが開きます。



このパネルでは、2つの円、3つの円のベン図をコントロールすることができます。

動作の詳細な説明はプロシージャ内にコメントとして書かれています。

プロシージャの内容

```
////////////////////////////////////  
// このファイルには、単純なベン図の計算および表示に使われるプロシージャが含まれています。  
// ここにはおそらく、あなたが読みたいと思うよりも多くの詳細が含まれていますが、それらはコードを  
// 拡張または変更したい方のために記載しています。  
// 詳細に興味がない場合は、以下の「2つの円ベン図」または「3つの円ベン図」に直接スキップできます。  
  
// 基本的な定義：  
// =====
```

```

// プロシージャは、重なり合った円からなる図を表示します。
// 最初の円は原点 (0,0) を中心とし、半径 R1=1 単位です。
// 第 2 の円は、正の x 軸を中心に配置されます。
// 第 3 の円が必要な場合は、x 軸の下に中心を配置します。
// 符号の単純な変更で、この選択が逆転する可能性があることに注意してください。

// 2 つの円からなるベン図から始めて、R1=1 と仮定し、2 つのパラメーター R2 および d
// (C1 と C2 の間の距離) を求めます。
// この計算は、wp に 4 点の数値ウェーブを渡して findD1andAlpha(a1Fraction,a2Fraction,wp)
// を使うと実行できます。

// 3 つの円からなるベン図は、原点を中心に半径 R1=1 の円 C1 を基点として作成されます。
// 第 2 の円 C2 は正の x 軸を中心に配置され、第 3 の円は x 軸の下方に中心を置いています。
// 完全に同等の解決策は、3 番目の円を x 軸の上方に中心を置くことです。
// 2 つの解は、3 番目の円の中心の y 成分の符号の選択のみが異なります。
// 3 番目の円の中心を見つけるには、まず R3 と原点からの距離 C3 を求めます。
// R3 が指定されたら、Optimize を使って 2 番目の円の中心からの距離を計算します。
// 2 つの距離は、x 軸の上または下で交差します。
// x 軸の下にある点を、3 番目の円の中心として選択します。

// 関数 find3CircleSolution(f12,f21,f13,f31,f23) を使って、3 つの円の半径と中心を計算する
// ことができます。
// f32 の解は他の 5 つのパラメーターで完全に指定されるため、f32 を指定する必要はありません。

// ***一部のケースでは、ここで使われているアルゴリズムが解決策を見つけることができない場合があること
// に注意してください。
// それは解が存在しないことを意味するものではありません。
// それは、このパラメーターの選択では、ルートの探索または最適化操作で解を見つけることができなかったこと
// を意味します。

// 2 つの円ベン図：
// =====
// 2 つの分数を使って 2 つの円からなるベン図を表示する：
// f1=intersectionArea/(pi*R1^2)
// f2=intersectionArea/(pi*R2^2)
// そして次を使います： Venn2Graph(f1,f2)

// 3 つの円ベン図：
// =====
// 分数 fij (i と j は円インデックス) を使って、3 つの円からなるベン図を表示する：
// 次を使います： Venn3Graph(f12,f21,f13,f31,f23)

// 3 つ以上の円からなる図を作成する必要がある場合：
// =====
// これはかなり複雑になる可能性があります。
// なぜなら、すべての可能な重なり組み合わせを考慮する必要があるからです。
// この実装の一つの方法は、ここに提供されているコードを使って、最初の 3 つの（完全に重なり合う）円を
// 描画することです。
// 追加する各円（半径 Ri の円 i）に対して、その円と交わる円を 1 つ以上見つけます。

// 新しい円 i と 1 つの円（円 j など）だけが交差する場合、ここで紹介する方法を使って、Ri、Alphai、
// および dij を計算できます。
// Rj と R1=1 の差を正しく考慮してください。
// dij と dji が与えられた場合、これらを組み合わせて円 j の中心を基準点とする弧を定義し、円 i の
// 中心をその弧上にある適切な位置に選択します。この位置は、円 i 以外の何物とも交わらないようにする
// 必要があります。

```

```

// 一方、円 i が 2 つ以上の円と交わる場合、2 つの円（例えば j と k）のみを考慮すれば十分です。

// ここでは、find3CircleSolution() 関数と同じアプローチを使います。
// ただし、j と k の中心を接続する直線を「axis_jk」と呼ぶものとします。
// 一般解は、axis_jk から対称的に等距離にある 2 つの可能な中心を導きます。
// この場合、j と k だけが円 i と交わる 2 つの円である場合、2 つの中心の選択は任意です。
// そうでない場合は、任意の追加の円と正しい交点を与える中心を選択する必要があります。

// ここで計算されていない数量を計算する必要がある場合：
// =====
// 円は、2 つのウェーブのペアを使って表現します。
// これらは、Igor の標準的なコマンドや関数に組み込むことができます。

////////////////////////////////////
// 計算：
////////////////////////////////////

constant kTolerance=0.001

// 次の関数は、円 1 の半径を 1 に正規化した 2 つの円の交点の解を求めます。
Function find2CircleSolution(a1Fraction,a2Fraction,outWave)
    Variable a1Fraction,a2Fraction
    Wave outWave

    if(WaveExists(outWave))
        outWave=Nan // 解が見つからない場合
    endif

    Make/O/D pw={{a1Fraction},{a2Fraction}}
    // 根を見つけるために、複数の推測の配列を作成し、最終結果が実際の根に似ているかどうかを検証
    // します。
    // V_flag は、実際に根が見つかったかどうかを示さないため、ほぼ無意味であることに注意して
    // ください。
    Make/FREE guessx={0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9}
    Variable count=0
    do
        FindRoots/q /X={guessx[count],guessx[count]} fafb, pw
        Wave W_YatRoot
        MatrixOP/FREE/O ff=sum(abs(W_YatRoot))
        if((V_flag==0 && ff[0]<kTolerance) || count==8)
            break
        else
            count+=1
        endif
    while(1)

    if(ff[0]<kTolerance)
        Wave W_root
        Variable d1=W_root[0] // 垂直線から最初の中心までのオフセット
        Variable b=W_root[1] // これは、中心間の距離または (b)
        Variable d2=b-d1 // 垂直線から 2 番目の中心までのオフセット
        Variable R2=sqrt(d2^2+1-d1^2) // 2 番目の半径
        KillWaves/z W_Root,W_YatRoot

        if(WaveExists(outWave)) // 出力ウェーブにパラメーターをロード
            SetDimLabel 0,0,R2, outWave
            outWave[0]=R2

```

```

        SetDimLabel 0,1,b, outWave
        outWave[1]=b
        SetDimLabel 0,2,d1, outWave
        outWave[2]=d1
        SetDimLabel 0,3,d2, outWave
        outWave[3]=d2
    endif
    return 0 // 成功としてマーク
else
    doAlert 0,"Solution could not be found. Please check your parameters."
    return -1 // 失敗としてマーク
endif
End

```

// 3つの円のケースの解を見つけます。

// 再び、 $R_1=1$ と仮定し、まず R_2 、 d_1 、およびアルファの解を求めます。

Function

find3CircleSolution(a12Fraction,a21Fraction,a13Fraction,a31Fraction,a23Fraction)

Variable a12Fraction,a21Fraction,a13Fraction,a31Fraction,a23Fraction

Make/O/N=4/FREE venSolution1

if(find2CircleSolution(a12Fraction,a21Fraction,venSolution1)==0)

Variable R2=venSolution1[%R2]

Variable d12=venSolution1[1] // c1 の中心から c2 までの距離

// 円を定義するウェーブを作成または更新：

compIllustrate2(1,R2,d12)

else

return 0

endif

Make/O/N=4/FREE venSolution2

if(find2CircleSolution(a13Fraction,a31Fraction,venSolution2)==0)

Variable d13=venSolution2[1] // c1 の中心から c3 までの距離

else

return 0

endif

// この時点では、 R_1 と R_3 は分かっていますが、その間の距離は不明です。

make/FREE/N=4 wp23 // FindRoots に対するパラメーターウェーブ

Variable R3=venSolution2[%R2]

Variable d11=venSolution1[%d1]

wp23[0]=a23Fraction

wp23[1]=R2

wp23[2]=R3

// この時点で、 c_2 と c_3 の間の距離を探そうとします。

Optimize/Q/L=(abs(R2-R3)+0.1)/H=(R2+R3)/T=0.001 rootFunc3,wp23

if(V_Flag==0)

Variable d23=V_minloc

Variable xc=(d23^2-d13^2-d12^2)/(-2*d12) // 円3の中心のx座標

Variable yc=-sqrt(d13^2-xc^2) // 中心のy座標；+ または -

compIllustrate3(xc,yc,R3)

else

```

Wave/z thirdCircle
doAlert 0, "Solution for third circle could not be found."
if(waveExists(thirdCircle))
    thirdCircle=nan
endif
endif
End

// ここでは、dのみを求めたいと考えています。
Function rootFunc3(w,d23)
Wave w
Variable d23

Variable r2=w[1]
Variable r3=w[2]
Variable a=(R3^2-R2^2+d23^2)/(2*d23)

Variable res= abs(w[0]-(acos(a/R3)*R3^2+acos((d23-a)/R2)*R2^2-d23*sqrt(R3^2-a^2)))/(pi*r2^2)
return res
End

// 2つの円の交点のルートを求めるユーザー関数です。
// このバージョンは Francis Dalaudier 氏によって書かれました。
// 他の表現方法に比べて、滑らかな表面を表現する時に、ルートの探索器が処理しやすいという利点があります。

Function fafb(pw, ad, ab)
Wave pw,ad,ab // 探索された fab ; a & d (in) ; fa & fb (out)
Variable a = ad[0], d = ad[1] // [-1,1] 内の a ; d >= 0

Variable h,r,ta,tb,b,sa,sb,fa,fb

b = (d - a)
h = sqrt(1 - a^2) // 1^2 = a^2 + h^2
r = sqrt(h^2 + b^2) // r^2 = b^2 + h^2
tb = acos(b/r)
sb = tb*r^2 - b*h // cb からの表面寄与
ta = acos(a/1)
sa = ta*1^2 - a*h // ca からの表面寄与
fa = (sa + sb)/Pi // ca に対する一般的な分率
fb = (sa + sb)/(Pi*r^2) // cb に対する一般的な分率

ab[0] = fa - pw[0] ; ab[1] = fb - pw[1] // fa, fb に対する目標値
End

// 上記の関数と同じアプローチですが、ここでは c2 と c3 の間の距離 d を解くだけです。
Function fd2CirclesIntersectionArea(r2,r3,d)
Variable r2,r3 // 2つの半径
Variable d // d は a と b の和であり、2つの中心間の距離

Variable h,ta,tb,b,sa,sb,a

if(d==0) // 関数が d=0 で検索する場合でも、NaN を回避したいと思います
    d=1e-9
endif
a = r3*(r2^2-r3^2-d^2)/(-2*r3*d) // a=r3*cos(theta1) from cos() の法則
if(a>d)

```

```

        a=d
    endif

    b=d-a
    h = sqrt( r3^2 - a^2)           // r3^2 = a^2 + h^2
    tb = acos(b/r2)
    sb = tb*r2^2 - b*h               // c2 からの表面寄与
    ta = acos(a/r3)
    sa = ta*r3^2 - a*h               // ca からの表面寄与
    return sa+sb
End

////////////////////////////////////
// 表示関係
////////////////////////////////////
constant ptsInCircle=200

// 円を示すウェーブを計算します：
Function compIllustrate2(rad1,rad2,b)
    Variable rad1,rad2,b

    // 最初の円は原点を中心に置かれているものと仮定します。
    // 201 ポイントの 2 列のウェーブを使って表示します：
    Make/O/N=(ptsInCircle+1,2) firstCircle
    Variable i,da=2*pi/ptsInCircle
    for(i=0;i<ptsInCircle;i+=1)
        firstCircle[i][0]=rad1*cos(i*da)
        firstCircle[i][1]=rad1*sin(i*da)
    endfor
    // 円を閉じます：
    firstCircle[ptsInCircle][0]=firstCircle[0][0]
    firstCircle[ptsInCircle][1]=firstCircle[0][1]

    // 2 番目の円は、d1+d2 の距離右側に中心があります：
    Variable cx=b
    Variable cy=0                     // 3 目を追加する準備をします
    Make/O/N=(ptsInCircle+1,2) secondCircle
    for(i=0;i<ptsInCircle;i+=1)
        secondCircle[i][0]=cx+rad2*cos(i*da)
        secondCircle[i][1]=cy+rad2*sin(i*da)
    endfor
    // 円を閉じます：
    secondCircle[ptsInCircle][0]=secondCircle[0][0]
    secondCircle[ptsInCircle][1]=secondCircle[0][1]
End

// 円を示すウェーブを計算します：
Function compIllustrate(rad1,rad2,d1)
    Variable rad1,rad2,d1

    Variable aa2=(rad2/rad1)^2         // アルファの二乗
    Variable bb2=(d1/rad1)^2
    Variable d2=rad1*sqrt(aa2-1+bb2)

    // 最初の円は原点を中心に置かれているものと仮定します。
    // 201 ポイントの 2 列のウェーブを使って表示します：

```

```

Make/O/N=(ptsInCircle+1,2) firstCircle
Variable i,da=2*pi/ptsInCircle
for(i=0;i<ptsInCircle;i+=1)
    firstCircle[i][0]=rad1*cos(i*da)
    firstCircle[i][1]=rad1*sin(i*da)
endfor
// 円を閉じます :
firstCircle[ptsInCircle][0]=firstCircle[0][0]
firstCircle[ptsInCircle][1]=firstCircle[0][1]

// 2 番目の円は、d1+d2 の距離右側に中心があります :
Variable cx=d1+d2
Variable cy=0 // 3 つ目を追加する準備をします
Make/O/N=(ptsInCircle+1,2) secondCircle
for(i=0;i<ptsInCircle;i+=1)
    secondCircle[i][0]=cx+rad2*cos(i*da)
    secondCircle[i][1]=cy+rad2*sin(i*da)
endfor
// 円を閉じます :
secondCircle[ptsInCircle][0]=secondCircle[0][0]
secondCircle[ptsInCircle][1]=secondCircle[0][1]

End

// 3 番目の円を計算して作成します :
Function compIllustrate3(xc,yc,rad3)
    Variable xc,yc,rad3

    Variable i,da=2*pi/ptsInCircle
    Make/O/N=(ptsInCircle+1,2) thirdCircle
    for(i=0;i<ptsInCircle;i+=1)
        thirdCircle[i][0]=xc+rad3*cos(i*da)
        thirdCircle[i][1]=yc+rad3*sin(i*da)
    endfor
    // 円を閉じます :
    thirdCircle[ptsInCircle][0]=thirdCircle[0][0]
    thirdCircle[ptsInCircle][1]=thirdCircle[0][1]

End

Function fillCircle(xwave,ywave,rr,gg,bb,aa,gname)
    wave xwave,ywave
    String gname
    Variable rr,gg,bb,aa

    SetDrawLayer/W=$gname ProgFront
    SetDrawEnv/W=$gname xcoord=bottom,ycoord=left,fillfgc=(rr,gg,bb,aa),fillpat=1
    DrawPoly/W=$gname xwave[0],ywave[0], 1, 1, xwave,ywave

End

Function drawVenn2(x1wave,y1wave,x2wave,y2wave,rr1,gg1,bb1,aa1,rr2,gg2,bb2,aa2)
    Wave x1wave,y1wave,x2wave,y2wave
    Variable rr1,gg1,bb1,aa1,rr2,gg2,bb2,aa2

    if(wintype("venn2")==0)
        Display/N=Venn2 y1wave vs x1wave
    end
end

```

```

        ModifyGraph width={Plan,1,bottom,left}
        AppendToGraph y2wave vs x2wave
    endif

    SetDrawLayer/w=venn2/k ProgFront
    fillCircle(x1wave,y1wave,rr1,gg1,bb1,aa1,"venn2")
    fillCircle(x2wave,y2wave,rr2,gg2,bb2,aa2,"venn2")
End

Function
drawVenn3(x1wave,y1wave,x2wave,y2wave,x3wave,y3wave,rr1,gg1,bb1,aa1,rr2,gg2,bb2,aa2,rr3,gg3,bb3,aa3)
    Wave x1wave,y1wave,x2wave,y2wave,x3wave,y3wave
    Variable rr1,gg1,bb1,aa1,rr2,gg2,bb2,aa2,rr3,gg3,bb3,aa3

    if(wintype("venn3")==0)
        Display/N=Venn3 y1wave vs x1wave
        ModifyGraph width={Plan,1,bottom,left}
        AppendToGraph y2wave vs x2wave
        AppendToGraph y3wave vs x3wave
    endif

    SetDrawLayer/w=venn3/k ProgFront
    fillCircle(x1wave,y1wave,rr1,gg1,bb1,aa1,"venn3")
    fillCircle(x2wave,y2wave,rr2,gg2,bb2,aa2,"venn3")
    fillCircle(x3wave,y3wave,rr3,gg3,bb3,aa3,"venn3")
End

```

// 2つの円からなるベン図を表示し、交差する領域がそれぞれ全体の円面積の分数 f1 と f2 を表すようにします。

```

Function Venn2Graph(f1,f2)
    Variable f1,f2

    String curDF=GetDataFolder(1)
    SetDataFolder root:
    NewDataFolder/O/S Venn2
    Make/O/N=4 ven2Solution
    if(find2CircleSolution(f1,f2,ven2Solution)==0)
        Variable R2=ven2Solution[%R2]
        Variable b=ven2Solution[%b]
        // 円を定義するウェーブを作成または更新します：
        compIllustrate2(1,R2,b)
    endif

    Wave/Z firstCircle,secondCircle // these are 2 2D waves.
    if(WaveExists(firstCircle) && WaveExists(secondCircle))
        splitWave/O/Name="firstCircle0;firstCircle1;" firstCircle
        splitWave/O/Name="secondCircle0;secondCircle1;" secondCircle
        Wave firstCircle0,firstCircle1,secondCircle0,secondCircle1
        // 以下の2つの RGBA 値を編集して、異なる色を使用できます：

        drawVenn2(firstCircle0,firstCircle1,secondCircle0,secondCircle1,65535,0,0,22000,
0,0,65535,22000)
    else
        Wave/Z firstCircle0,secondCircle0
        if(WaveExists(firstCircle0))
            firstCircle0=nan
        endif
    endif

```

```

        if(WaveExists(secondCircle0))
            secondCircle0=nan
        endif
    endif
    KillWaves/z firstCircle,secondCircle
    SetDataFolder curDF
End

```

// 3つの円からなるベン図を表示し、交差する領域がそれぞれ全体の円面積の分数 f1 と f2 を表すようにします。

```

Function Venn3Graph(f12,f21,f13,f31,f23)
    Variable f12,f21,f13,f31,f23

    String curDF=GetDataFolder(1)
    SetDataFolder root:
    NewDataFolder/O/S Venn3
    find3CircleSolution(f12,f21,f13,f31,f23)
    Wave firstCircle,secondCircle,thirdCircle // 3つの2Dウェーブです。
    if(WaveExists(firstCircle) && WaveExists(secondCircle) &&
WaveExists(thirdCircle))
        splitWave/O/Name="firstCircle0;firstCircle1;" firstCircle
        splitWave/O/Name="secondCircle0;secondCircle1;" secondCircle
        splitWave/O/Name="thirdCircle0;thirdCircle1;" thirdCircle
        Wave
        firstCircle0,firstCircle1,secondCircle0,secondCircle1,thirdCircle0,thirdCircle1

        drawVenn3(firstCircle0,firstCircle1,secondCircle0,secondCircle1,thirdCircle0,thi
rdCircle1,65535,0,0,22000,0,0,65535,22000,0,65535,0,22000)
    else
        Wave/Z firstCircle0,secondCircle0,thirdCircle0
        if(WaveExists(firstCircle0))
            firstCircle0=nan
        endif
        if(WaveExists(secondCircle0))
            secondCircle0=nan
        endif
        if(WaveExists(thirdCircle0))
            thirdCircle0=nan
        endif
    endif
    KillWaves/z firstCircle,secondCircle,thirdCircle
    SetDataFolder curDF
End

```

```

Function twoCirclesSetVarProc(sva) : SetVariableControl
    STRUCT WMSetVariableAction &sva

    switch( sva.eventCode )
        case 1: // マウスを離す
        case 2: // Enter キー
        case 3: // ライブアップデート
            ControlInfo/W=venn f12setvar
            Variable f12=V_value
    endswitch

```

```

        ControlInfo/W=venn f21setvar
        Variable f21=V_value
        Venn2Graph(f12,f21)
        break
    case -1:      // コントロールが Kill される
        break
endswitch

return 0

End

Function threeCircleSetVarProc(sva) : SetVariableControl
    STRUCT WMSetVariableAction &sva

    switch( sva.eventCode )
        case 1:      // マウスを離す
        case 2:      // Enter キー
        case 3:      // ライブアップデート
            ControlInfo/W=venn f12setvar1
            Variable f12=V_value
            ControlInfo/W=venn f21setvar1
            Variable f21=V_value
            ControlInfo/W=venn f13setvar1
            Variable f13=V_value
            ControlInfo/W=venn f31setvar1
            Variable f31=V_value
            ControlInfo/W=venn f23setvar1
            Variable f23=V_value

            Venn3Graph(f12,f21,f13,f31,f23)
            break
        case -1:      // コントロールが Kill される
            break
    endswitch

    return 0

End

Window Venn() : Panel
    PauseUpdate; Silent 1      // ウィンドウを構築
    NewPanel /W=(504,258,935,462) as "Venn"
    SetDrawLayer UserBack
    DrawText 22,24,"Two circles Venn area ratios"
    DrawText 22,87,"Three circles Venn area ratios"
    DrawText 18,164,"Note: If one or more circle disappears then there is no
solution ¥rfor the specified area ratios."
    SetVariable
f12SetVar,pos={17.00,33.00},size={81.00,18.00},bodyWidth=60,proc=twoCirclesSetVarProc,t
itle="f12"
    SetVariable f12SetVar,fSize=12,limits={0,1,0.05},value= _NUM:0.3
    SetVariable
f21SetVar,pos={108.00,33.00},size={81.00,18.00},bodyWidth=60,proc=twoCirclesSetVarProc,
title="f21"
    SetVariable f21SetVar,fSize=12,limits={0,1,0.05},value= _NUM:0.3

```

```

    SetVariable
f12SetVar1,pos={20.00,99.00},size={81.00,18.00},bodyWidth=60,proc=threeCircleSetVarProc
,title="f12"
    SetVariable f12SetVar1,fSize=12,limits={0,1,0.05},value= _NUM:0.3
    SetVariable
f21SetVar1,pos={111.00,99.00},size={81.00,18.00},bodyWidth=60,proc=threeCircleSetVarProc
,title="f21"
    SetVariable f21SetVar1,fSize=12,limits={0,1,0.05},value= _NUM:0.3
    SetVariable
f13SetVar1,pos={202.00,99.00},size={81.00,18.00},bodyWidth=60,proc=threeCircleSetVarProc
,title="f13"
    SetVariable f13SetVar1,fSize=12,limits={0,1,0.05},value= _NUM:0.3
    SetVariable
f23SetVar1,pos={291.00,99.00},size={83.00,18.00},bodyWidth=60,proc=threeCircleSetVarProc
,title="f23"
    SetVariable f23SetVar1,fSize=12,limits={0,1,0.05},value= _NUM:0.3
    Execute/Q/Z "SetWindow kwTopWin sizeLimit={46,109,inf,inf}"
    // sizeLimit は Igor 7 以降が必要
EndMacro

////////////////////////////////////

```