Illustrate the Use of Continouse Distributions

by Nasser Abbasi

```
Manipulate[distribution[], {{distribution, doExpDensity, "select distribution"},
  {doExpDensity -> "Exponential Density",
   doGammaDensityScale -> Style["Gamma Density (scale model)", FontSize → 9],
   doGammaDensityRate -> Style["Gamma Density (rate model)", FontSize → 9],
   doNormalDistribution -> "Normal Distribution",
   doUniformCont -> "Uniform Distribution",
   doBetaDensity -> "Beta Density",
   doCauchy -> "Cauchy",
   doStudentT -> "StudentT",
   doChiSquare -> "ChiSquare",
   doChi -> "Chi"}, ControlType → PopupMenu,
  ControlPlacement → Top}, AutorunSequencing → {6}, Initialization :→
  {
   Get["BarCharts`"];

   gBernoulli = 0;
   gBinomial = 1;
   gGeometric = 2;
   gNegativeBinomial = 3;
   gHyperGeometric = 4;
   gPoisson = 5;
   gExpDensity = 6;
   gDebug = False;

   Needs["Histograms`"];
   Needs["BarCharts`"];


   (************************************)
   (*   processAllRandomFucntions     *)
   (***********************************)
   processAllRandomFucntions[firstPDF_, firstPDFpars_, thefunc_, thefuncPars_, xFrom_, xTo_,
     yFrom_, yTo_] := Module[{func, funcPars, mapping, allSolutions, nSolutions,
      yPDFsymbolic, yPDFnumeric, xPDFnumeric, p1, p2, title, z1, z2, label, tmp},

     If[gDebug, Print["xFrom=", xFrom, " xTo=", xTo, " yFrom=", yFrom, " yTo=", yTo]];

     func = ReleaseHold[thefunc];
     funcPars = ReleaseHold[thefuncPars];
     mapping = Y == func;
```

```
Quiet[Check[allSolutions = X /. Solve[mapping, X],
  Return[Text["Unable to find Inverse for function provided!"]]
 ]
];

If[gDebug, Print["allSolutions=", allSolutions]];
nSolutions = Length[allSolutions];
If[nSolutions > 2, Return[Text["Do not know yet how to do equation
     with more than 2 solutions...maybe next version.."]]];

yPDFsymbolic = 0;
yPDFnumeric = 0;

Which[nSolutions == 1,
 {z1 = (allSolutions /. {Y → y})[[1]];
  yPDFsymbolic = (Abs[D[z1, y]] * PDF[firstPDF, x]) /. {x → z1};
  yPDFnumeric = yPDFsymbolic /. firstPDFpars /. funcPars;
  If[gDebug, Print["yPDFnumeric", yPDFnumeric]];
 },
 nSolutions == 2,
 {z1 = allSolutions[[1]] /. {Y → y};
  z2 = allSolutions[[2]] /. {Y → y};
  If[gDebug, Print["z1=", z1, "z2=", z2]];
  tmp =  (PDF[firstPDF, x] /. {x → z1}) + (PDF[firstPDF, x] /. {x → z2});
  yPDFsymbolic = Abs[D[z1, y]] * tmp;
  If[gDebug, Print["yPDFsymbolic=", yPDFsymbolic]];
  yPDFnumeric = yPDFsymbolic /. firstPDFpars /. funcPars;
  If[gDebug, Print[yPDFnumeric]];
 }
];

xPDFnumeric = PDF[firstPDF, x] /. firstPDFpars;

p1 = Plot[ xPDFnumeric
  , {x, xFrom, xTo},
  PlotRange → {All, {yFrom, yTo}},
  PlotStyle → Red
 ];

title = Grid[
  {
   {"", "Symbolic", "Numeric"},
   {Style["X", Red],
    Style[Text[PDF[firstPDF, x]], Red], Text[Style[xPDFnumeric, Red]]},
   {"Y", Text[yPDFsymbolic], Text[yPDFnumeric]}
  },
  Frame → All,
  Spacings → 1,
  ItemSize → {Full, Full}
 ];

p2 = Plot[ yPDFnumeric, {y, xFrom, xTo}, PlotRange → All];

label = Row[{Style["X", Red], Style["->Y", Black]}];
```

```
  Return[Grid[{{title}, {Show[{p1, p2}, PlotLabel → label, ImageSize → {500, 300}]}}]]
 ];

(*************************************)
(*   Gama    for funnctions of R.V.   *)
(*************************************)
processGammaDensityRate[lambda_,
  alpha_, thefunc_, thefuncPars_, width_, yaxis_] := Module[
  {firstPDF, firstPDFpars, xFrom, xTo, yFrom, yTo},

  firstPDF = GammaDistribution[λ, α];
  firstPDFpars = {λ → lambda, α → alpha};

  yTo = .3;
  yTo = N[yTo + yaxis];
  yFrom = 0;

  xTo = 60;
  xTo = N[xTo + width];
  xFrom = 0;

  processAllRandomFucntions[firstPDF,
   firstPDFpars, thefunc, thefuncPars, xFrom, xTo, yFrom, yTo]
 ];

(*************************************)
(*   X=Uniform R.V.                  *)
(*************************************)
processUniformCont[themin_, themax_, thefunc_, thefuncPars_, width_, yaxis_] := Module[
  {firstPDF, firstPDFpars, xFrom, xTo, yFrom, yTo},

  firstPDF = UniformDistribution[{min, max}];
  firstPDFpars = {min → themin, max → themax};

  yTo = 1.5;
  yTo = N[yTo + yaxis];
  yFrom = 0;

  xTo = themax;
  xTo = N[xTo + width];
  xFrom = themin;

  processAllRandomFucntions[firstPDF,
   firstPDFpars, thefunc, thefuncPars, xFrom, xTo, yFrom, yTo]
 ];
(*************************************)
(*   Exp  for funnctions of R.V.      *)
(*************************************)
processExp[lambda_, thefunc_, thefuncPars_, width_, yaxis_] := Module[
  {firstPDF, firstPDFpars, xFrom, xTo, yFrom, yTo},

  firstPDF = ExponentialDistribution[λ];
  firstPDFpars = {λ → lambda};
```

```
  yTo = 5;
  yTo = N[yTo + yaxis];
  yFrom = 0;

  xTo = 5;
  xTo = N[xTo + width];
  xFrom = 0;

  processAllRandomFucntions[firstPDF,
   firstPDFpars, thefunc, thefuncPars, xFrom, xTo, yFrom, yTo]
 ];
(*************************************)
(*   Normal  for funnctions of R.V.   *)
(*************************************)
processNormal[mean_, std_, thefunc_, thefuncPars_, centerAround_, width_, yaxis_] :=
 Module[{firstPDF, firstPDFpars, xFrom, xTo, yFrom, yTo},

  firstPDF = NormalDistribution[μ, σ];
  firstPDFpars = {μ → mean, σ → std};

  yTo = PDF[firstPDF, x] /. firstPDFpars /. x → mean;
  yTo = yTo + .1 * yTo;
  yTo = N[yTo + yaxis];
  yFrom = 0;

  xTo = mean + 10 std;
  xTo = N[xTo + width];
  xFrom = mean - 6 std;
  xFrom = N[xFrom - width];

  processAllRandomFucntions[firstPDF,
   firstPDFpars, thefunc, thefuncPars, xFrom, xTo, yFrom, yTo]
 ];

(****************************************)
(*         style function               *)
(****************************************)
st[text_, fontSize_?NumberQ] := Style[text, FontSize → fontSize];


(****************************************)
(*         Start of continouse R.V.     *)
(*          Exponential Density         *)
(****************************************)
expDensity[λ_, maxT_] := Module[{mean, var, median, title, p1, p2, t},
  If[gDebug, Print["Enter expDensity[]"]];

  mean = Mean[ExponentialDistribution[λ]];
  var = Variance[ExponentialDistribution[λ]];
  median = N[Log[2] / λ];

  title = "f(t)=λe^(-λt)   Mean:1/λ  =" <> ToString[mean] <>
```

```
    "\nVariance:  1/λ² =" <> ToString[var] <> "   Median: Ln[2]/λ =" <> ToString[median];

  p1 = Plot[PDF[ExponentialDistribution[λ], t], {t, 0, maxT},
    Frame → True,
    FrameLabel → {{st["f(t)", 10], None}, {None, st[title, 10]}},
    ImageSize → {250},
    PlotRange → All
   ];

  title = "\nCDF\n1-e^-λt";
  p2 = Plot[CDF[ExponentialDistribution[λ], t], {t, 0, maxT},
    Frame → True,
    FrameLabel → {{st["F(t)", 10], None}, {None, st[title, 10]}},
    ImageSize → {250},
    PlotRange → All
   ];

  Labeled[Grid[{{p1, p2}},
    Alignment → {Top, Top},
    Frame → None,
    Spacings → 2,
    ItemSize → {Full, Full}
   ], st["Special case of Gamma density when Shape parameter is 1
      and scale parameter the mean interval between arrivals", 10]
  ]

 ];

doExpDensity[] := Module[{},
  If[gDebug, Print["Enter doExpDensity[]"]];
  Manipulate[expDensity[λ, If[maxT == 0, maxT = 0.001, maxT]],
   {{λ, .8, "Failure rate? Failure per unit time? λ="},
    0.01, 10, .01, Appearance → "Labeled"},
   {{maxT, 5, "[display] adjust time scale"}, 0.0, 100, .1, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];
(****************************************)
(*           Gamma Density scale        *)
(*  α=SHAPE, β=scale                    *)
(****************************************)
gammaDensityScale[β_, α_, maxT_] := Module[{mean, var, median, title, p1, p2, h},
  If[gDebug, Print["Enter gammaDensity[]"]];

  mean = Mean[GammaDistribution[α, β]];
  var = Variance[GammaDistribution[α, β]];

  title = "g(t) = 1/(β^α Γ(α)) t^(α-1) e^(-t/β)   t≥ 0\nMean:α β  =" <>
    ToString[mean] <> "   Variance:  α β² =" <> ToString[var];

  p1 = Plot[PDF[GammaDistribution[α, β], t], {t, 0.0001, maxT},
```

```
    Frame → True,
    FrameLabel → {{st["g(t)", 10], None}, {None, st[title, 10]}},
    ImageSize → {250},
    PlotRange → All
   ];

  title = "\nCDF(t)\t Γ[α, 0, t/β] / Γ[σ] ";
  p2 = Plot[CDF[GammaDistribution[α, β], t], {t, 0, maxT},
    ImageSize → {250},
    Frame → True,
    FrameLabel → {{st["F(t)", 10], None}, {None, st[title, 10]}},
    PlotRange → All];

  Labeled[Grid[{{p1, p2}},
    Alignment → {Top, Top},
    Frame → None,
    Spacings → 2,
    ItemSize → {Full, Full}
   ], st[
    "Reprenent the probability of the α^th arrival when arrivals are produced by Poisson
      process and arrival rate with mean λ. Used in queuing models, flow of
      items through manufacturing, load on web servers,telecom exchange", 10]
  ]
 ];

doGammaDensityScale[] := Module[{},
  If[gDebug, Print["Enter GammaDensity[]"]];
  Manipulate[If[β == 0, β = 0.0001]; If[α == 0, α = 0.0001];
   gammaDensityScale[β, α, If[maxT == 0, maxT = 0.001, maxT]],
   {{β, 1, "Scale parameter β="}, 0, 10, .1, Appearance → "Labeled"},
   {{α, 5, "Shape parameter α="}, 0, 10, .1, Appearance → "Labeled"},
   {{maxT, 20, "[display] adjust x-axis (operating time)"},
    0.0, 500, .1, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];

(****************************************)
(*           Gamma Density rate         *)
(*  α=SHAPE, λ=rate                     *)
(****************************************)
gammaDensityRate[λ_, α_, maxT_] := Module[{mean, var, median, title, p1, p2, h},
  If[gDebug, Print["Enter gammaDensity[]"]];

  mean = Mean[GammaDistribution[α, 1 / λ]];
  var = Variance[GammaDistribution[α, 1 / λ]];

  title = "g(t) = λ^α/Γ(α) t^(α-1) e^(-λt)   t ≥ 0\nMean:α 1/λ  =" <>
    ToString[mean] <> "  Variance: α (1/λ)^2 =" <> ToString[var];
```

```
  p1 = Plot[PDF[GammaDistribution[α, 1 / λ], t], {t, 0.0001, maxT},
    Frame → True,
    FrameLabel → {{st["g(t)", 10], None}, {None, st[title, 10]}},
    ImageSize → {250},
    PlotRange → All
   ];

  title = "\nCDF(t)\t Γ[α, 0, λt] / Γ[σ]";

  p2 = Plot[CDF[GammaDistribution[α, 1 / λ], t], {t, 0, maxT},
    ImageSize → {250},
    Frame → True,
    FrameLabel → {{st["F(t)", 10], None}, {None, st[title, 10]}},
    PlotRange → All];

  Labeled[Grid[{{p1, p2}},
    Alignment → {Top, Top},
    Frame → None,
    Spacings → 2,
    ItemSize → {Full, Full}
   ], st[
    "Reprenent the probability of the α^th arrival when arrivals are produced by Poisson
      process and arrival rate with mean λ. Used in queuing models, flow of
      items through manufacturing, load on web servers,telecom exchange", 10]
  ]

 ];

doGammaDensityRate[] := Module[{},
  If[gDebug, Print["Enter doExpDensity[]"]];
  Manipulate[If[λ == 0, λ = 0.0001]; If[α == 0, α = 0.0001];
   gammaDensityRate[λ, α, If[maxT == 0, maxT = 0.001, maxT]],
   {{λ, 1, "rate parameter λ="}, 0, 10, .1, Appearance → "Labeled"},
   {{α, 5, "Shape parameter α="}, 0, 10, .1, Appearance → "Labeled"},
   {{maxT, 20, "[display] adjust x-axis (operating time)"},
    0.0, 500, .1, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];


(****************************************)
(*           NormalDistribution         *)
(****************************************)
normalDistribution[μ_, var_] := Module[{maxX, x, title, p1, p2, pts, h, σ},
  If[gDebug, Print["Enter normalDistribution[]"]];
  σ = Sqrt[var];
  maxX = 10;
```

```
title = Grid[{{"f(x) = 1/(σ√(2π)) e^(-(x-μ)^2/2σ^2)   -∞≤x≤∞"},
    {Grid[{{"Mean μ=" <> ToString[μ], "Median=" <> ToString[μ]},
       {"Mode=" <> ToString[μ], "Variance σ^2=" <> ToString[var]}}]}},
   Alignment → Left,
   Frame → None,
   Spacings → 1,
   ItemSize → {Full}];

p1 = Plot[PDF[NormalDistribution[μ, σ], x], {x, -3 * σ , μ + 3 σ},
   Frame → True,
   FrameLabel → (Style[#, FontSize → 12] & /@ {"x", "f(x)", title}),
   ImageSize → {250},
   PlotRange → All
  ];

p2 = Plot[CDF[NormalDistribution[μ, σ], x], {x, -3 * σ , μ + 3 σ},
   ImageSize → {250},
   Frame → True,
   FrameLabel → (Style[#, FontSize → 12] & /@
      {"x", "F(x)", Text["\nCDF(x)\t 1/2 (1+Erf[(x - μ)/(Sqrt[2] σ)])\n"]}),
   PlotRange → All];

Grid[{{p1, p2}},
  Alignment → {Top, Top},
  Frame → None,
  Spacings → 2,
  ItemSize → {Full, Full}
 ]
];

doNormalDistribution[] := Module[{},
  If[gDebug, Print["Enter doExpDensity[]"]];
  Manipulate[normalDistribution[μ, var],
   {{μ, 0, "Mean  μ="}, 0, 100, .1, Appearance → "Labeled"},
   {{var, .5, "Variance σ^2="}, 0.01, 100, .05, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];

(****************************************)
(*           Beta Density                *)
(****************************************)
betaDensity[α_, β_] := Module[{title, p1, p2, mean, var, x},
  If[gDebug, Print["Enter betaDensity[]"]];
  mean = N[Mean[BetaDistribution[α, β]]];
  var = N[Variance[BetaDistribution[α, β]]];
```

```
  title = "f(x) = ((1-x)^(-1+β) x^(-1+α))/Beta[α,β]   0≤x≤1\nMean: α/(α + β) =" <>
    ToString[mean] <> "  Var: (α β)/((α + β)^2 (1 + α + β)) =" <> ToString[var];

  p1 = Plot[PDF[BetaDistribution[α, β], x], {x, 0, 1},
    Frame → True,
    FrameLabel → {{st["f(x)", 10], None}, {None, st[title, 10]}},
    ImageSize → {250},
    PlotRange → All
   ];

  title = Text["\nCDF(x)\n\nBetaRegularized[x,α,β]"];
  p2 = Plot[CDF[BetaDistribution[α, β], x], {x, 0, 1},
    ImageSize → {250},
    Frame → True,
    FrameLabel → {{st["F(x)", 10], None}, {None, st[title, 10]}},
    PlotRange → All];

  Labeled[Grid[{{p1, p2}},
    Alignment → {Top, Top},
    Frame → None,
    Spacings → 2,
    ItemSize → {Full, Full}
   ], st[
    "Verstile distribution over finite field. Used to represent quantities whose values
      are restricted to specific interval. Area of applications: Tolerance
      limits, quality control, reliability ", 10]
  ]

 ];

doBetaDensity[] := Module[{},
  If[gDebug, Print["Enter doExpDensity[]"]];
  Manipulate[betaDensity[If[α == 0, α = 10^-6, α], If[β == 0, β = 10^-6, β]],
   {{α, 6, "α="}, 0, 100, .05, Appearance → "Labeled"},
   {{β, 2, "β="}, 0, 100, .05, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];

(****************************************)
(*        Cauchy                        *)
(****************************************)
cauchy[a_, b_, maxX_] := Module[{title, p1, p2, x},
  If[gDebug, Print["Enter betaDensity[]"]];
  title = "f(x) is propertional to (1+(x-a)^2/b^2)^-1   \nMean: Indeterminate" <>
    "  Var: Indeterminate=";
  p1 = Plot[PDF[CauchyDistribution[a, b], x], {x, -maxX, maxX},
    Frame → True,
```

```
    FrameLabel → (Style[#, FontSize → 10] & /@ {"x", "f(x)", title}),
    ImageSize → {250},
    PlotRange → All
   ];

  p2 = Plot[CDF[CauchyDistribution[a, b], x], {x, -maxX, maxX},
    ImageSize → {250},
    Frame → True,
    FrameLabel →
     (Style[#, FontSize → 10] & /@ {"x", "F(x)", Text["CDF(x)\n1/2 + ArcTan[(-a+x)/b]/π"]}),
    PlotRange → All];

  Grid[{{p1, p2}},
   Alignment → {Top, Top},
   Frame → None,
   Spacings → 2,
   ItemSize → {Full, Full}
  ]
 ];

doCauchy[] := Module[{},
  If[gDebug, Print["Enter doCauchy[]"]];
  Manipulate[cauchy[a, If[b == 0, b = 10^-6, b], maxX],
   {{a, 0, "Location parameter a="}, -1000, 1000, .1, Appearance → "Labeled"},
   {{b, 20, "Scale parameter b="}, 0, 100, .05, Appearance → "Labeled"},
   {{maxX, 200, "[display] control max x range="}, 0, 5000, 1, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];

(*****************************************)
(*          Student T                   *)
(*****************************************)
student[n_] := Module[{title, p1, p2, x, maxX = 6, mean, var},
  If[gDebug, Print["Enter student[]"]];
  mean = Mean[StudentTDistribution[n]];
  var = Variance[StudentTDistribution[n]];

  title = Text["f(x)  (n/(x^2+n))^((1+n)/2)/(Sqrt[n] Beta[n/2, 1/2])  \nMean: (0 If n>1) =" <>
     ToString[mean] <> "  Var: (n/(n-2) if n>2) =" <> ToString[var]];

  p1 = Plot[PDF[StudentTDistribution[n], x], {x, -maxX, maxX},
    Frame → True,
    FrameLabel → (Style[#, FontSize → 8] & /@ {"x", "f(x)", title}),
    ImageSize → {250},
    PlotRange → All
   ];
```

```
  p2 = Plot[CDF[StudentTDistribution[n], x], {x, -maxX, maxX},
    ImageSize → {250},
    Frame → True,
    FrameLabel → (Style[#, FontSize → 10] & /@ {"x", "F(x)",
        Text["\nCDF(x)\n\n1/2 (1+BetaRegularized[n/(n + x^2),1,n/2,1/2] Sign[x])"]}),
    PlotRange → All];

  Grid[{{p1, p2}},
   Alignment → {Top, Top},
   Frame → None,
   Spacings → 2,
   ItemSize → {Full, Full}
  ]
 ];

doStudentT[] := Module[{},
  If[gDebug, Print["Enter doStudentT[]"]];
  Manipulate[If[n == 0, n = N[10^-6]]; student[n],
   {{n, 1, "number of degress of freedom (positive real number) n="},
    0, 15, .001, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];
(****************************************)
(*          chi square                  *)
(****************************************)
chisquare[n_, maxT_] := Module[{title, p1, p2, t, mean, var},
  If[gDebug, Print["Enter ChiSquare[]"]];
  mean = Mean[ChiSquareDistribution[n]];
  var = Variance[ChiSquareDistribution[n]];

  title = Text[
    "f(t) (2^(-n/2) e^(-t/2) t^(-1+n/2))/Gamma[n/2]  \nMean: n=" <> ToString[mean] "  Var: 2 n=" <> ToString[var]];

  p1 = Plot[PDF[ChiSquareDistribution[n], t], {t, -maxT, maxT},
    Frame → True,
    FrameLabel → {{st["f(t)", 10], None}, {None, st[title, 10]}},
    ImageSize → {250},
    PlotRange → All
   ];

  title = Text["\nCDF(t)\n\nGammaRegularized[n/2,0,t/2]"];
  p2 = Plot[CDF[ChiSquareDistribution[n], t], {t, -maxT, maxT},
    ImageSize → {250},
    Frame → True,
```

```
    FrameLabel → {{st["F(t)", 10], None}, {None, st[title, 10]}},
    PlotRange → All];

  Labeled[Grid[{{p1, p2}},
    Alignment → {Top, Top},
    Frame → None,
    Spacings → 2,
    ItemSize → {Full, Full}
   ],
   st["Special case of a Gamma distribution when shape parameter set to the degrees of
      freedom divided by 2 and scale parameter is set to 2.", 10]
  ]
 ];

doChiSquare[] := Module[{},
  If[gDebug, Print["Enter doChiSquare[]"]];
  Manipulate[If[n == 0, n = N[10^-6]]; chisquare[n, maxT],
   {{n, 5, "number of degress of freedom (positive real number) n="}, 0, 100, .001,
    Appearance → "Labeled"},
   {{maxT, 10, "[display] control max t range="}, 0, 1000, 1, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];
(*****************************************)
(*          Chi                         *)
(*****************************************)
chi[n_, maxX_] := Module[{title, p1, p2, x, mean, var},
  If[gDebug, Print["Enter Chi[]"]];
  mean = N[Mean[ChiDistribution[n]]];
  var = N[Variance[ChiDistribution[n]]];
```

title = Text["\t\tf(x) = $\frac{2^{1-\frac{\nu}{2}} e^{-\frac{x^2}{2}} x^{-1+\nu}}{\text{Gamma}\left[\frac{\nu}{2}\right]}$\nMean $\frac{\sqrt{2}\,\text{Gamma}\left[\frac{1+\nu}{2}\right]}{\text{Gamma}\left[\frac{\nu}{2}\right]}$=" <>

ToString[mean] <> "   Var 2 $\left(-\frac{\text{Gamma}\left[\frac{1+\nu}{2}\right]^2}{\text{Gamma}\left[\frac{\nu}{2}\right]^2}+\frac{\text{Gamma}\left[\frac{2+\nu}{2}\right]}{\text{Gamma}\left[\frac{\nu}{2}\right]}\right)$=" <> ToString[var]];

```
  p1 = Plot[PDF[ChiDistribution[n], x], {x, -maxX, maxX},
    Frame → True,
    FrameLabel → (Style[#, FontSize → 8] & /@ {"x", "f(x)", title}),
    ImageSize → {300},
    PlotRange → All
   ];

  p2 = Plot[CDF[ChiDistribution[n], x], {x, -maxX, maxX},
    ImageSize → {250},
    Frame → True,
    FrameLabel → (Style[#, FontSize → 10] & /@
```

```
        {"x", "F(x)", Text["\nCDF\n\nGammaRegularized[n/2,0,x^2/2]"]}),
    PlotRange → All];

  Grid[{{p1, p2}},
   Alignment → {Top, Top},
   Frame → None,
   Spacings → 2,
   ItemSize → {Full, Full}
  ]
 ];

doChi[] := Module[{},
  If[gDebug, Print["Enter doChi[]"]];
  Manipulate[If[n == 0, n = N[10^-6]]; chi[n, maxX],
   {{n, 5, "number of degress of freedom (positive real number) n="}, 0, 100, .001,
    Appearance → "Labeled"},
   {{maxX, 10, "[display] control max x range="}, 1, 1000, 1, Appearance → "Labeled"},
   ContinuousAction → True
  ]
 ];

(****************************************)
(*           Uniform Continous          *)
(****************************************)
uniformCont[minX_, maxX_, scale_] := Module[{title, p1, p2, x, mean, var},
  If[gDebug, Print["Enter uniformCont[]"]];

  mean = N[Mean[UniformDistribution[{minX, maxX}]]];
  var = N[Variance[UniformDistribution[{minX, maxX}]]];

  title = Text["\t\tf(x)=1/(max - min)\nMean (max + min)/2=" <>
     ToString[mean] <> "    Var 1/12 (max-min)^2=" <> ToString[var]];

  p1 = Plot[PDF[UniformDistribution[{minX, maxX}], x], {x, minX - 0.1 minX, maxX + 0.1 maxX},
    Frame → True,
    FrameLabel → (Style[#, FontSize → 8] & /@ {"x", "f(x)", title}),
    ImageSize → {250},
    PlotRange → All
   ];

  p2 = Plot[CDF[UniformDistribution[{minX, maxX}], x], {x, minX, maxX},
    ImageSize → {250},
    Frame → True,
    FrameLabel → (Style[#, FontSize → 10] & /@ {"x", "F(x)", Text["\nCDF\n"]}),
    PlotRange → All];

  Grid[{{p1, p2}},
   Alignment → {Top, Top},
```

```
      Frame → None,
      Spacings → 2,
      ItemSize → {Full, Full}
     ]
    ];

  doUniformCont[] := Module[{},
    If[gDebug, Print["Enter doUniformCont[]"]];
    Manipulate[If[max > min, uniformCont[min, max, scale], Text["Min must be <= Max"]],
     {{min, 2, "min="}, -1000, 1000, 0.01, Appearance → "Labeled"},
     {{max, 10, "max="}, -1000 + 0.01, 1000 + 0.01, 0.01, Appearance → "Labeled"},
     {{scale, False, "Scale P(x) axis to be 1 always"}, {True, False}},
     ContinuousAction → True
    ]
   ];

 }
]
```

Out[1]=