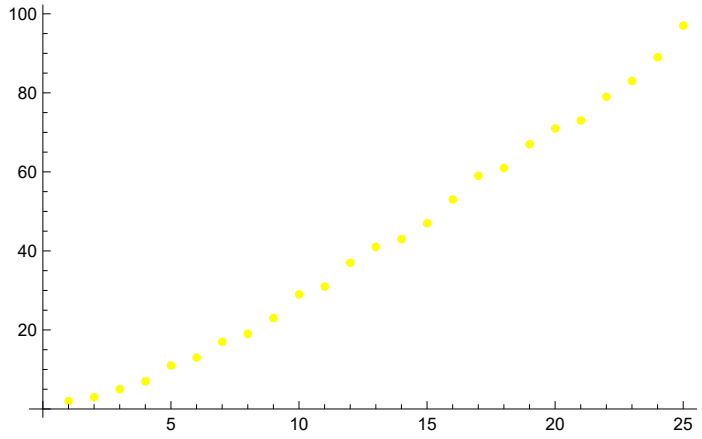
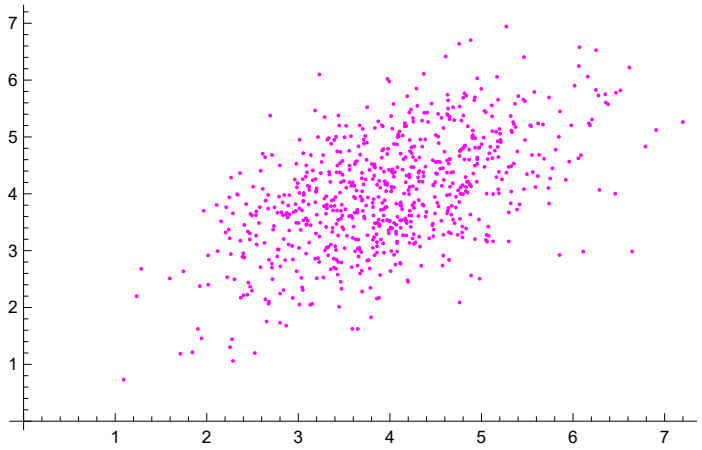
In[•]:=
```
ListPlot[Prime[Range[25]], PlotStyle → Yellow]
```

Out[•]=



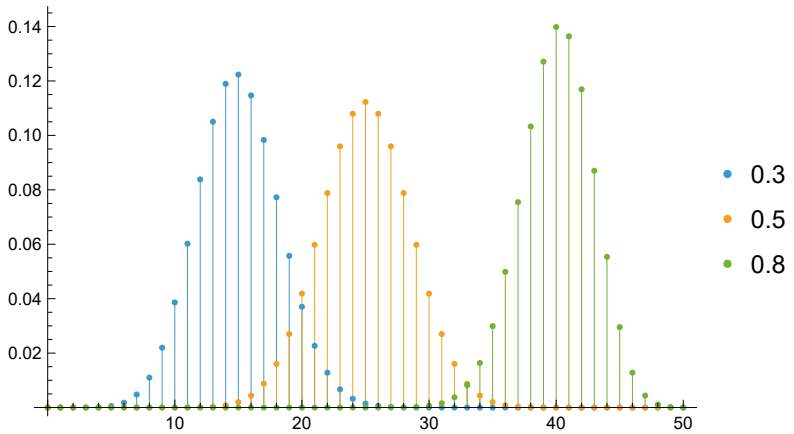
In[•]:=
```
ListPlot[RandomVariate[BinormalDistribution[{4, 4}, {1, 1}, 0.5], 750],
 PlotStyle → Magenta]
```

Out[•]=



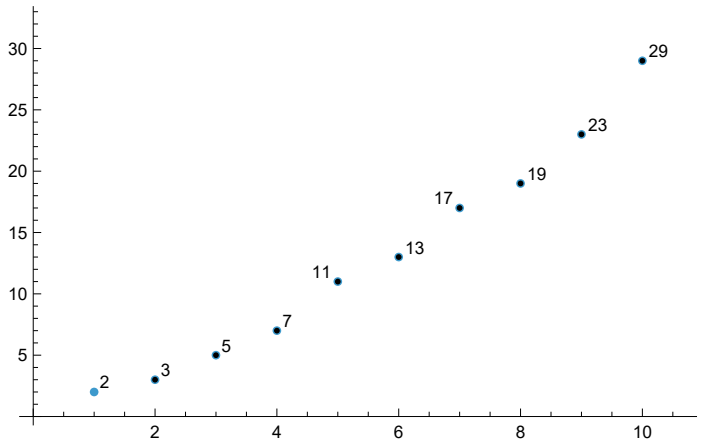
In[•]:=
```
ListPlot[Table[{k, PDF[BinomialDistribution[50, p], k]},
  {p, {0.3, 0.5, 0.8}}, {k, 0, 50}], Filling → Axis, PlotLegends → {0.3, 0.5, 0.8}]
```

Out[•]=



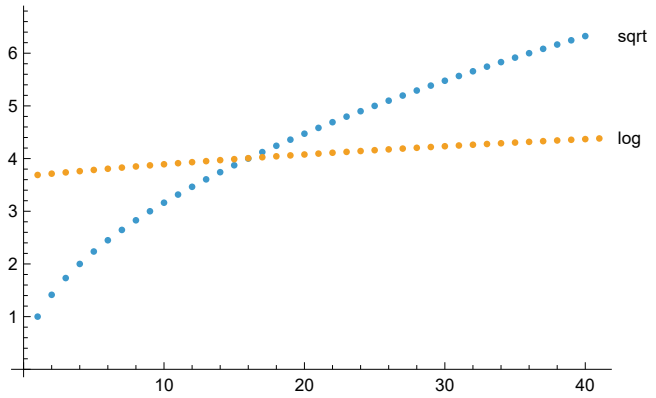
In[•]:=
```
ListPlot[Labeled[#, #] & /@ Table[Prime[n], {n, 10}], PlotStyle → PointSize[Medium]]
```

Out[•]=



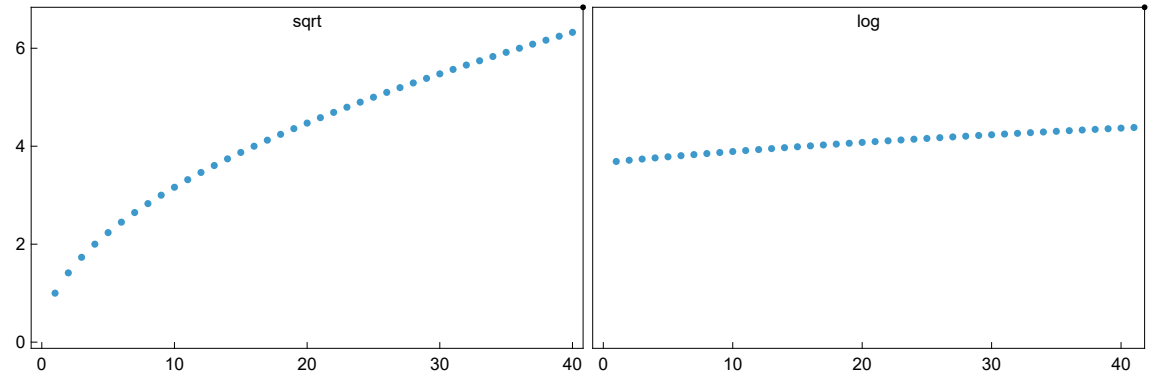
In[•]:=
```
ListPlot[{Labeled[Sqrt[Range[40]], "sqrt"], Labeled[Log[Range[40, 80]], "log"]}]
```

Out[•]=



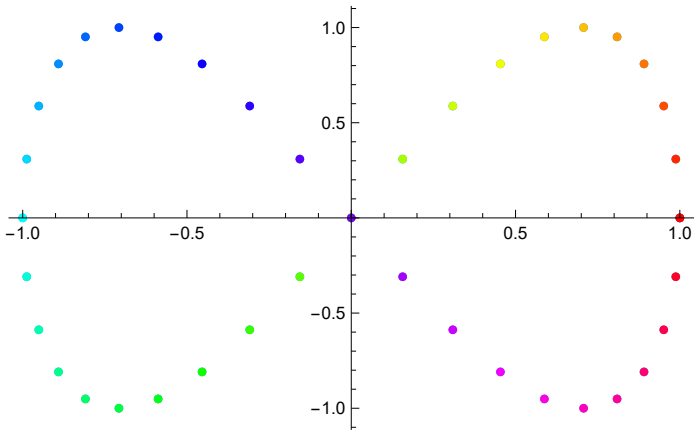
In[•]:=
```
ListPlot[{Labeled[Sqrt[Range[40]], "sqrt"],
  Labeled[Log[Range[40, 80]], "log"]}, PlotLayout → "Row"]
```

Out[•]=



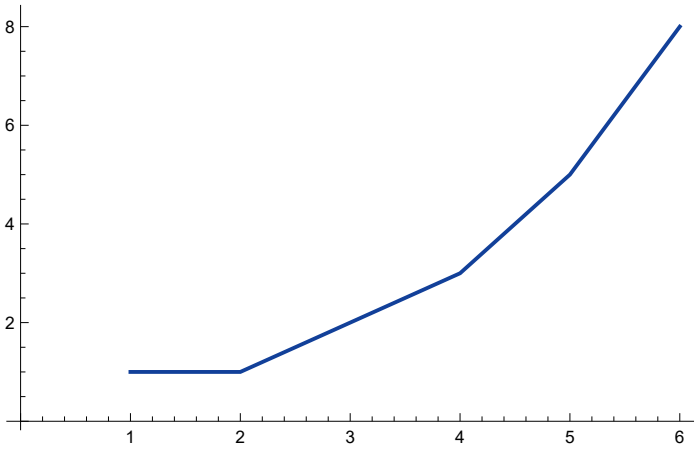
In[•]:=
```
ListPlot[Table[Style[{Cos[t], Sin[2 t]}, Hue[t / (2 Pi)]], {t, 0, 2 Pi, Pi / 20}],
 PlotStyle → PointSize[Medium]]
```

Out[•]=



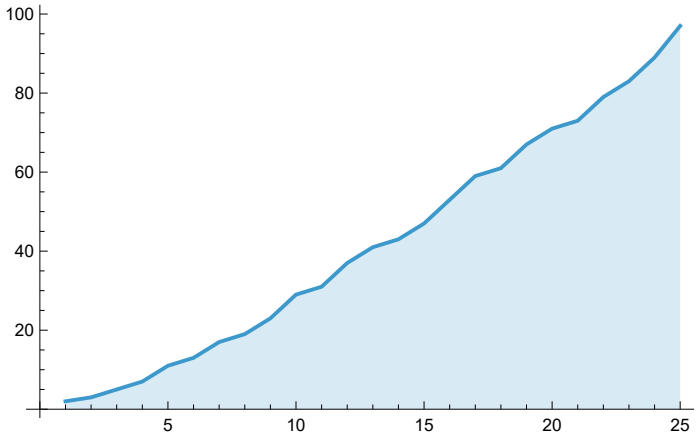
In[•]:=
```
ListLinePlot[{1, 1, 2, 3, 5, 8}, PlotStyle → DarkBlue]
```

Out[•]=



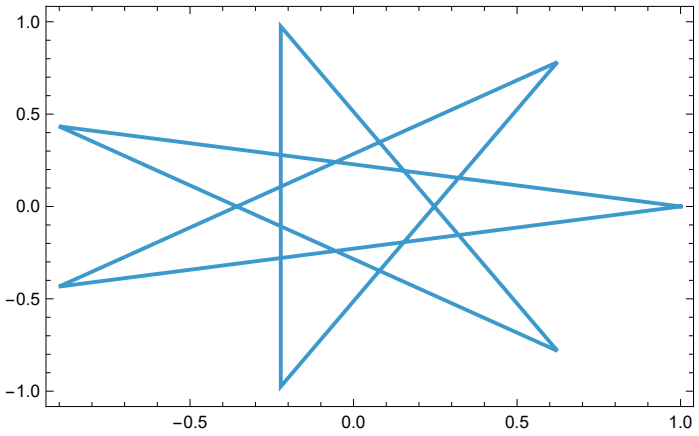
In[•]:=
```
ListLinePlot[Prime[Range[25]], Filling → Axis]
```

Out[•]=



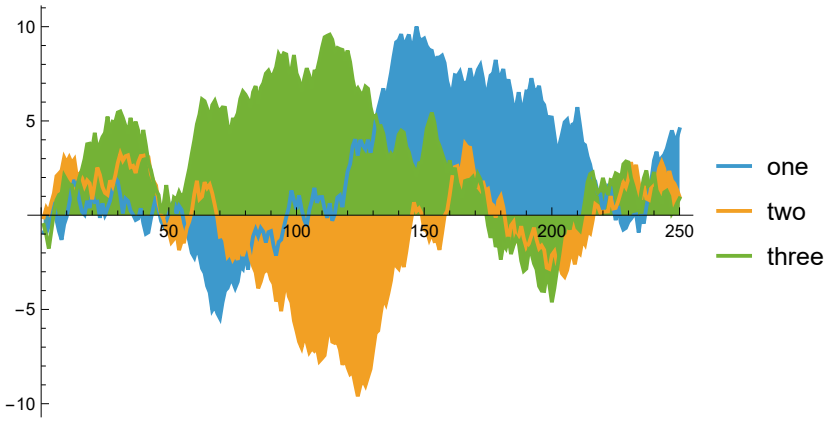
In[•]:=
```
ListLinePlot[Table[{Cos[k 2 Pi / 7], Sin[k 2 Pi / 7]}, {k, 0, 21, 3}],
 Frame → True, Axes → False]
```

Out[•]=



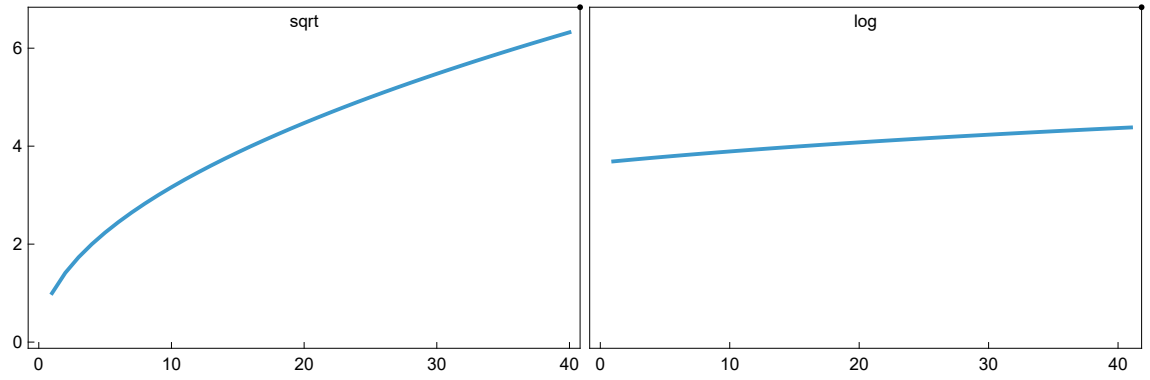
In[•]:=
```
ListLinePlot[Table[Accumulate[RandomReal[{-1, 1}, 250]], {3}],
 Filling → Axis, PlotLegends → {"one", "two", "three"}]
```

Out[•]=



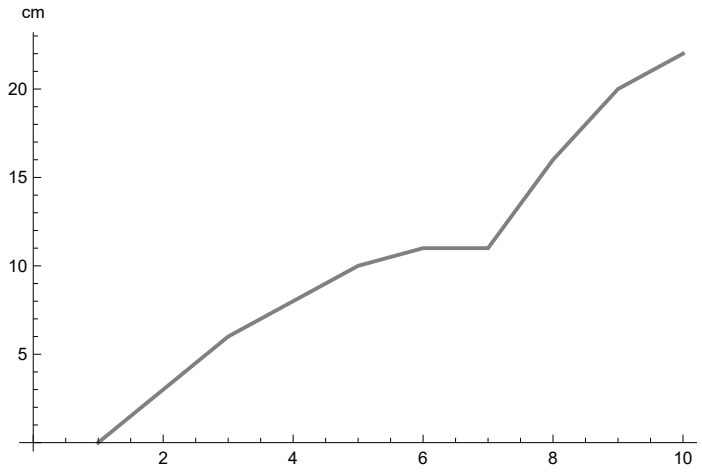
In[•]:=
```
ListLinePlot[{Labeled[Sqrt[Range[40]], "sqrt"],
  Labeled[Log[Range[40, 80]], "log"]}, PlotLayout → "Row"]
```

Out[•]=



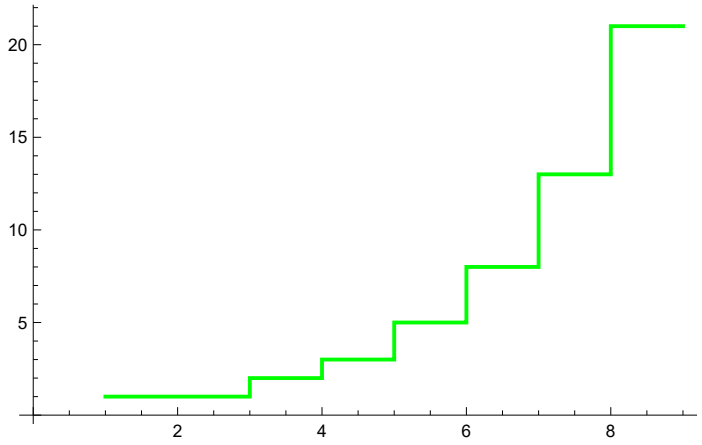
```
In[•]:= ListLinePlot[Quantity[{0, 3, 6, 8, 10, 11, 11, 16, 20, 22}, "Centimeters"],
         PlotStyle → Gray, AxesLabel → Automatic]
```

Out[•]=



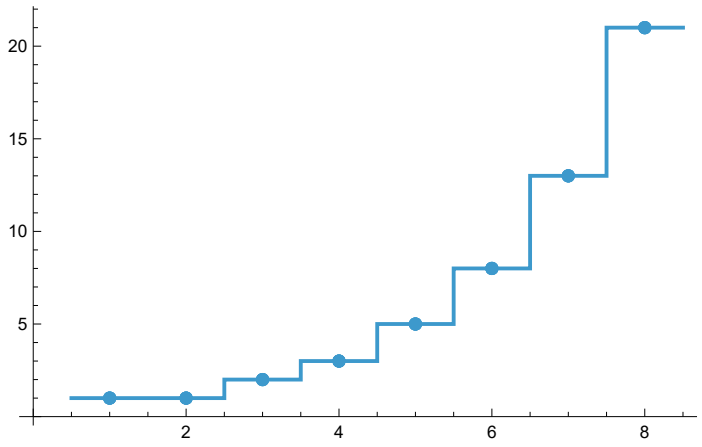

```
In[•]:= ListStepPlot[{1, 1, 2, 3, 5, 8, 13, 21}, PlotStyle → Green]
```

Out[•]=



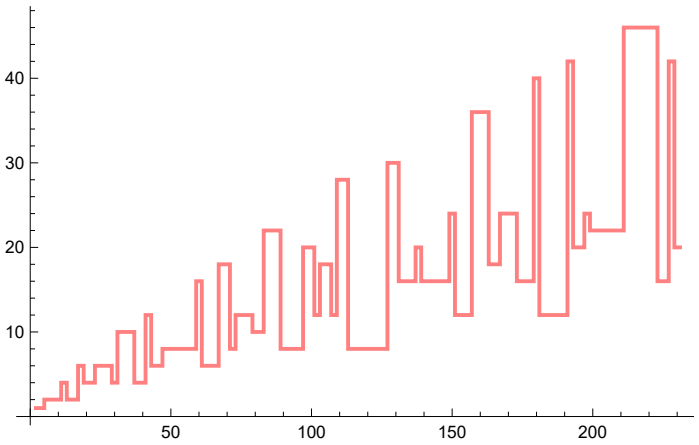

```
In[•]:= ListStepPlot[{1, 1, 2, 3, 5, 8, 13, 21}, Center, Mesh → Full]
```

Out[•]=



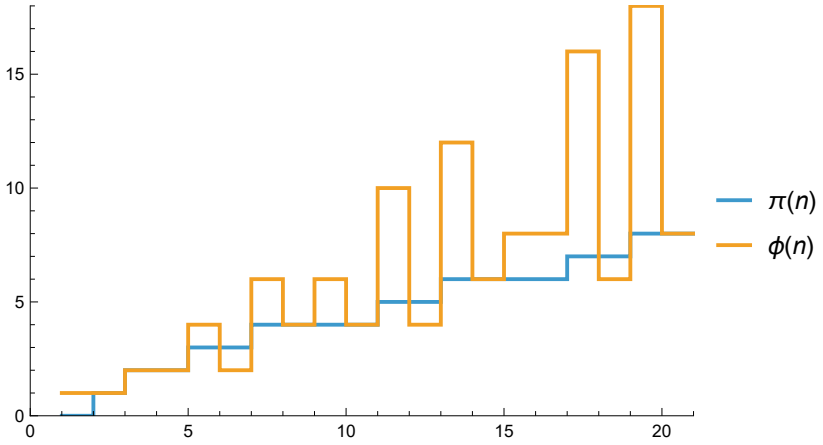
In[•]:=
```
ListStepPlot[Table[{Prime[n], EulerPhi[n]}, {n, 50}], PlotStyle → Pink]
```

Out[•]=



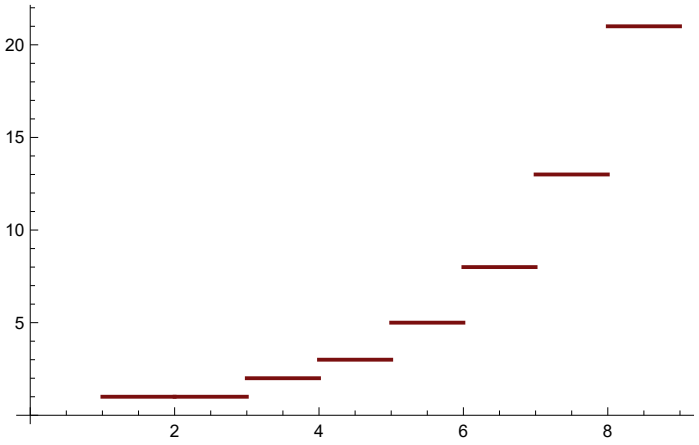
In[•]:=
```
ListStepPlot[{Legended[PrimePi[Range[20]], PrimePi[n]],
  Legended[EulerPhi[Range[20]], EulerPhi[n]]}]
```

Out[•]=



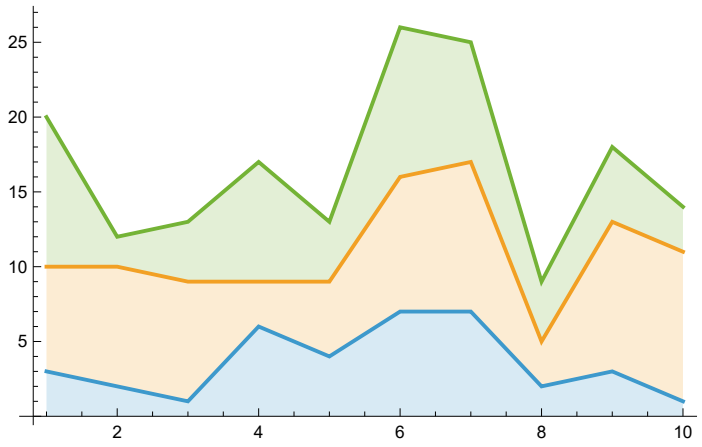
In[•]:=
```
ListStepPlot[{1, 1, 2, 3, 5, 8, 13, 21}, PlotStyle → DarkRed, Joined → False]
```

Out[•]=



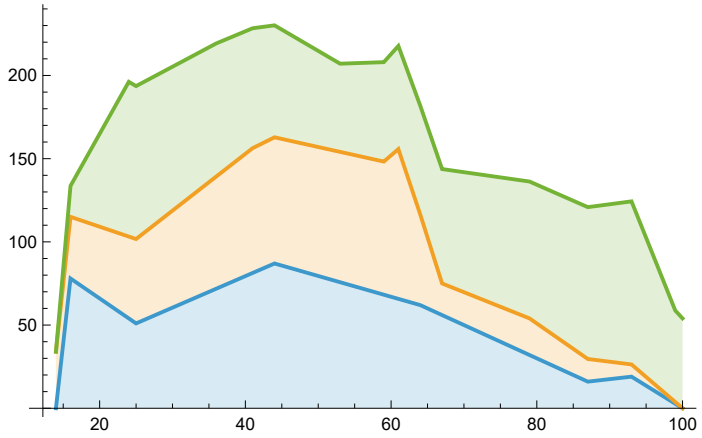
In[•]:=
```
StackedListPlot[{{3, 2, 1, 6, 4, 7, 7, 2, 3, 1},
  {7, 8, 8, 3, 5, 9, 10, 3, 10, 10}, {10, 2, 4, 8, 4, 10, 8, 4, 5, 3}}]
```

Out[•]=



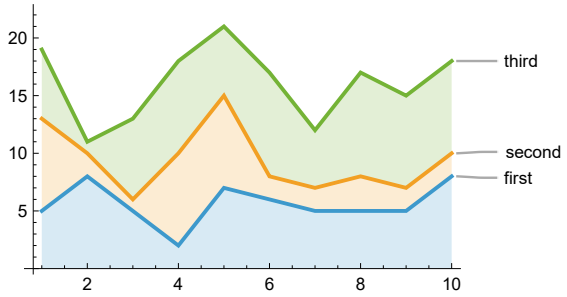
In[•]:=
```
d = {{{16, 78}, {25, 51}, {44, 87}, {64, 62}, {87, 16}, {93, 19}},
   {{14, 34}, {41, 75}, {59, 80}, {61, 90}, {67, 19}, {79, 22}},
   {{24, 93}, {36, 80}, {53, 53}, {93, 98}, {99, 55}, {100, 54}}};
StackedListPlot[d]
```

Out[•]=



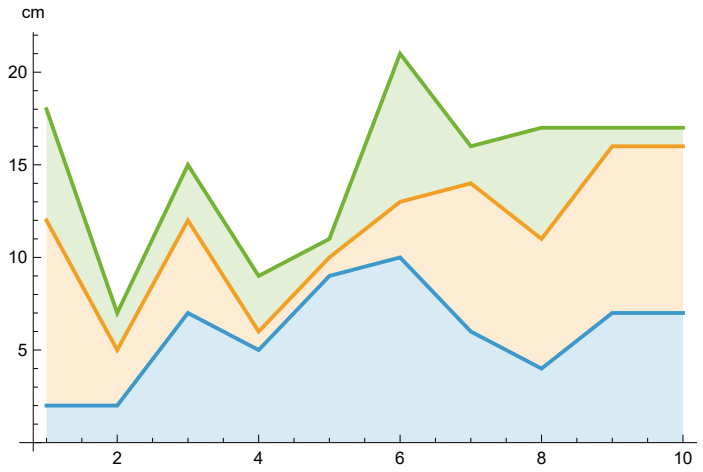
In[•]:=
```
StackedListPlot[MapIndexed[Callout[#, IntegerName[First[#2], "Ordinal"]] &,
  RandomInteger[{1, 10}, {3, 10}]], ImageSize → 300]
```

Out[•]=



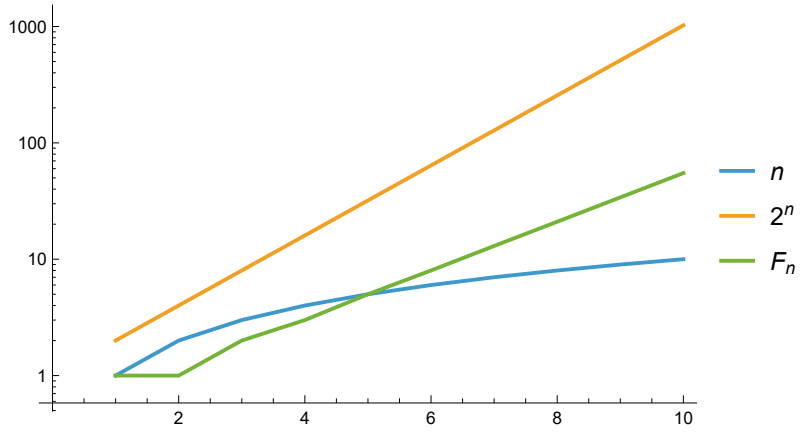
In[•]:=
```
StackedListPlot[
 Quantity[RandomInteger[{1, 10}, {3, 10}], "Centimeters"], AxesLabel → Automatic]
```

Out[•]=

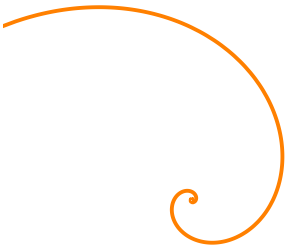


In[•]:=
```
{data1, data2, data3} = Transpose[Table[{n, 2^n, Fibonacci[n]}, {n, 10}]];
ListLogPlot[{data1, data2, data3}, Joined → True, PlotLegends → {n, 2^n, Fibonacci[n]}]
```

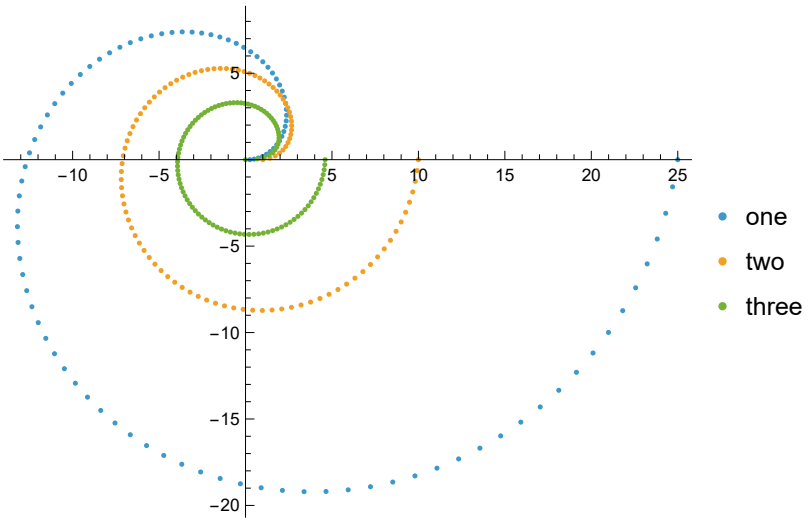
Out[•]=



```
In[•]:= PolarPlot[(GoldenRatio^n) / Sqrt[5], {n, 0, 40}, PlotStyle → Orange, Axes → False]
Out[•]=
```

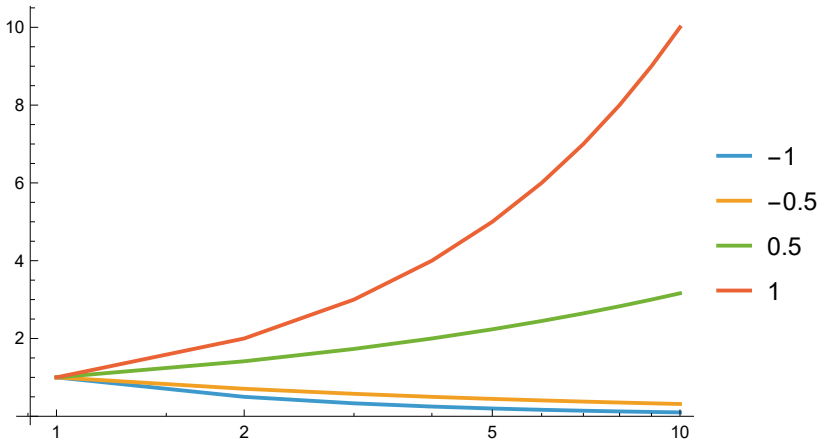


```
In[•]:= ListPolarPlot[{Range[100] / 4, Sqrt[Range[100]], Log[Range[100]]},
         DataRange → {0, 2 Pi}, PlotLegends → {"one", "two", "three"}]
Out[•]=
```



In[•]:=
```
ListLogLinearPlot[Table[{n, n^k}, {k, {-1, -0.5, 0.5, 1}}, {n, 1, 10}],
 Joined → True, PlotLegends → {-1, -0.5, 0.5, 1}]
```

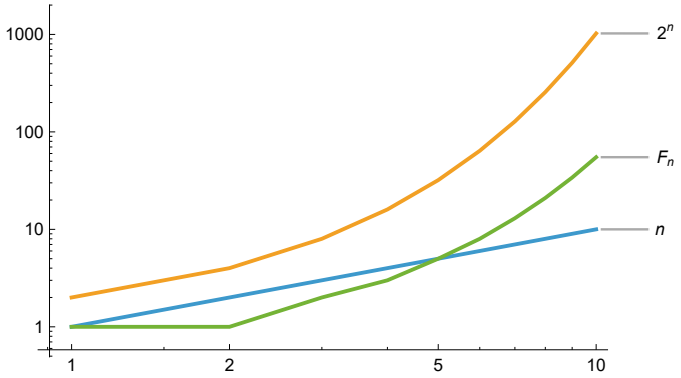
Out[•]=




```
{data1, data2, data3} = Transpose[Table[{n, 2^n, Fibonacci[n]}, {n, 10}]];
```

In[•]:=
```
ListLogLogPlot[{data1, data2, data3}, Joined → True, PlotLabels → {n, 2^n, Fibonacci[n]}]
```

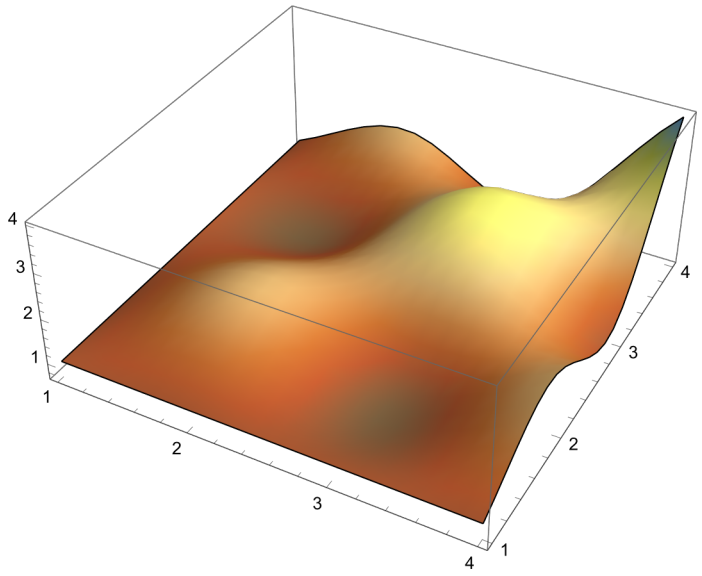
Out[•]=



In[•]:=
```
data = {{1, 1, 1, 1}, {1, 2, 1, 2}, {1, 1, 3, 1}, {1, 2, 1, 4}}
ListPlot3D[data, Mesh → None, InterpolationOrder → 3, ColorFunction → "SouthwestColors"]
```

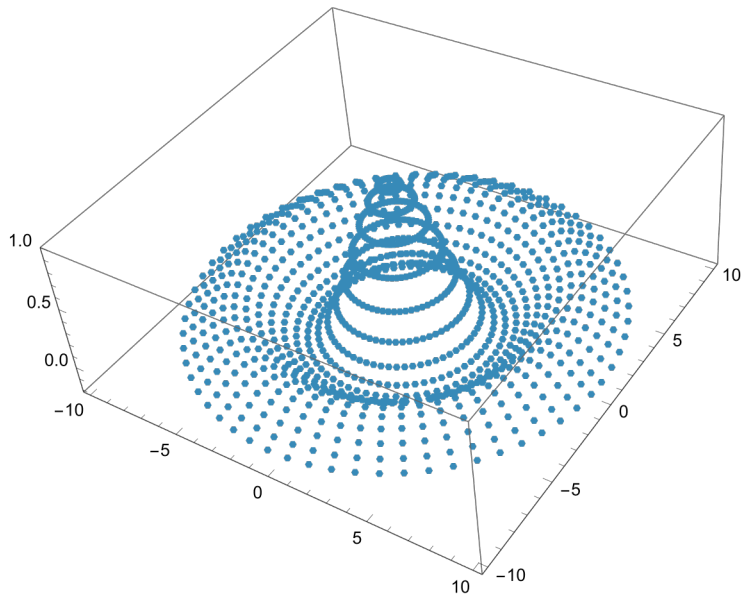
Out[•]=
```
{{1, 1, 1, 1}, {1, 2, 1, 2}, {1, 1, 3, 1}, {1, 2, 1, 4}}
```

Out[•]=



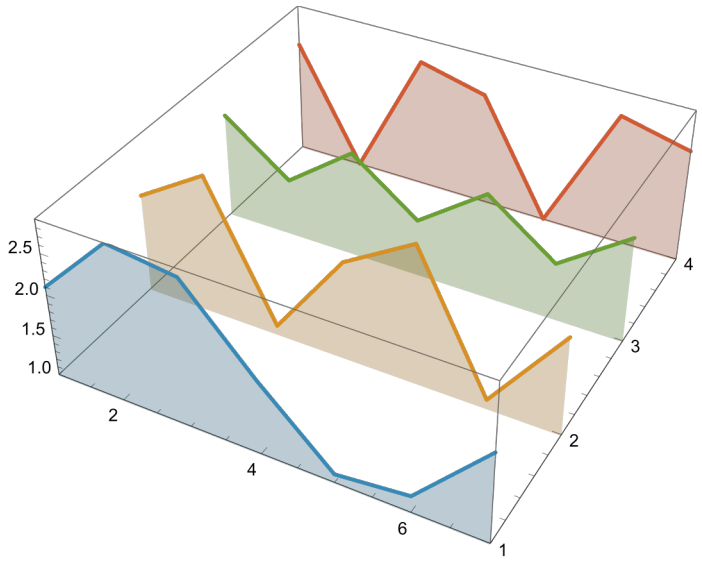
In[•]:=
```
data =
  Flatten[Table[{r Cos[t], r Sin[t], Sinc[r]}, {r, 0, 10, 0.5}, {t, 0, 2 Pi, 0.1}], 1];
ListPointPlot3D[data]
```

Out[•]=



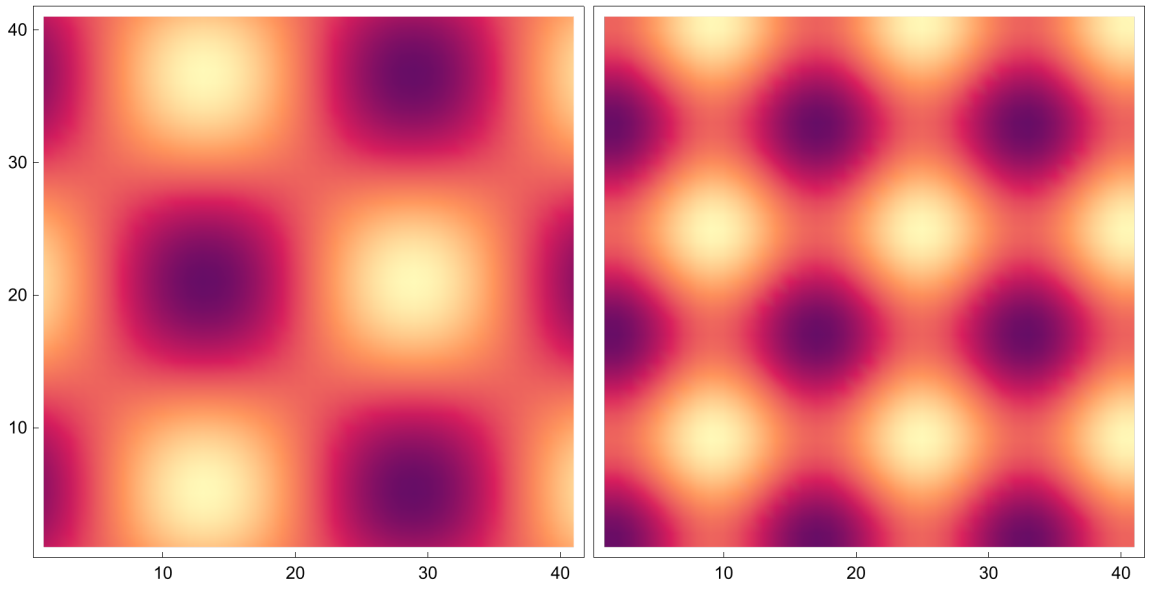
In[•]:=
```
ListLinePlot3D[{{2.1, 2.9, 2.8, 1.9, 1.1, 1.2, 2.1}, {2.2, 2.7, 1.1, 2.2, 2.7, 1.1, 2.2},
  {2.3, 1.7, 2.3, 1.7, 2.3, 1.7, 2.3}, {2.4, 1., 2.6, 2.4, 1., 2.6, 2.4}}, Filling → Axis]
```

Out[•]=



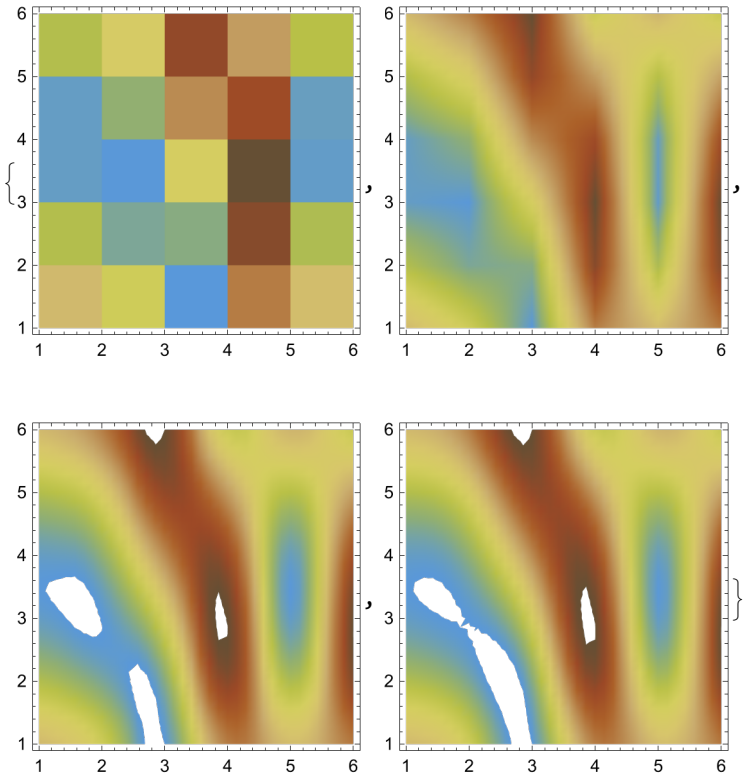
In[•]:=
```
ListDensityPlot[{Table[Sin[x] Cos[y], {y, -4, 4, 0.2}, {x, -4, 4, 0.2}],
  Table[Sin[x + y] Cos[x - y], {y, -4, 4, 0.2}, {x, -4, 4, 0.2}]}, PlotLayout → "Row"]
```

Out[•]=



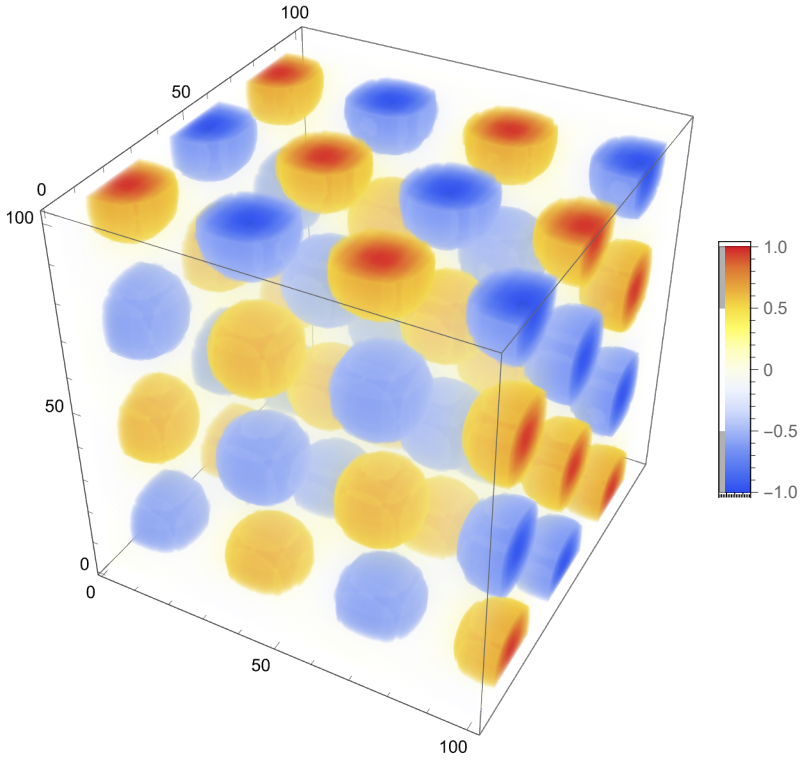
```
In[•]:= data = Table[Sin[j^2 + i], {i, 0, Pi, Pi / 5}, {j, 0, Pi, Pi / 5}];
        Table[ListDensityPlot[data, Mesh → None, InterpolationOrder → o,
          ColorFunction → "SouthwestColors"], {o, {0, 1, 2, 3}}]
```

Out[•]=



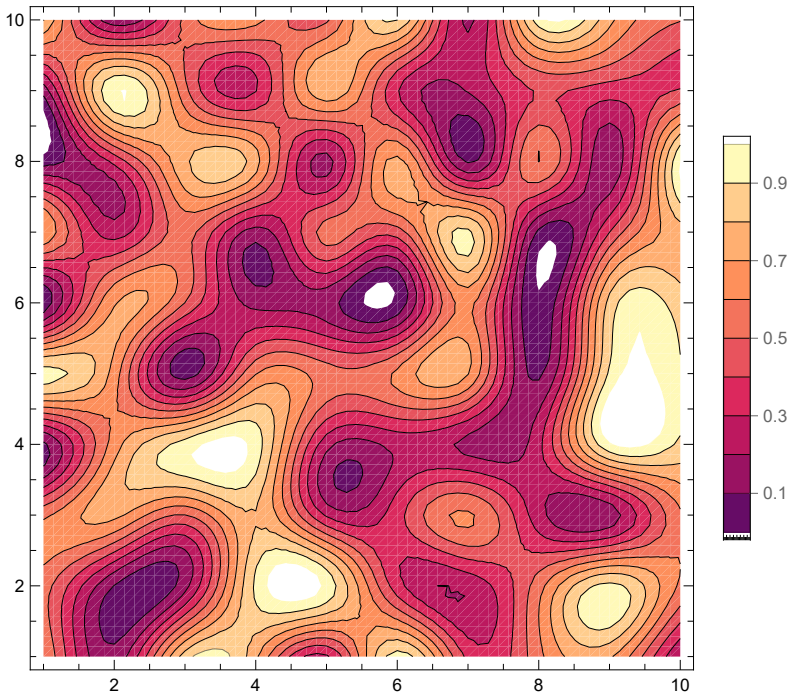
```
In[•]:= data = Table[Sin[x] Cos[y] Sin[z], {z, -5, 5, 0.1}, {y, -5, 5, 0.1}, {x, -5, 5, 0.1}];
        ListDensityPlot3D[data, ColorFunction → "TemperatureMap", PlotLegends → Automatic]
```

Out[•]=



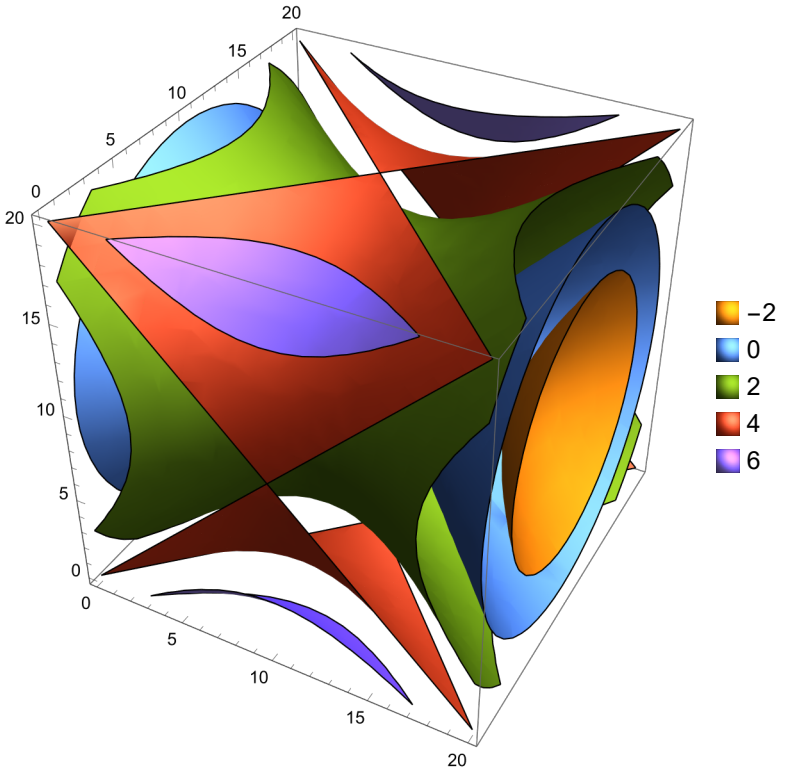
In[•]:=
```
ListContourPlot[RandomReal[1, {10, 10}], InterpolationOrder → 3, PlotLegends → Automatic]
```

Out[•]=



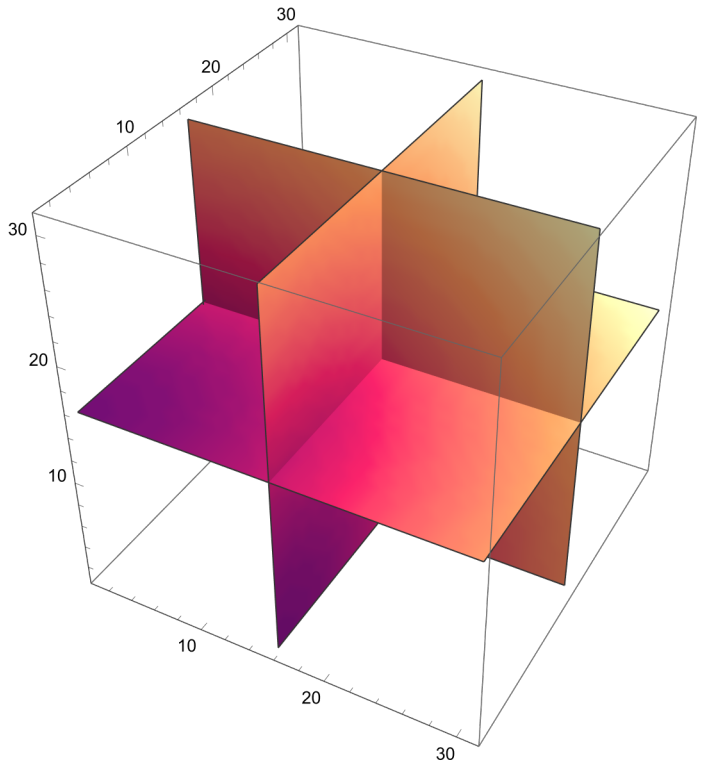
In[•]:=
```
ListContourPlot3D[
 Table[x^2 + y^2 - z^2, {x, -2, 2, 0.2}, {y, -2, 2, 0.2}, {z, -2, 2, 0.2}],
 Contours → 5, Mesh → None, PlotLegends → "Expressions"]
```

Out[•]=



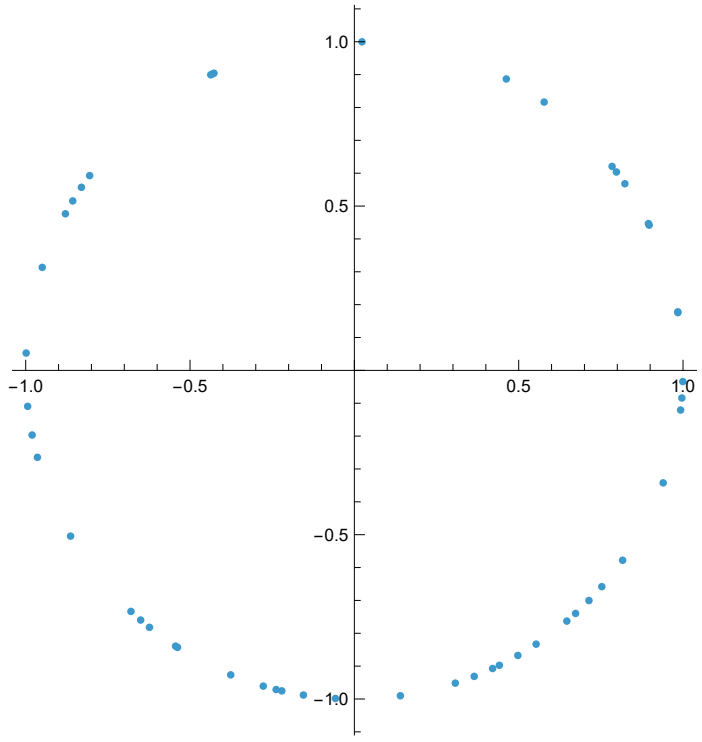
In[•]:=
```
data = Table[Sqrt[x^2 + y^2 + z^2], {z, 0, 3, 0.1}, {y, 0, 3, 0.1}, {x, 0, 3, 0.1}];
ListSliceDensityPlot3D[data, "CenterPlanes"]
```

Out[•]=



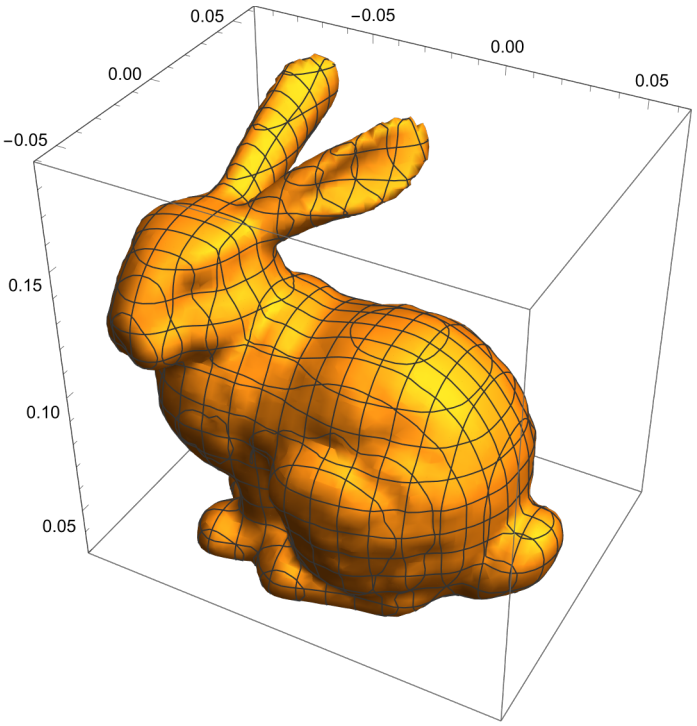
In[•]:=
```
data = Table[{Cos[t], Sin[t]}, {t, RandomReal[{0, 2 Pi}, 50]}];
ListPlot[data, AspectRatio → Automatic]
```

Out[•]=



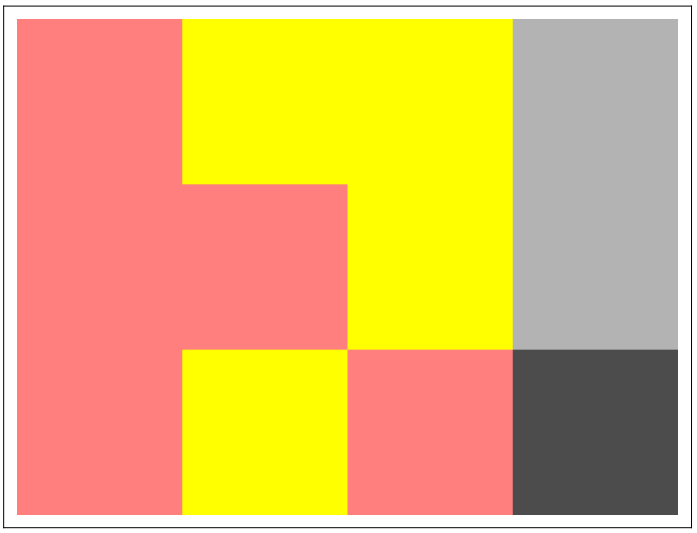
In[•]:=
```
data = ExampleData[{"Geometry3D", "StanfordBunny"}, "VertexData"];
ListSurfacePlot3D[data, MaxPlotPoints → 50]
```

Out[•]=



In[•]:=
```
ArrayPlot[{{1, 0, 0, 0.3}, {1, 1, 0, 0.3}, {1, 0, 1, 0.7}},
 ColorRules → {1 → Pink, 0 → Yellow}]
```

Out[•]=

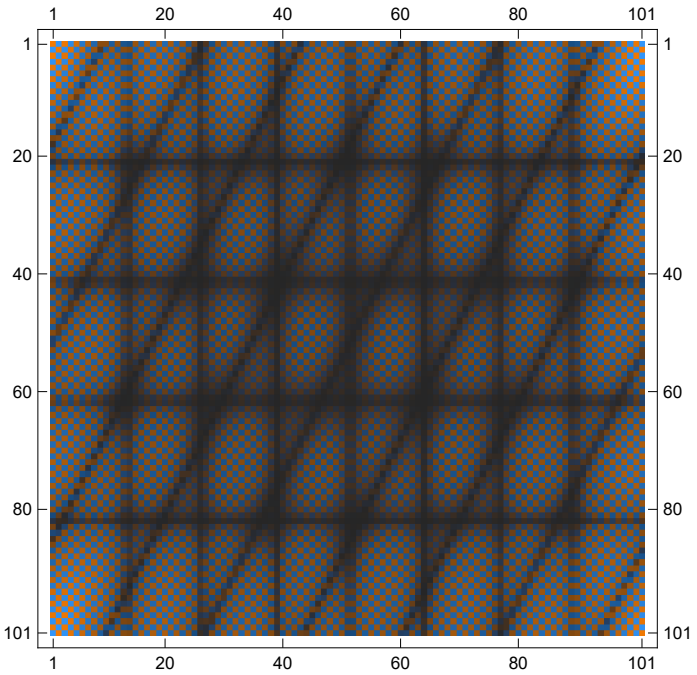


In[•]:=
```
ReliefPlot[Import["http://exampledata.wolfram.com/hailey.dem.gz", "Data"],
 ColorFunction → "GreenBrownTerrain"]
```

Out[•]=

In[•]:=
```
MatrixPlot[
 Fourier[Table[UnitStep[i, 4 - i] UnitStep[j, 7 - j], {i, -50, 50}, {j, -50, 50}]]]
```

Out[•]=

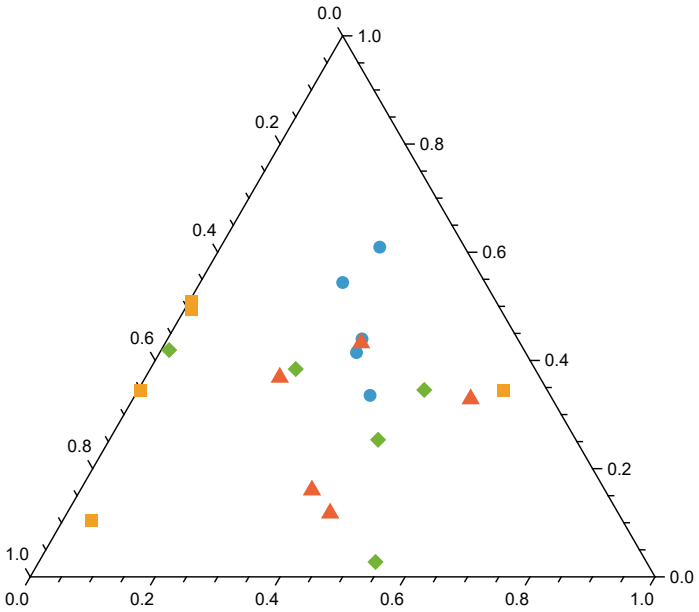


In[•]:=
```
d = RandomReal[5, {5, 5, 5}];
ArrayPlot3D[d, ColorFunction → "Rainbow"]
```

Out[•]=

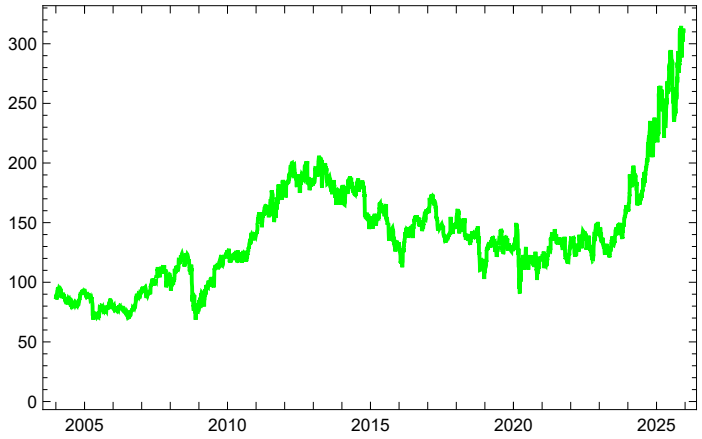
In[•]:=
```
TernaryListPlot[RandomReal[1, {4, 5, 3}], PlotMarkers → Automatic]
```

Out[•]=



In[•]:=
```
DateListPlot[FinancialData["IBM", "Jan. 1, 2004"], PlotStyle → Green]
```

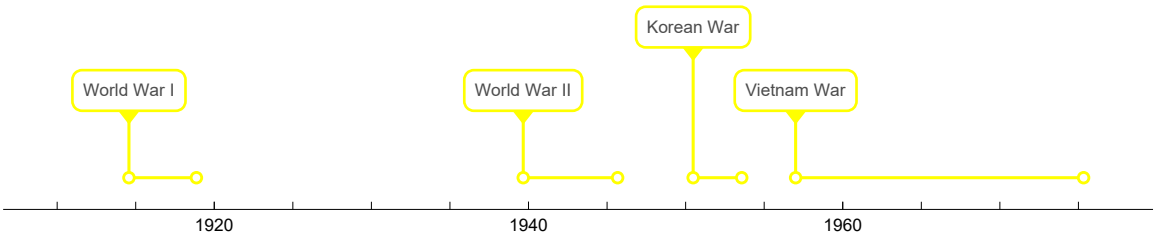
Out[•]=



Out[•]=

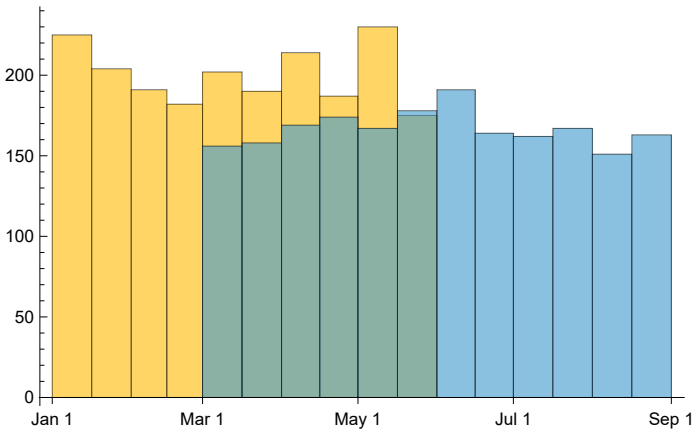
In[•]:=
```
TimelinePlot[{Entity["HistoricalEvent", "WorldWar1"],
  Entity["HistoricalEvent", "WorldWar2"], Entity["HistoricalEvent", "VietnamWar"],
  Entity["HistoricalEvent", "KoreanWarBegins"]}, PlotStyle → Yellow]
```

Out[•]=



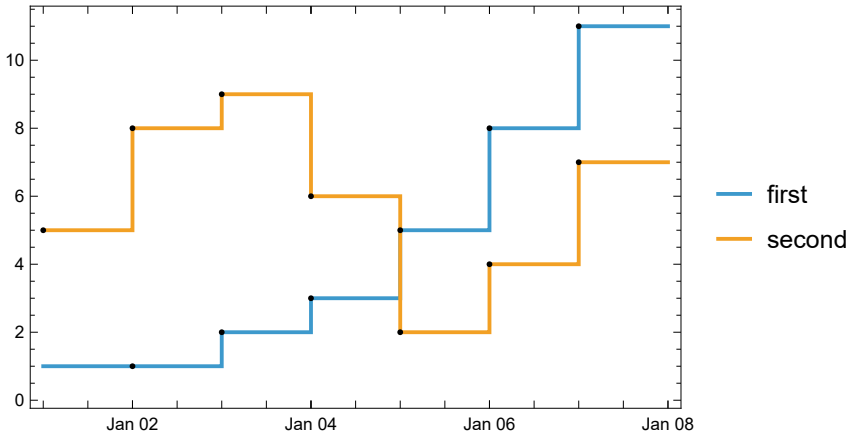
In[•]:=
```
data1 = RandomChoice[DateRange[{2010, 1, 1}, {2010, 5, 31}, "Day"], 2000];
data2 = RandomChoice[DateRange[{2010, 3, 1}, {2010, 8, 31}, "Day"], 2000];
DateHistogram[{data1, data2}]
```

Out[•]=



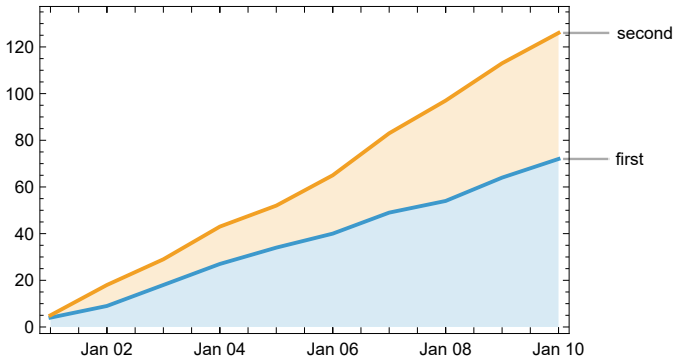
In[•]:=
```
data1 = TimeSeries[{1, 1, 2, 3, 5, 8, 11}, {"Jan 1, 2015"}];
data2 = TimeSeries[{5, 8, 9, 6, 2, 4, 7}, {"Jan 1, 2015"}];
DateListStepPlot[{data1, data2}, PlotLegends → {"first", "second"}]
```

Out[•]=



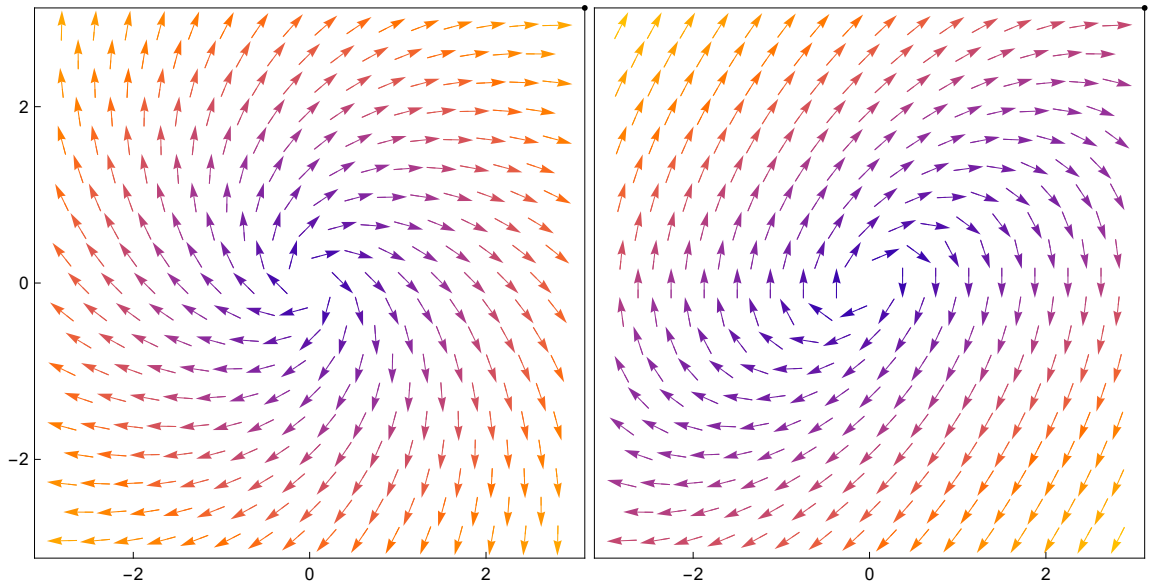
In[•]:=
```
StackedDateListPlot[
 {{4, 9, 18, 27, 34, 40, 49, 54, 64, 72}, {1, 9, 11, 16, 18, 25, 34, 43, 49, 54}},
 {2013, 1, 1}, Joined → True, PlotLabels → {"first", "second"}]
```

Out[•]=



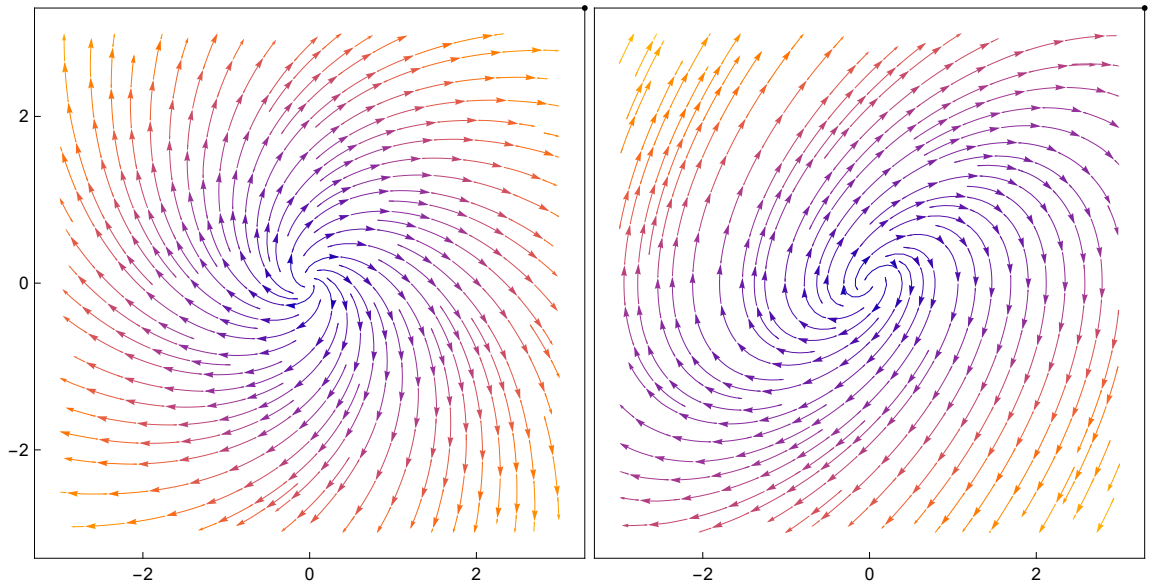
In[•]:=
```
ListVectorPlot[{Table[{{x, y}, {x + y, y - x}}, {x, -3, 3, 0.2}, {y, -3, 3, 0.2}],
  Table[{{x, y}, {y, y - x}}, {x, -3, 3, 0.2}, {y, -3, 3, 0.2}]}, PlotLayout → "Row"]
```

Out[•]=



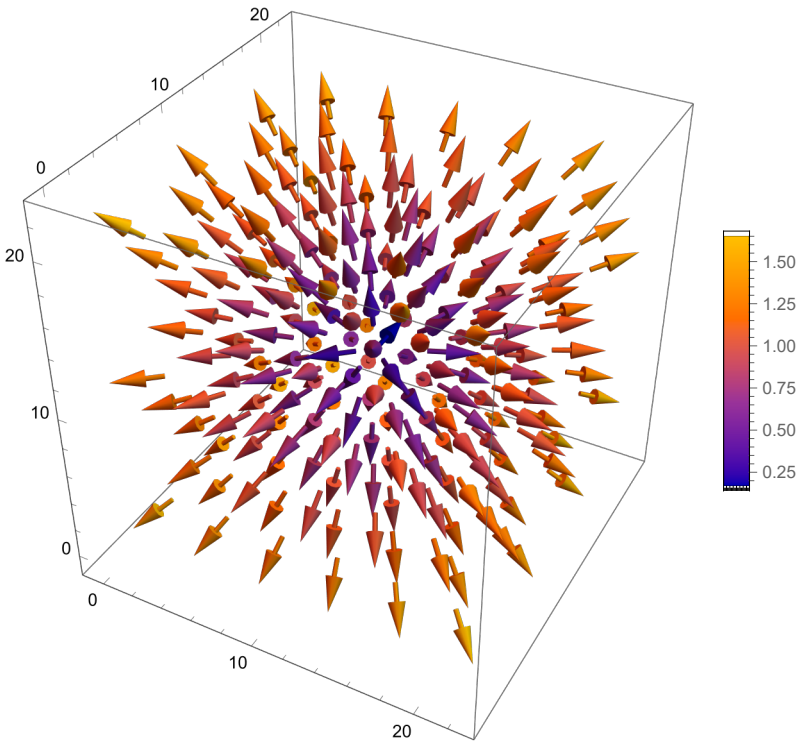
In[•]:=
```
ListStreamPlot[{Table[{{x, y}, {x + y, y - x}}, {x, -3, 3, 0.2}, {y, -3, 3, 0.2}],
  Table[{{x, y}, {y, y - x}}, {x, -3, 3, 0.2}, {y, -3, 3, 0.2}]}, PlotLayout → "Row"]
```

Out[•]=



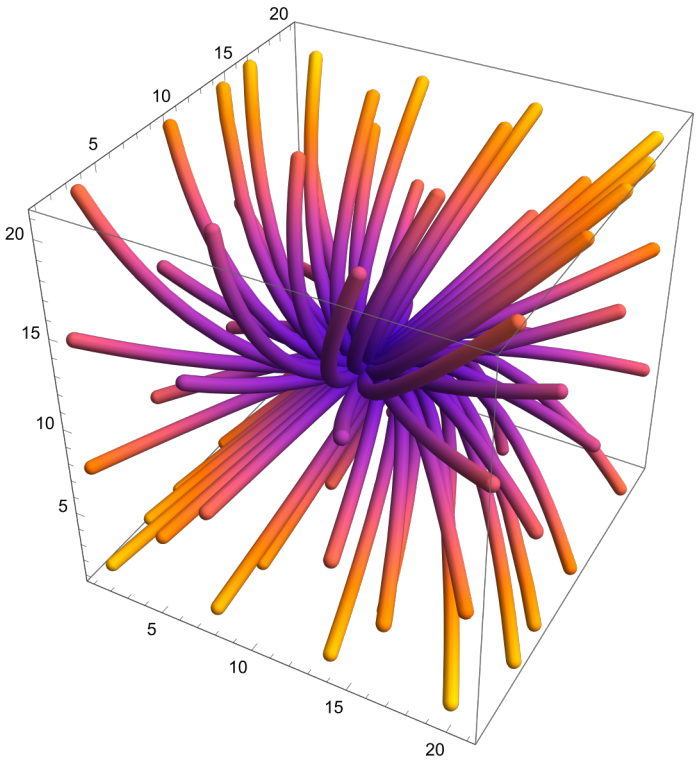
In[•]:=
```
ListVectorPlot3D[
 Table[{x, y, z}, {x, -1, 1, .1}, {y, -1, 1, .1}, {z, -1, 1, .1}], PlotLegends → Automatic]
```

Out[•]=



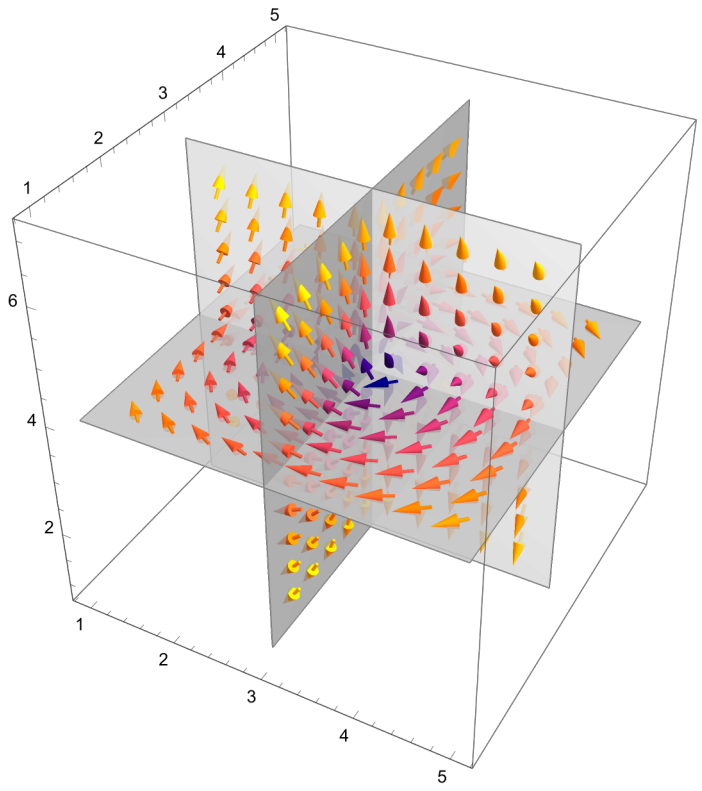
In[•]:=
```
ListStreamPlot3D[Table[{x, y + z, z}, {x, -1, 1, .1}, {y, -1, 1, .1}, {z, -1, 1, .1}],
 StreamMarkers → "Tube"]
```

Out[•]=



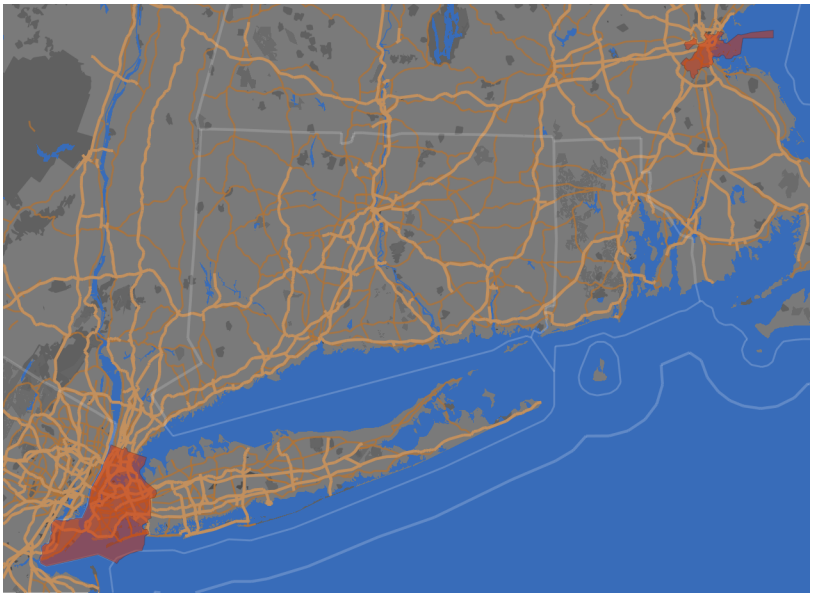
In[•]:=
```
data = Table[{y, -x, z}, {z, -3, 3}, {y, -2, 2}, {x, -2, 2}];
ListSliceVectorPlot3D[data, "CenterPlanes"]
```

Out[•]=



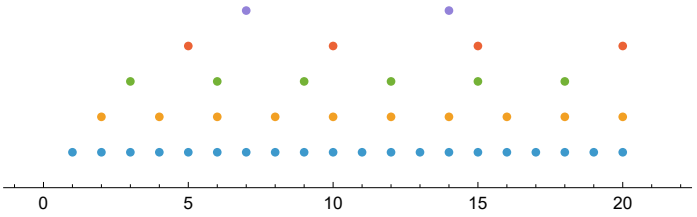
In[•]:=
```
GeoListPlot[{Entity["City", {"NewYork", "NewYork", "UnitedStates"}],
  Entity["City", {"Boston", "Massachusetts", "UnitedStates"}]}]
```

Out[•]=



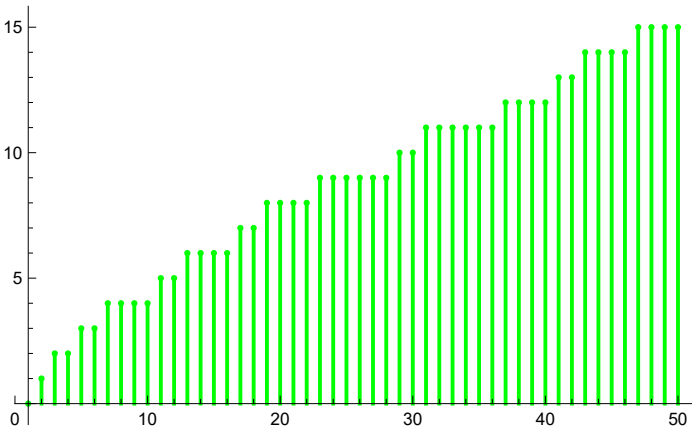
In[•]:=
```
NumberLinePlot[{Range[20], 2 Range[10], 3 Range[6], 5 Range[4], 7 Range[2]}]
```

Out[•]=

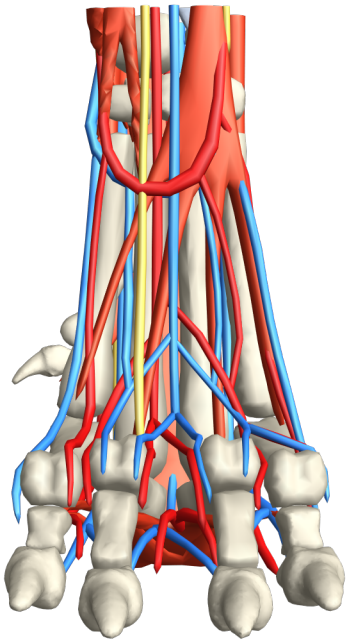


In[•]:=
```
DiscretePlot[PrimePi[k], {k, 1, 50}, PlotStyle → Green]
```

Out[•]=

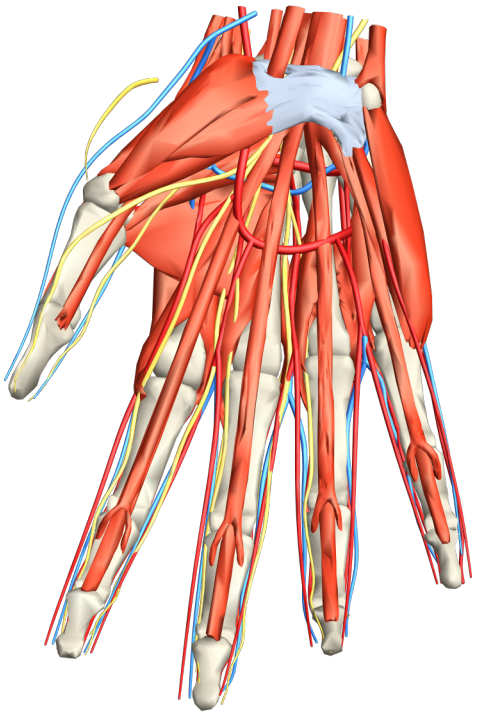


Out[•]=



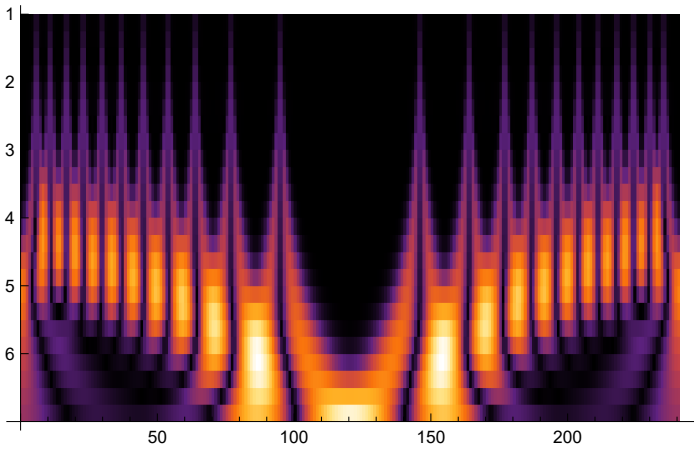
In[•]:=
```
AnatomyPlot3D[{Entity["AnatomicalStructure", "RightHand"]}]
```

Out[•]=



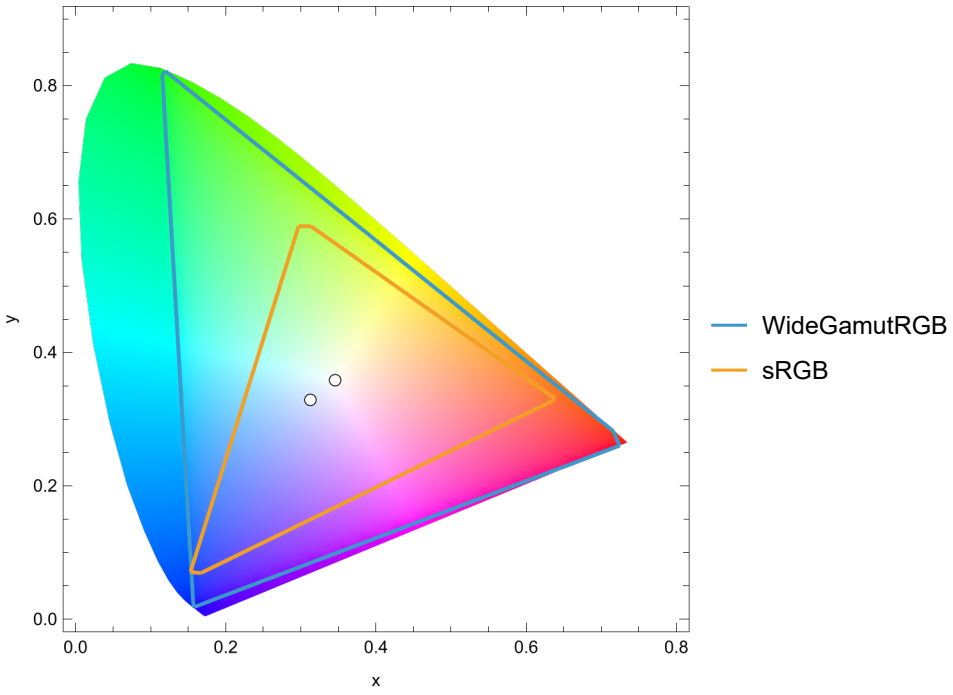
In[•]:=
```
ContinuousWaveletTransform[Table[Sign[Cos[x^2]], {x, -6, 6, 0.05}]];
ContinuousWaveletTransform[Table[Sign[Cos[x^2]], {x, -6, 6, 0.05}]];
WaveletScalogram[%]
```

Out[•]=



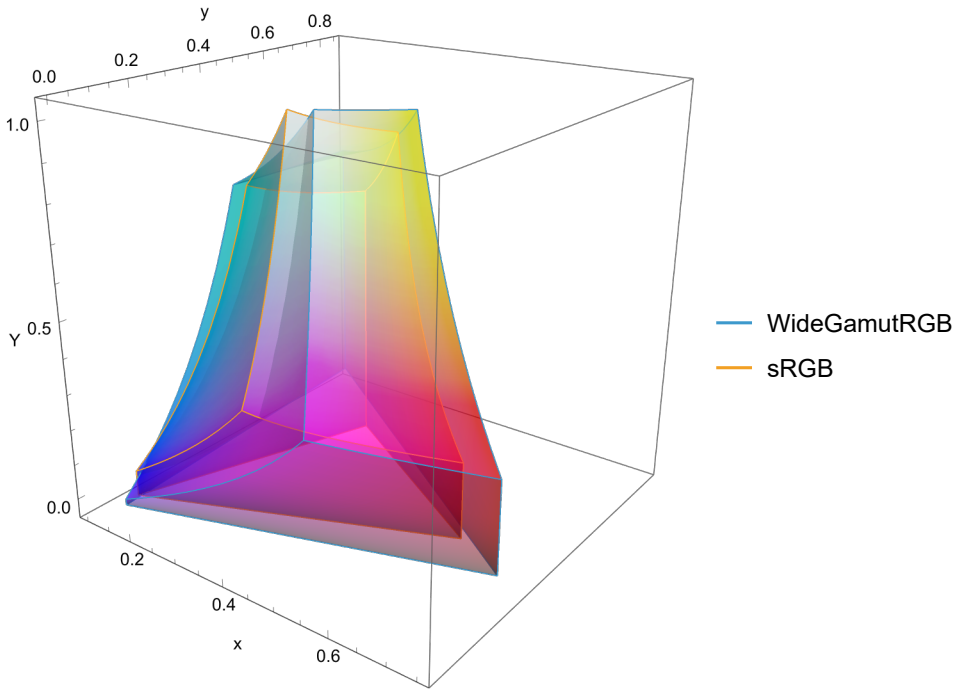
In[•]:=
```
ChromaticityPlot[{"WideGamutRGB", "sRGB"}]
```

Out[•]=



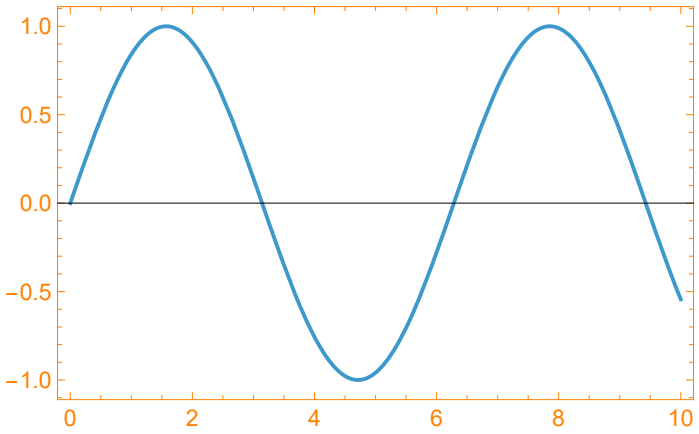
In[•]:=
```
ChromaticityPlot3D[{"WideGamutRGB", "sRGB"}]
```

Out[•]=



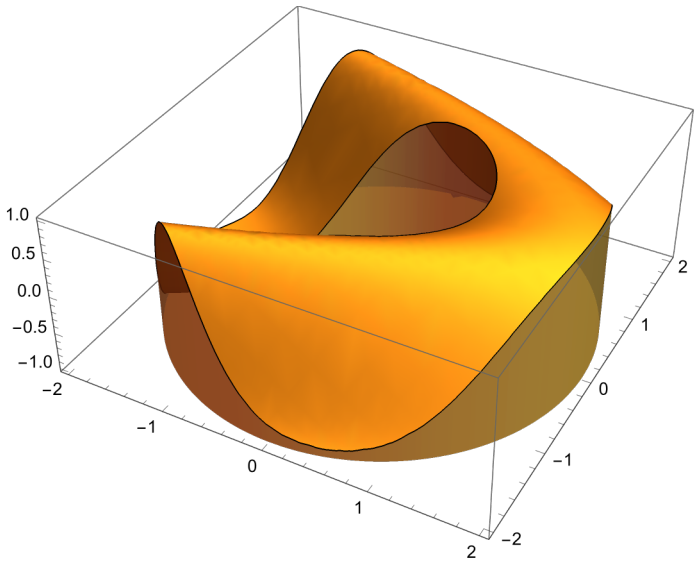
In[•]:=
```
Plot[Sin[x], {x, 0, 10}, Frame → True, FrameStyle → Directive[Orange, 12]]
```

Out[•]=



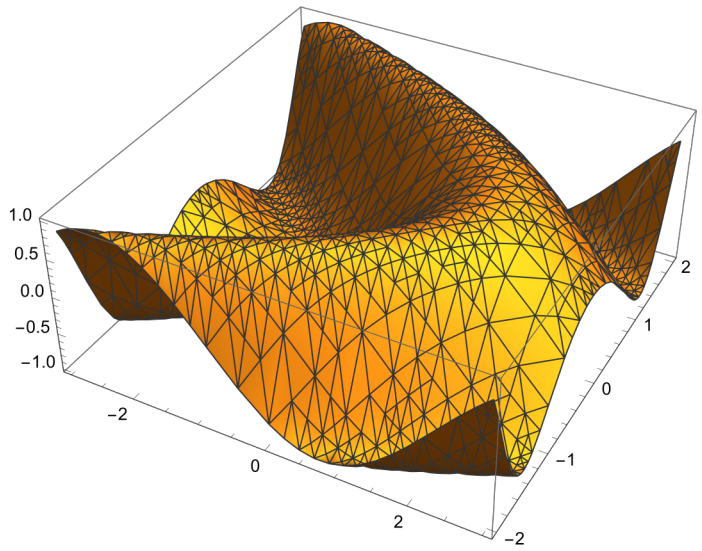
In[•]:=
```
Plot3D[Sin[x + y^2], {x, -2, 2}, {y, -2, 2}, RegionFunction → (1 < #1^2 + #2^2 < 4 &),
 Filling → Bottom, FillingStyle → Opacity[0.7], Mesh → None]
```

Out[•]=



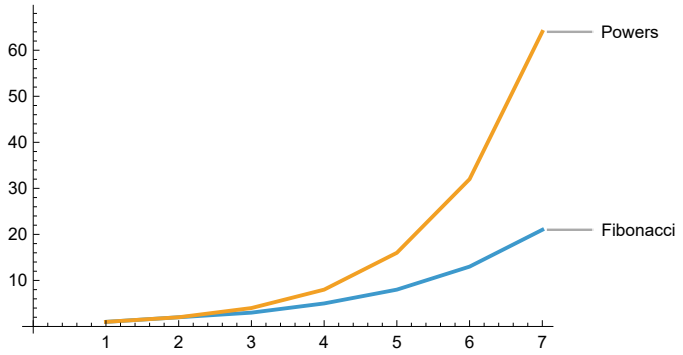
In[•]:=
```
Plot3D[Sin[x + y^2], {x, -3, 3}, {y, -2, 2}, Mesh → All]
```

Out[•]=



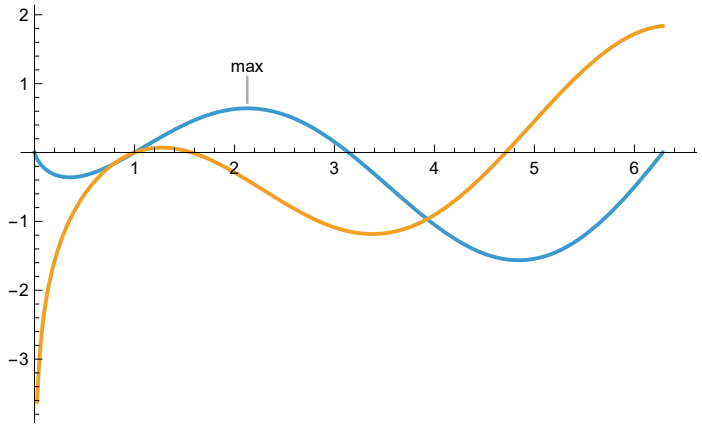
In[•]:=
```
ListLinePlot[{{1, 2, 3, 5, 8, 13, 21}, {1, 2, 4, 8, 16, 32, 64}},
 PlotLabels → {"Fibonacci", "Powers"}]
```

Out[•]=



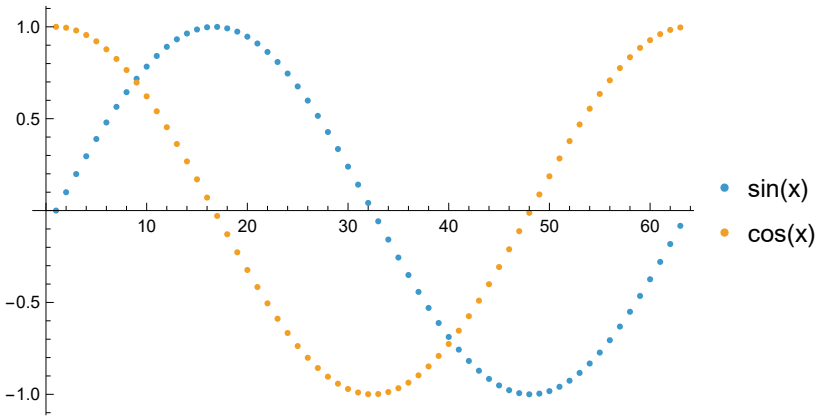
In[•]:=
```
Plot[{Callout[Log[x] Sin[x], "max", Above], Log[x] Cos[x]}, {x, 0, 2 π}]
```

Out[•]=



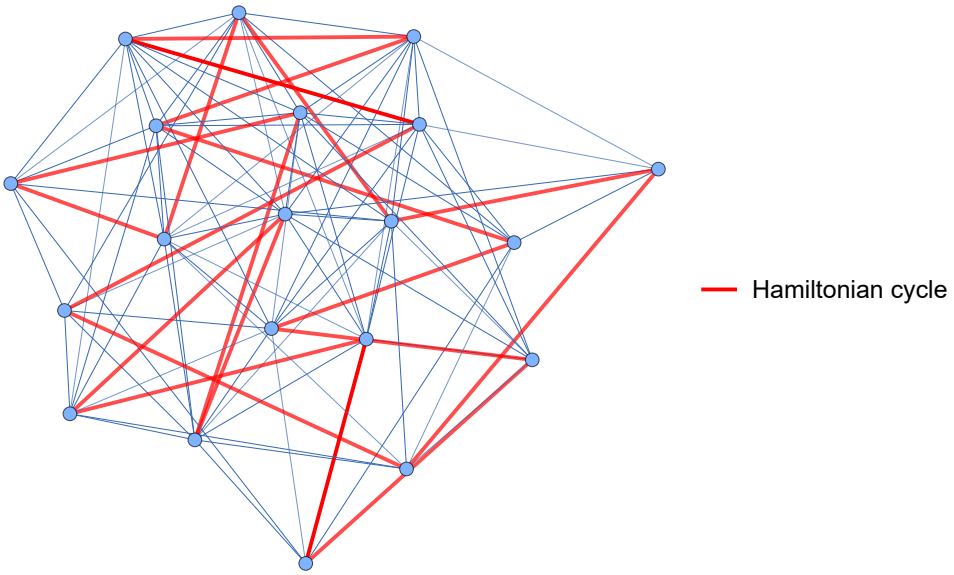
In[•]:=
```
ListPlot[Table[f, {f, {Sin[x], Cos[x]}}, {x, 0, 2 π, 0.1}],
 PlotLegends → {"sin(x)", "cos(x)"}]
```

Out[•]=



In[•]:=
```
g = RandomGraph[{20, 100}];
h = FindHamiltonianCycle[g];
Legended[HighlightGraph[g, Style[h, Directive[Thick, Red]]],
 LineLegend[{Directive[Thick, Red]}, {"Hamiltonian cycle"}]]
```

Out[•]=

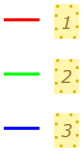


In[•]:=
```
LineLegend[{Red, Green, Blue}, {"red", "green", "blue"}]
```

Out[•]=

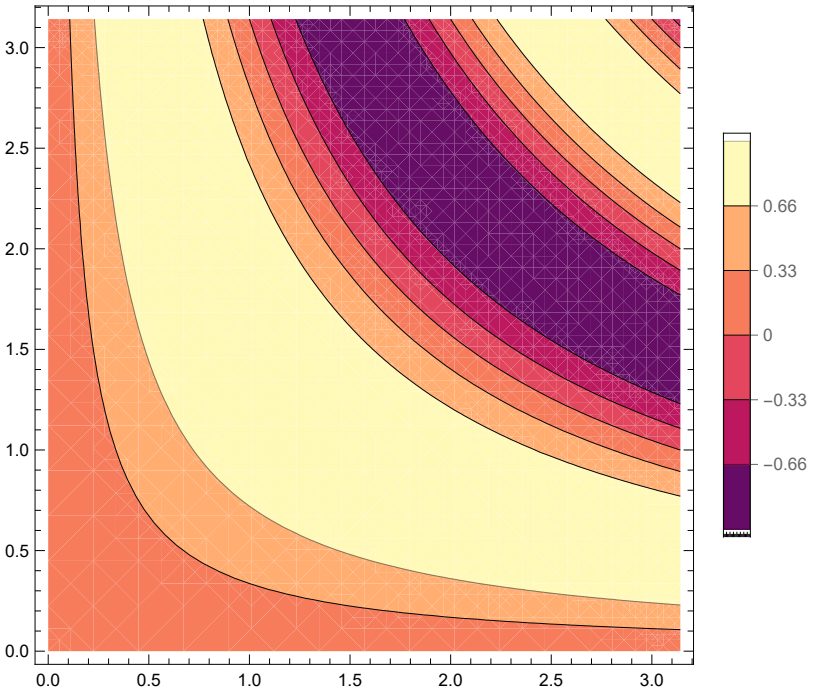
In[•]:=
```
LineLegend[{Red, Green, Blue}, Automatic]
```

Out[•]=



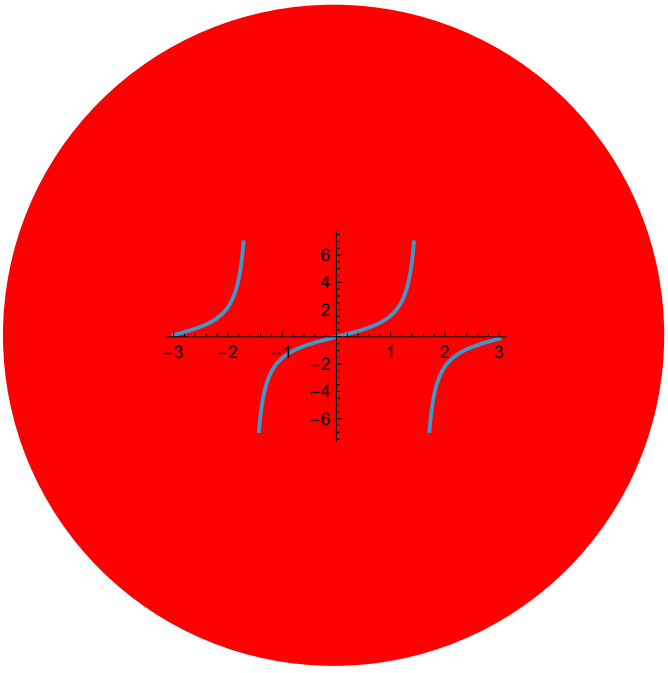
In[•]:=
```
ContourPlot[Sin[x y], {x, 0, π}, {y, 0, π}, Contours → 5, PlotLegends → Automatic]
```

Out[•]=



In[•]:=
```
Graphics[{Red, Disk[], Inset[Plot[Tan[x], {x, -3, 3}]]}]
```

Out[•]=



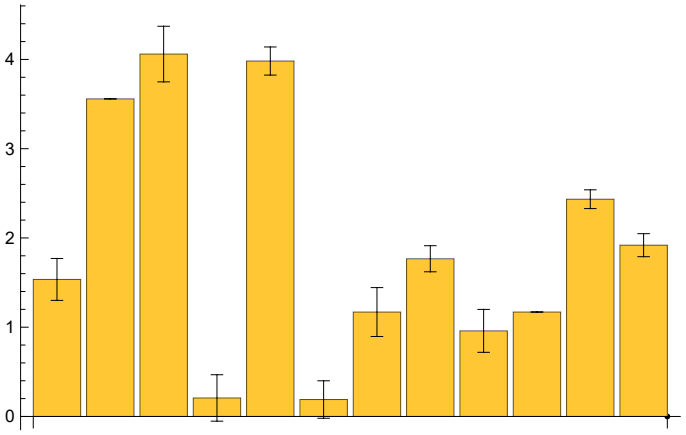
In[•]:=
```
Around[56.723, {0.18, 0.076}]
```

Out[•]=

$$56.72^{+0.08}_{-0.18}$$

In[•]:=
```
BarChart[Table[Around[RandomReal[5], RandomReal[0.5]], 12]]
```

Out[•]=



In[•]:=
```
EntityValue[Golden Gate Bridge BRIDGE, "LongestSpan"]
```

Out[•]=

4200. ft

In[•]:=
```
EntityRegister[
 EntityStore[RelationalDatabase[FindFile["ExampleData/ecommerce-database.sqlite"]]]]
```

Out[•]=

{productlines, customers, orderdetails, payments, employees, offices, orders, products}

In[•]:=
```
FormulaLookup["kinetic energy"]
FormulaData["KineticEnergy"]
```

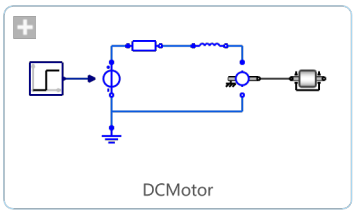
Out[•]=

{KineticEnergy, KineticEnergyRelativistic}

Out[•]=

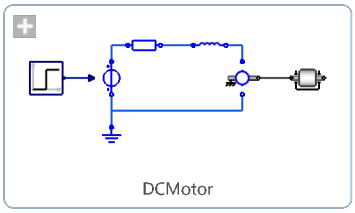
$$K == \frac{1}{2} m v^2$$

In[•]:=
```
SystemModel["IntroductoryExamples.MultiDomain.DCMotor"]
```

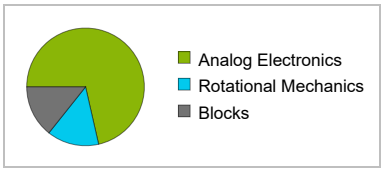


["Summary"]

Out[•]=



Out[•]=

ModelName	IntroductoryExamples.MultiDomain.DCMotor
Description	Dynamic model of a DC motor
SimulationModel	True
Plots	1
Components	7
Balanced	—
SystemEquations	—
SystemVariables	—
Domain	

In[•]:=
```
ThermodynamicData["Ammonia", "SoundSpeed"]
```

Out[•]=

429.403 m/s

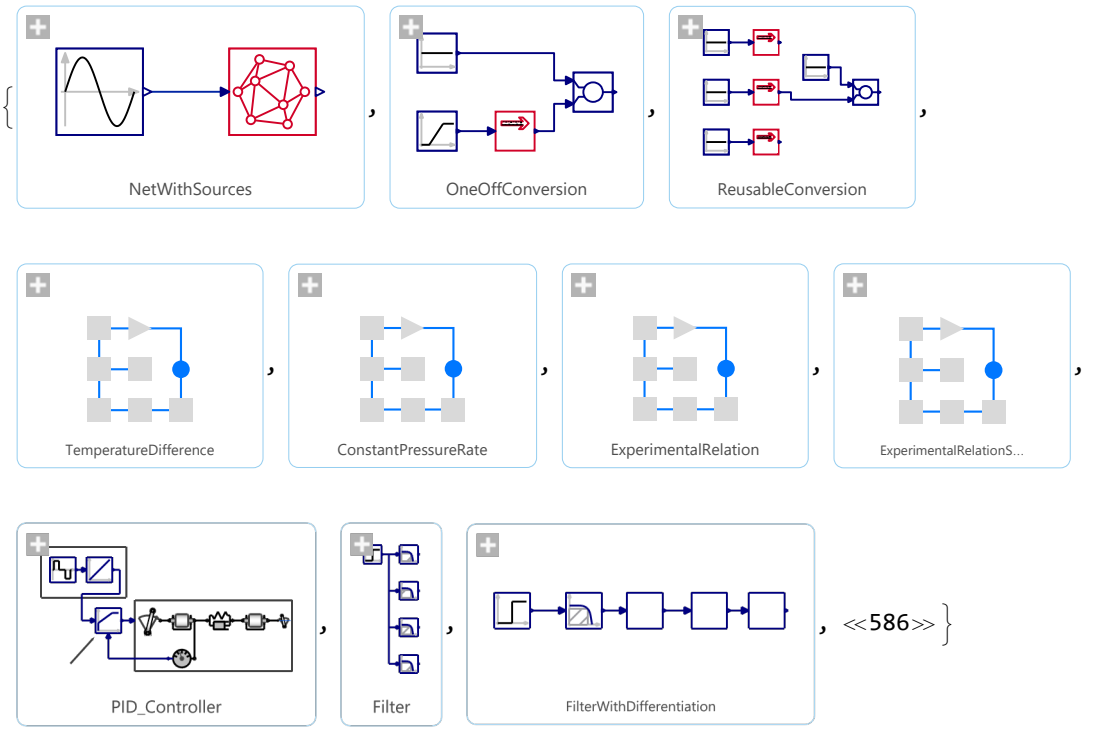
In[•]:=
```
PsychrometricPropertyData[<|"DryBulbTemperature" → Quantity[23, "DegreesCelsius"],
  "DewPointTemperature" → Quantity[14, "DegreesCelsius"],
  "Pressure" → Quantity[800, "Millibars"]|>, "HumidityRatio"]
```

Out[•]=

0.0127 lb/lb

In[•]:= **SystemModelExamples["Models"] // Shallow**

Out[•]//Shallow=



In[•]:= **FinancialData[All]**

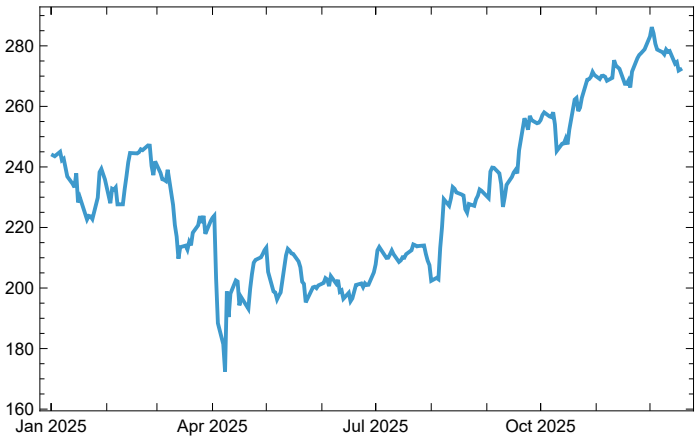
Out[•]=

{AMEX:ACCS, AMEX:ACU, AMEX:AEF, AMEX:AEON, AMEX:AIM, AMEX:AIRI, AMEX:AMBO, AMEX:AMS, AMEX:AMZE, AMEX:APT,
AMEX:APUS, AMEX:AREN, AMEX:ARMN, AMEX:ARMP, AMEX:ASM, AMEX:ATCH, AMEX:ATNM, AMEX:AUST, AMEX:AWX, AMEX:AXIL,
AMEX:AZTR, AMEX:BATL, AMEX:BCV, AMEX:BCV-PR-A, AMEX:BDL, AMEX:BGI, AMEX:BHB, AMEX:BHM, AMEX:BKTI,
AMEX:BMNR, AMEX:BQ, AMEX:BRBS, AMEX:BRIA, AMEX:BRN, AMEX:BTG, AMEX:BURU, AMEX:CANF, AMEX:CATX, AMEX:CCEL,
AMEX:CET, AMEX:CEV, AMEX:CIK, AMEX:CIX, AMEX:CKX, AMEX:CLDI, AMEX:CLM, AMEX:CMCL, AMEX:CMT, AMEX:CNL,
AMEX:COE, AMEX:COHN, AMEX:CPHI, AMEX:CRF, AMEX:CTGO, AMEX:CTM, AMEX:CVM, AMEX:CVR, AMEX:CVU, AMEX:CYBN,
AMEX:DC, AMEX:DC-WS, AMEX:DDC, AMEX:DHY, AMEX:DIT, AMEX:DNN, AMEX:DSS, AMEX:DVS, AMEX:DXF, AMEX:EAD,
AMEX:ECF, AMEX:ECF-PR-A, AMEX:EIM, AMEX:ELA, AMEX:ELLO, AMEX:ELMD, AMEX:EMX, AMEX:EONR, AMEX:EONR-WS,
AMEX:EP, AMEX:EPM, AMEX:EQX, AMEX:ERC, AMEX:ERH, AMEX:ESP, AMEX:EVI, AMEX:EVV, AMEX:EXOD, AMEX:FAX,
AMEX:FCO, ··· 58 301 ···, TSE:9941, TSE:9946, TSE:9948, TSE:9950, TSE:9955, TSE:9956, TSE:9959, TSE:9960,
TSE:9962, TSE:9969, TSE:9972, TSE:9973, TSE:9974, TSE:9976, TSE:9978, TSE:9979, TSE:9980, TSE:9982,
TSE:9983, TSE:9984, TSE:9986, TSE:9987, TSE:9989, TSE:9990, TSE:9991, TSE:9993, TSE:9994, TSE:9996,
TSE:9997, ^DJI, ^DJUSCH, ^DJUSRE, ^MID, ^SGX, ^SPX, ^NDX, ^COMP, ^NYA, ^SP500TR, ^XAX, ^000001, ^000002,
^000003, ^399106, ^399107, ^399108, ^AEX, ^AMX, ^ATX, ^BEL, ^BTK, ^BUX, ^BVSP, ^CSE, ^CSEC, ^DAX, ^EFEI,
^EFON, ^HSCEI, ^HSI, ^HUI, ^IBEX, ^JKSE, ^KLSE, ^KSI, ^KSIC, ^MDAX, ^MERV, ^MXX, ^NIFTY, ^NKY, ^NZD, ^OBX,
^PX1, ^RTS, ^SENSEX, ^SET50, ^SMI, ^SX5E, ^TOPX, ^TWII, ^VALUA, ^VALUG, ^VDAX, ^VSTX, ^XAO, ^XJO, ^XOI}

Full expression not available (original memory size: 2.6 MB)

In[ ]:=
```
DateListPlot[FinancialData["NASDAQ:AAPL", "Jan. 1, 2025"]]
```

Out[ ]=




```
* (WANE NO NB Assistant : ResourceFunction["InstallNotebookAssistant"][]) *
```