Drawing Statistical Charts
Extension Module

A guide to new visualization types

Adam Žilavý
Based on the original work by:
Tamara Kocurová, Adriana Kašparová, Tomáš Hála

Version 0.43
December 31, 2025

Contents

1 About the extension

This document serves as a user guide for the **Statistical Charts Extension** module for ConTEXt. It builds upon the original *statistical-charts* module created by Tamara Kocurová, Adriana Kašparová, and Tomáš Hála.
While the original module focuses on basic charts: Area, Bar, Column, Line, Pie, Radar, Scatter, Stock. This extension introduces advanced statistical visualizations widely used in data analysis.

1.1 Parts of the module

The extension consists of the following files:

- `t-statistical-charts-extension.mkiv` – contains the definition of TEX commands for the new chart types,
- `t-statistical-charts-extension.lua` – contains the complete implementation of drawing algorithms and calculations,
- `t-readdata.lua` – shared module used for reading and processing user data.

1.2 Usage

To use the extension in your document, simply load the module in your preamble. Since this is an extension, ensure all related files are in your ConTEXtpath.

```
\usemodule[statistical-charts-extension]
```

To display the chart properly, place it inside a standard ConTEXt figure environment:

```
\startplacefigure[title={...}, ...]
  ...
\stopplacefigure
```

1.3 New Supported Chart Types

This module adds the following chart types to the ConTEXt enviroment:

- **Histogram charts:** Visualizing frequency distribution of numerical data,
- **Boxplot charts:** Displaying data distribution based on a five-number summary,
- **Error bar charts:** Indicating error or uncertainty in measurements,
- **Step charts:** Emphasizing sudden changes in data at discrete intervals,
- **ECDF charts:** Empirical Cumulative Distribution Function visualizations,
- **KDE charts:** Kernel Density Estimation for smooth probability density curves,
- **Violin charts:** A hybrid of boxplots and kernel density plots,
- **Heatmap charts:** Visualizing data magnitude as color in 2D matrices,
- **Polar charts:** Plotting data on a circular coordinate system,
- **Sunburst charts:** Visualizing hierarchical data structures.

2 Histogram charts

Histogram is an approximate representation of the distribution of numerical data. Histogram is one of the most important graphical objects in statistical practice (Scott, 2009). To construct a histogram, the first step is to `bin` the range of values. That means, divide the entire range of values into a series of intervals and then count how many values fall into each interval. Unlike a bar chart, which relates to two variables – categories and values, a histogram relates to only one variable – data distribution. The height of the bar represents the frequency of data points within that bin. The subtypes of the histogram chart are:

- basic,
- stacked,
- grouped.

2.1 Histogram chart parameters

The following parameters (and their default values) can be set in the histogram chart.

To work with the axes:

xlength: length of x-axis
ylength: length of y-axis
padding: additional space around the minimum and maximum data values on the x-axis

To set the bins and data range:

bins: specific number of bins to divide the data into. If not set, it is calculated automatically using the square root rule ($k = \lceil \sqrt{n} \rceil$).
bandwidth: specific width of the bins. If set, this overrides the `bins` parameter.
min: starting value for the first bin for aligning bins into nice numbers.
max: ending value for the calculation.
boundaries: determines how values on the edge of intervals are handled. Can be set to `left` or `right`. If set to `right`, a value equal to the bin edge is included in the lower bin (e.g., 75 falls into 70–75).

To set the fill:

fillcolor: color of the bars
fillpalette: palette of colors used for stacked histograms
filltransparency: transparency of the bars

To set the grid and labels:

grid: on/off
xgridlines: yes/no
ygridlines: yes/no

xaxislabels: yes/no
yaxislabels: yes/no
axesunits: yes/no
xunit: label for the x-axis
yunit: label for the y-axis

2.2 Basic histogram

The basic histogram visualizes the frequency distribution of a single dataset. The x-axis represents the data ranges bins, and the y-axis represents the frequency of data points falling into those bins.

The example in ***Table 1.1*** shows the distribution of heights of trees sample. The data is provided as a simple list of numbers: 61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87

Height Range (cm)	Number of Trees
60–65	3
65–70	3
70–75	8
75–80	10
80–85	5
85–90	1

Table 2.1 Distribution of tree height according to measured intervals.

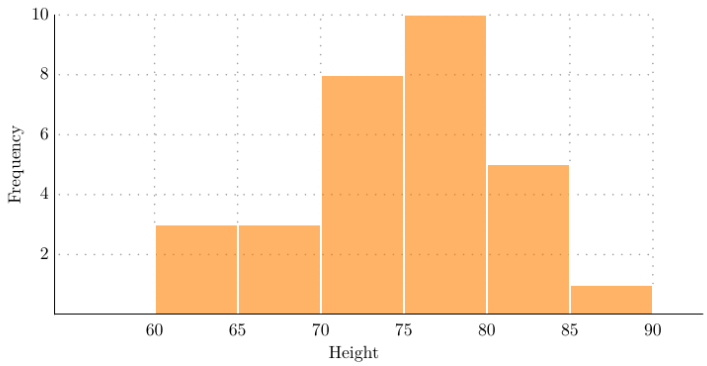

Chart 2.1 Basic histogram with one data series.

In this example, `bandwidth` is set to 5 to create 5 cm intervals. The `min` parameter forces the bins to start at 60, and `boundaries=right` ensures that a value like 75 is counted in the 70–75 interval. The ***Chart 1.1*** was created using the following commands:

```
\def\trees{61, 63, 64, 66, 68, 69, 71, 72, 72, 73, 73, 74, 74, 75, 76,
           76, 77, 77, 78, 78, 79, 79, 79, 80, 81, 82, 83, 84, 85, 87}

\histogramchart[basic][
    xlength=12,ylength=6,
    min=60,
    bandwidth=5,
    boundaries=right,
    fillcolor=orange,
    filltransparency=0.6,
    grid=on,
    xaxislabels=yes,
    yaxislabels=yes,
    padding=0.1,
    axesunits=yes,
    xunit={Height},
    yunit={Frequency}][data={\trees}]
```

2.3 Stacked histogram

The stacked histogram is useful when the dataset contains multiple series that need to be compared within the same distribution ranges. The bars are stacked on top of each other, showing the cumulative frequency for each bin while retaining the contribution of each individual series.
(Yi, 2025) This type requires structured data.
The example in ***Chart 1.2*** shows production time in minutes for three different shifts.

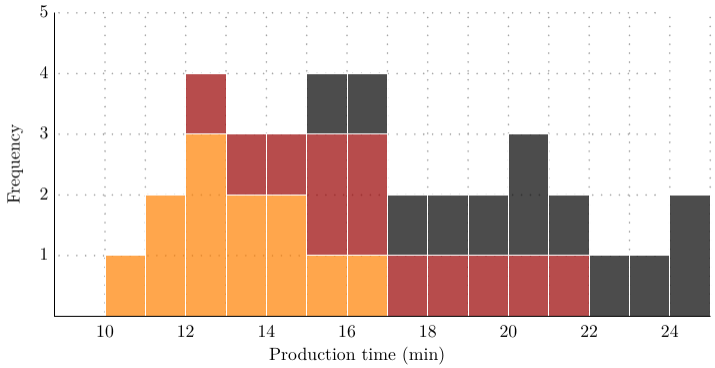

Chart 2.2 Stacked histogram with multiple data series.

The ***Chart 1.2*** was created by using the commands below.

```
\definepalette[autumn][orange, darkred, brown]

\def\hours{{10, 11, 11, 12, 12, 12, 13, 13, 14, 14, 15, 16},
           {12, 13, 14, 15, 15, 16, 16, 17, 18, 19, 20, 21},
           {15, 16, 17, 18, 19, 20, 20, 21, 22, 23, 24, 25}}

\histogramchart[stacked][
    xlength=12,ylength=6,
    min=10,
    bandwidth=2,
    fillpalette=autumn,
    filltransparency=0.7,
    linecolor=white,
    linewidth=0.5,
    grid=on,
    xaxislabels=yes,
    yaxislabels=yes,
    padding=0,
    axesunits=yes,
    xunit={Production time (min)},
    yunit={No. of pieces}][data={\hours}]
```

2.4 Grouped histogram

The grouped histogram displays multiple data series as adjacent bars within each bin. This allows for direct comparison of frequency distribution across different groups for the same value ranges, rather than cumulating them as in the stacked histogram.
The following example compares the same production time data as above, but uses a grouped layout.

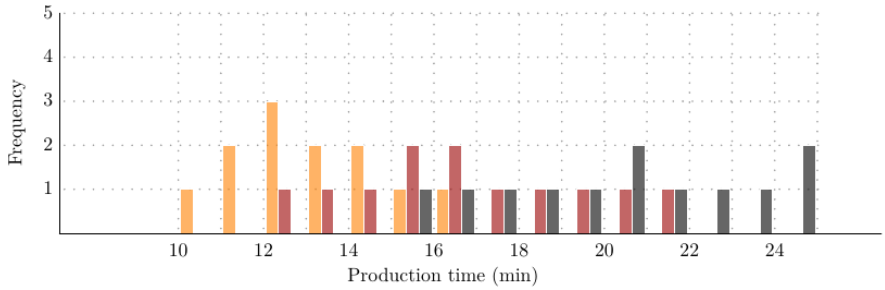

Chart 2.3 Grouped histogram with multiple data series.

```
\definepalette[autumn][orange, darkred, brown]

\def\hours{{10, 11, 11, 12, 12, 12, 13, 13, 14, 14, 15, 16},
           {12, 13, 14, 15, 15, 16, 16, 17, 18, 19, 20, 21},
           {15, 16, 17, 18, 19, 20, 20, 21, 22, 23, 24, 25}}

\histogramchart[grouped][
    xlength=14,ylength=6,
    min=10,
    bandwidth=1,
    fillpalette=autumn,
    filltransparency=1,
    linecolor=black,
    linewidth=0.5,
    grid=on,
    xaxislabels=yes,
    yaxislabels=yes,
    padding=0.1,
    axesunits=yes,
    xunit={Value},
    yunit={Frequency}][data={\hours}]
```

3 Boxplot charts

A boxplot displays the distribution of data based on a five-number summary: minimum, first quartile ($Q1$), median, third quartile ($Q3$), and maximum. It is statistically robust method to visualize the spread and skewness of data.(GeeksForGeeks, 2025)

One of the key features of this chart is the automatic detection of outliers. The module calculates the Interquartile Range ($IQR = Q3 - Q1$) and identifies any data points falling below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$ as outliers.(GeeksForGeeks, 2025)
The currently available subtype is:

- basic (with option to plot multiple boxes)

3.1 Boxplot chart parameters

The following parameters (and their default values) can be set in the boxplot chart.

To work with the axes:

xlength: length of x-axis
ylength: length of y-axis
padding: additional space around the minimum and maximum data values on the y-axis

To work with the box and whiskers:

boxwidth: width of the box relative to the distance between items
boxcolor: fill color of the box
boxpalette: palette of colors used when multiple groups are displayed
boxtransparency: transparency of the box fill
whiskerwidth: width of the horizontal lines at the end of whiskers

To work with lines and median:

linecolor: color of the whiskers and box borders
linewidth: thickness of the lines
mediancolor: color of the median line inside the box
medianwidth: thickness of the median line

To work with outliers:

outliers: determines if outliers are shown as dots
outliercolor: color of the outlier dots
outlierwidth: diameter of the outlier dots

To set the grid and labels:

grid:	on/off
ygridlines:	yes/no
xaxislabels:	yes/no
yaxislabels:	yes/no
axesunits:	yes/no
xunit:	label for the x-axis
yunit:	label for the y-axis

3.2 Basic boxplot with single series

The basic boxplot calculates statistics automatically from the provided raw data list. The example in ***Chart 2.1*** visualizes a dataset representing tha age in group A.

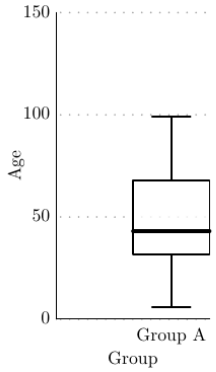

Chart 3.1 Boxplot with `outliers=no`.

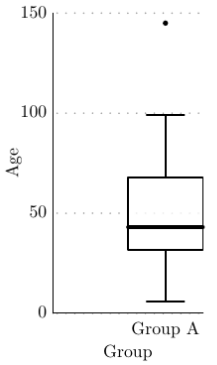

Chart 3.2 Boxplot with `outliers=yes`.

The ***Chart 2.1*** and ***Chart 2.2*** were created using the following commands:

```
\def\agegroup{32, 55, 33, 34, 43, 52, 6, 60, 70, 42,
              42, 66, 23, 31, 38, 48, 29, 22, 78, 32,
              54, 26, 77, 83, 77, 92, 23, 99, 54, 22, 145}

\boxplotchart[basic][
    xlength=3, ylength=6,
    ygridlines=yes,
    boxcolor=white,
    mediancolor=black,
    outliers=no,
    linewidth=1,
    axesunits=yes,
    xunit={Group},
    yunit={Age},]
    [xlabels={Group A}, data={\agegroup}]
```

Note: The only difference in the commands for those charts is that for ***Chart 2.1*** it is `outliers=no` and for ***Chart 2.2*** it is `outliers=yes`.

3.3 Multiple boxplots

Boxplots are ideal for comparing distributions across multiple groups side-by-side. You can provide data as a list of lists (e.g., `data={{...}, {...}}`).
The example in ***Chart 2.3*** compares recovery times for three different medical treatments. By using `boxpalette`, each box is assigned a different color from the defined palette. The `distance` parameter controls the spacing between the boxes.

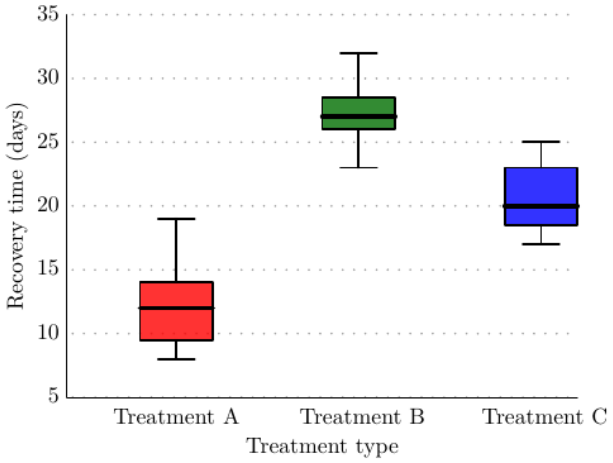

Chart 3.3 Grouped boxplot with multiple data series.

The ***Chart 2.3*** was created by using the commands below.

```
\definepalette[mycolors][red, green, blue]

\boxplotchart[basic][
    xlength=8, ylength=6,
    ygridlines=yes,
    boxwidth=1,
    distance=1.5,
    outliers=no,
    boxpalette=mycolors,
    axesunits=yes,
    xunit={Treatment type},
    yunit={Recovery time (days)},]
    [xlabels={Treatment A, Treatment B, Treatment C},
     data={{10, 12, 8, 15, 9, 13, 19},
           {27, 25, 29, 32, 28, 23, 27},
           {20, 18, 22, 24, 19, 17, 25}}]
```

4 Error bar charts

Error bars illustrate the margin of error for a survey estimate by showing how precise that estimate is. (Smith, 2025) They give a general idea of how precise a measurement is or how far from the reported value the true value might be.
In this module, the error bar chart visualizes the mean value as a point and the variability as a vertical line capped with horizontal lines. The chart can also connect the mean values with a line to show trends.

4.1 Error bar chart parameters

The following parameters (and their default values) can be set in the error bar chart. The chart also accepts standard axis and grid parameters defined in previous chapters.

To work with the axes:

xlength: length of x-axis
ylength: length of y-axis
padding: additional space around the minimum and maximum data values on the y-axis

To work with the error bars and dots:

errorbarcolor: color of the vertical error lines and caps
errorbarwidth: thickness of the error lines in pt
capwidth: width of the horizontal caps at the end of error bars relative to the distance between items
dots: determines if the mean value points are shown
dotscolor: color of the dots representing the mean value
dotswidth: diameter of the dots in `pt`

To work with the connecting line:

linecolor: color of the line connecting mean values
linewidth: thickness of the connecting line
linetransparency: transparency of the connecting line

To set the data:

data: list of mean values
errors: list of error margins according to the data values.

To set the grid and labels:

grid: on/off
xgridlines: yes/no
ygridlines: yes/no

xaxislabels: yes/no
yaxislabels: yes/no
axesunits: yes/no
xunit: label for the x-axis
yunit: label for the y-axis

4.2 Basic error bar chart

The basic error bar chart visualizes a dataset where each point represents a mean value, and the vertical bars represent the margin of error or standard deviation.
The example in ***Table 3.1*** shows the effect of different fertilizers on plant growth. The data corresponds to the ***Chart 3.1*** below.

Fertilizer Type	Mean Height (cm)	Error (± cm)	Value Range (Min – Max)
Control	12	1	11 – 13
Fertilizer A	18	6	12 – 24
Fertilizer B	22	3	19 – 25

Table 4.1 Impact of fertilizer on plant growth

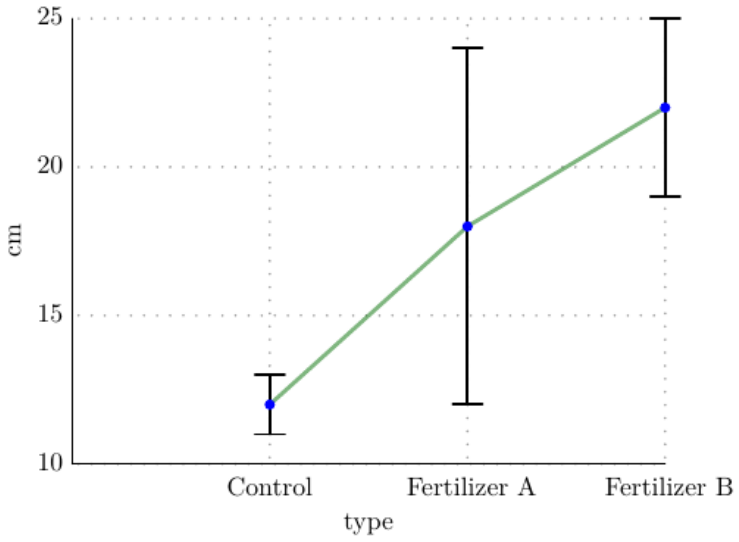

Chart 4.1 Error bar chart showing plant growth variability.

The ***Chart 3.1*** was created using the following commands:

```
\errorbarchart[basic][
    xlength=8,ylength=6,
    linecolor=green,
    grid=on,
    axesunits=yes,
    yunit=cm,
    xunit=Type,][
    xlabels={Control, Fertilizer A, Fertilizer B},
    data={12, 18, 22},
    errors={1, 6, 3}]
```

Error bars are also useful for time-series data to show how values and their uncertainty change over time. The module automatically scales the y-axis to accommodate the full range of the error bars.
The example in ***Table 3.2*** and ***Chart 3.2*** visualizes measured in span of four days.

Day	Speed (m/s)	Error (± m/s)	Value Range
Day 1	1013	2	1011 – 1015
Day 2	1005	3	1002 – 1008
Day 3	1020	2	1018 – 1022
Day 4	1010	1	1011 – 1009

Table 4.2 Measured speed over time

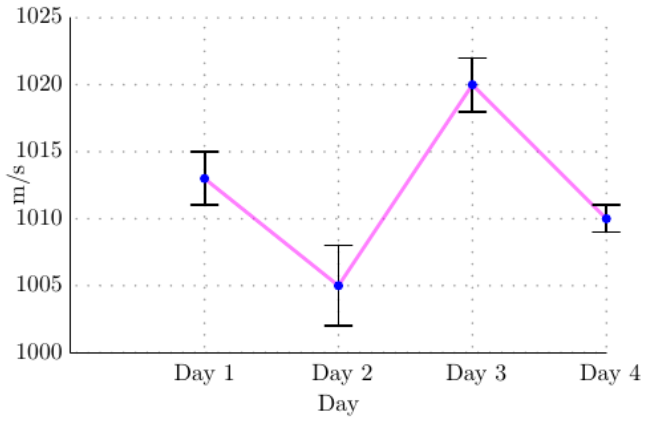

Chart 4.2 Error bar chart showing speed fluctuations.

The ***Chart 13.2*** was created using the following commands:

```
\errorbarchart[basic][
    xlength=8,ylength=5,
    linecolor=magenta,
    grid=on,
    axesunits=yes,
    yunit=m/s,
    xunit=Day,][
    xlabels={Day 1, Day 2, Day 3, Day 4},
    data={1013, 1005, 1020, 1010},
    errors={2, 3, 2, 1}]
```

5 Step charts

A step chart is a variation of a line chart that connects data points using a combination of vertical and horizontal lines, forming a step-like progression. Unlike a standard line chart, which connects points with the shortest distance, a step chart emphasizes that the value remains constant for a period of time until a sudden change occurs. This type of chart is particularly useful for visualizing data that changes at discrete intervals, such as interest rates, inventory levels, or daily production quotas.
The subtype of the step chart is:

- basic (with option to plot multiple step charts)

5.1 Step chart parameters

The following parameters (and their default values) can be set in the step chart.

To work with the axes:

xlength: length of x-axis
ylength: length of y-axis

To set the stepping line:

stepping: defines when the step occurs. Options are:
`after` – line remains horizontal after and steps vertically at the next point.
`before` – line steps vertically immediately and then goes horizontal.
stepmark: yes/no
stepmarkcolor: color of the step marks
stepmarkwidth: size of the step marks
linecolor: color of the step line
linepalette: palette of colors used when multiple series are displayed
linewidth: thickness of the line
linetransparency: transparency of the line

To set the dots:

dots: yes/no
dotscolor: color of the data points
dotspalette: palette of colors for dots in multiple series
dotswidth: diameter of the dots

To set the grid and labels:

grid: on/off
xgridlines: yes/no
ygridlines: yes/no

xaxislabels: yes/no
yaxislabels: yes/no
axesunits: yes/no
xunit: label for the x-axis
yunit: label for the y-axis

5.2 Basic step chart (Stepping After)

In the default configuration (`stepping=after`), the chart indicates that a value persists until a new value is recorded. Example table ***Table 4.1*** displays measured vales for every day. The data are shown in ***Chart 4.1*** for daily production figures, where the step occurs after the day is completed.

Mon	Tue	Wed	Thu	Fri	Sat
10	25	15	30	45	40

Table 5.1 Distribution of data over days

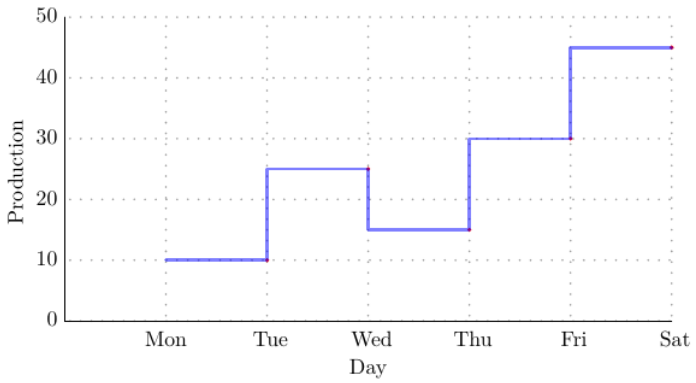

Chart 5.1 Basic step chart with `stepping=after`.

The ***Chart 4.1*** was created using the following commands:

```
\stepchart[basic][
    xlength=10,ylength=5,
    stepping=after,
    linecolor=blue,
    grid=on,
    padding=0.1,
    stepmark=yes,
    dots=no,
    axesunits=yes,
    xunit=Day,
    yunit=Production][
    xlabels={Mon, Tue, Wed, Thu, Fri, Sat},
    data={10, 25, 15, 30, 45, 45}]
```

5.3 Basic step chart (Stepping Before)

The appearance of the chart can be altered significantly by changing the `stepping` parameter to `before`. This is useful when the change in value happens immediately at the beginning of a period. The *Table 4.2* displays price evolution:

Jan	Feb	Mar	Apr	May	Jun
15	25	20	35	45	40

Table 5.2 Distribution of data over months

The *Chart 4.2* illustrates values from *Table 4.2* with parameter `dots=yes`. This allows dots to be shown for every beginning value.

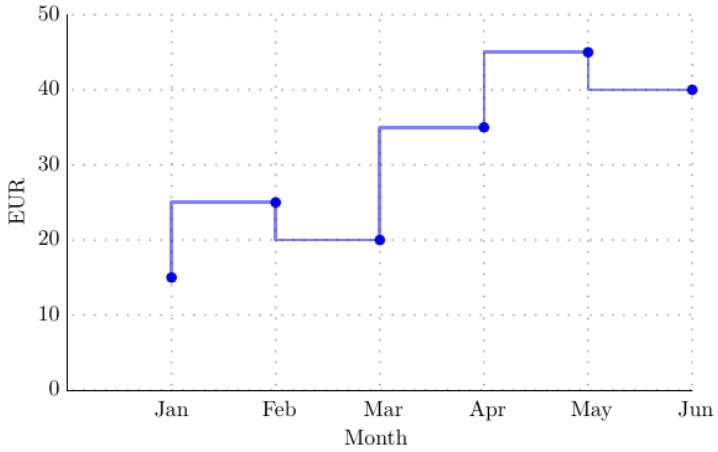

Chart 5.2 Step chart with `stepping=before` and `dots=yes`.

The *Chart 4.2* was created using the following commands:

```
\stepchart[basic][
    xlength=10,ylength=6,
   stepping=before,
    linecolor=blue,
    linewidth=3,
    linetransparency=0.4,
    dots=yes
    dotscolor=darkblue,
    dotswidth=5,
    grid=on,
    axesunits=yes,
    yunit=EUR][
    xlabels={Jan, Feb, Mar, Apr, May, Jun},
    data={15, 25, 20, 35, 45, 40}]
```

5.4 Multiple data series

Step charts can also display multiple data series simultaneously to compare different categorical progressions. When using multiple series, it is recommended to use palettes for lines and dots to automatically distinguish between the data sets. The example in ***Table 4.3*** compares power performance across four different measured periods.

Series	Q1	Q2	Q3	Q4
Red	10	15	20	25
Blue	30	28	35	32
Green	5	8	6	12
Orange	40	45	10	15

Table 5.3 Performance overview by Quarter

The example in ***Chart 4.3*** displays data from ***Table 4.3***.

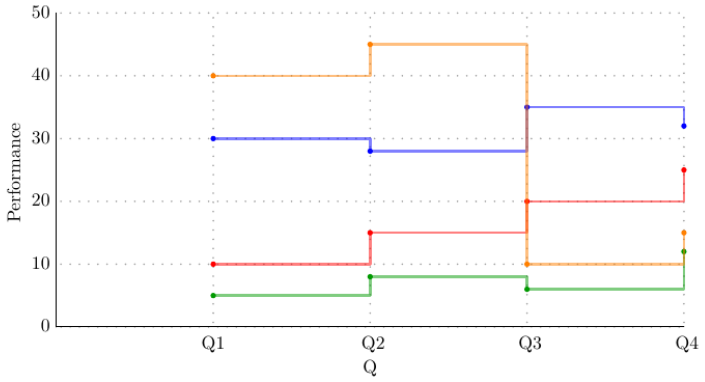

Chart 5.3 Step chart with multiple data series.

The ***Chart 4.3*** was created using the commands below. To distinguish line colors there we created custom palette `fourlines`.

```
\definepalette[fourlines][red, blue, darkgreen, orange]

\stepchart[basic][
    xlength=12,ylength=6,
    stepping=after,
    linepalette=fourlines,
    dotspalette=fourlines,
    linewidth=1.5,
    dots=yes,
    dotswidth=3,
    grid=on,
    axesunits=yes,
    xunit=Q,
    yunit=Performance][
    xlabels={Q1, Q2, Q3, Q4},
    data={{10, 15, 20, 25},
          {30, 28, 35, 32},
          { 5,  8,  6, 12},
          {40, 45, 10, 15}}]
```

6 ECDF charts

An Empirical Cumulative Distribution Function (ECDF) chart is a step function that visualizes the distribution of observations in a dataset. Unlike a histogram, which bins data into intervals, the ECDF plots each data point individually, showing the proportion of observations less than or equal to a specific value. The y-axis typically represents the cumulative probability (from 0 to 1) or the cumulative count.
The subtype of the ECDF chart is:

- basic (with option to plot multiple ECDF charts)

6.1 ECDF chart parameters

The following parameters (and their default values) can be set in the ECDF chart.

To work with the axes:

xlength: length of x-axis
ylength: length of y-axis
left: left margin of the chart.
padding: additional space around the minimum and maximum data values on the x-axis
showprobs: yes/no. Whether the y-axis displays probabilities (0.0–1.0) or absolute values.

To set the line style:

linecolor: color of the step line
linepalette: palette of colors used when multiple series are displayed
linewidth: thickness of the line
linetransparency: transparency of the line

To set the grid and labels:

grid: on/off
xgridlines: yes/no
ygridlines: yes/no
xaxislabels: yes/no
yaxislabels: yes/no
axesunits: yes/no
xunit: label for the x-axis
yunit: label for the y-axis

6.2 Basic ECDF chart

The basic ECDF chart sorts the input data automatically and plots the step function. The example in ***Chart X.1*** visualizes the distribution of response times. The y-axis automatically scales from 0 to 1 to represent probability.
Following sample data are used to display ***Chart 5.1***.

```
\def\sampledata{12, 15, 18, 22, 22, 23, 25, 28, 30, 33, 35, 40, 45, 10,
                50, 52, 48, 47, 36, 38, 41, 39, 40, 42, 43, 44, 45, 45}
```

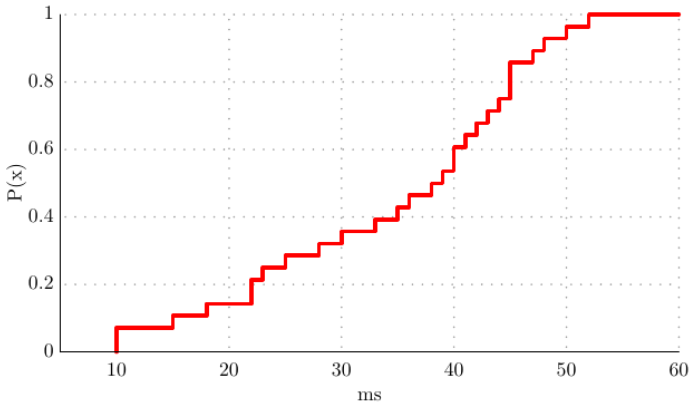

Chart 6.1 Basic ECDF chart with defined units.

The ***Chart 5.1*** was created using the following commands:

```
\ecdfchart[basic][
    xlength=10,ylength=6,
    left=1,
    linecolor=red,
    linewidth=2,
    grid=on,
    axesunits=yes,
    xunit=ms,
    yunit=P(x)][data={\sampledata}]
```

6.3 Multiple data series

The ECDF chart can also compare multiple distributions simultaneously. When multiple lists of data are provided, the module plots a separate curve for each dataset. Using `linetransparency` is recommended when curves overlap significantly.
The example in ***Chart 5.2*** compares three different distributions using a custom color palette

```
\definepalette[mypalette][blue, red, green]
```

and the use of this data. Please note that the size of individual data samples varies.

```
\def\multiplesampledata{{10, 12, 15, 18, 20, 25, 30},
                        {15, 18, 22, 25, 28, 35, 40, 45, 50},
                        {5, 8, 10, 12, 12, 14, 15}}
```

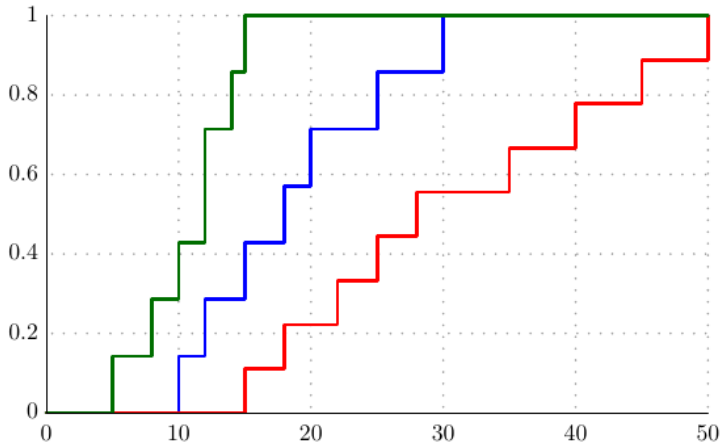

Chart 6.2 Comparison of three distributions using ECDF.

The ***Chart 5.2*** was created using the commands below:

```
\ecdfchart[basic][
    xlength=10,ylength=6,
    showprobs=yes,
    linecolor=darkblue,
    linepalette=mypalette,
    left=0][data={\multiplesampledata}]
```

7 KDE charts

A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset. KDE represents the data using a continuous probability density curve in one or more dimensions. (Waskom, 2024) It is often described as the smoothed *"brother"* of the histogram. While a histogram bins data into discrete intervals (buckets), resulting in a blocky appearance, the KDE chart calculates a continuous curve using a kernel function – typically Gaussian, to represent the data distribution. (Kumari, 2025) This makes it particularly useful for visualizing the shape of data without being influenced by the number of bins or their start points.
The subtype of the KDE chart is:

- basic (with option to plot multiple KDE charts)

7.1 KDE chart parameters

The following parameters (and their default values) can be set in the KDE chart.

To work with the axes:

xlength: length of x-axis
ylength: length of y-axis
padding: additional space around the minimum and maximum data values on the x-axis

To set the calculation:

bandwidth: controls the smoothness of the curve. If not set, it is calculated automatically based on the variance and size of the dataset.
resolution: the number of points used to calculate the curve

To set the visual style:

linecolor: color of the density curve line
linepalette: palette of colors used for multiple data series
linewidth: thickness of the line
fill: yes/no
fillcolor: color of the area fill
fillpalette: palette of fill colors for multiple series
filltransparency: transparency of the filled area

To set the grid and labels:

grid: on/off
xgridlines: yes/no
ygridlines: yes/no
xaxislabels: yes/no
yaxislabels: yes/no. *Note: In KDE, the y-axis represents calculated density.*

axesunits: yes/no
xunit: label for the x-axis
yunit: label for the y-axis

7.2 Basic KDE chart

The basic KDE chart visualizes the density distribution of a single dataset. By default, the module calculates an optimal bandwidth to create a smooth representation of the data.
The example in ***Chart 6.1*** uses the exact same tree height data as the Histogram example in ***Chart 1.1***. Comparing the two allows you to see how KDE provides a continuous alternative to the binned histogram.

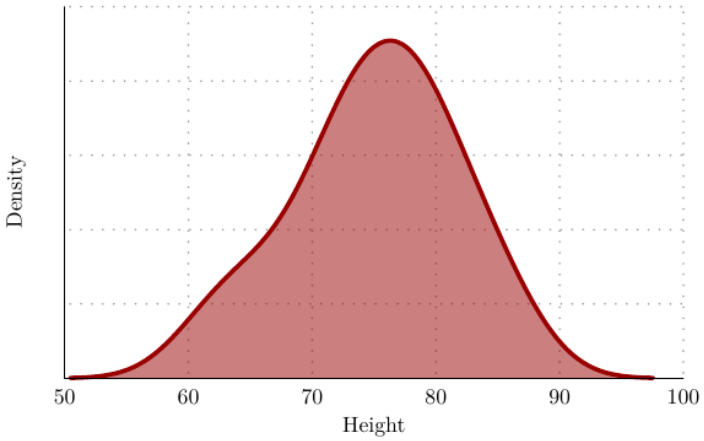

Chart 7.1 Basic KDE chart showing tree height distribution.

The ***Chart 6.1*** was created using the following commands:

```
\def\trees{61, 63, 64, 66, 68, 69, 71, 72, 72, 73, 73, 74, 74, 75, 76,
           76, 77, 77, 78, 78, 79, 79, 79, 80, 81, 82, 83, 84, 85, 87}

\kdechart[basic][
    xlength=10,ylength=6,
    fillcolor=darkred,
    filltransparency=0.5,
    linecolor=darkred,
    linewidth=2,
    grid=on,
    padding=0.1,
    axesunits=yes,
    xunit=Height,][data={\trees}]
```

7.3 Basic KDE with manual bandwidth

The shape of the KDE curve is highly dependent on the `bandwidth` parameter. A higher bandwidth results in a smoother curve, while a lower bandwidth reveals more details (potentially including noise). You can override the automatic calculation by setting this parameter manually.

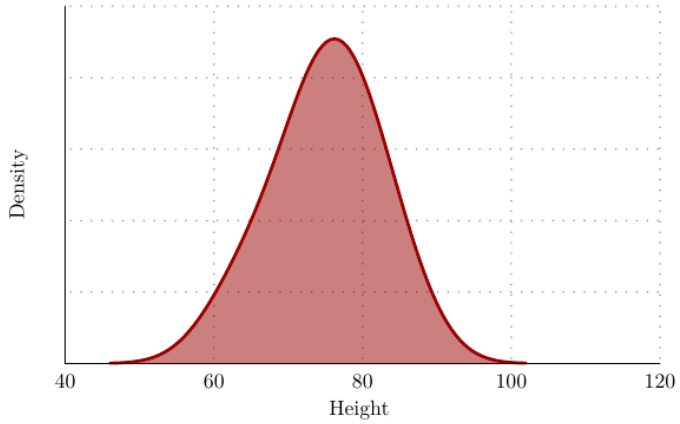

Chart 7.2 KDE chart with manual bandwidth showing the effect of an outlier.

The following example ***Chart 6.2*** demonstrates a dataset with a significant outlier (value 100) and a manually fixed bandwidth of 5.
The ***Chart 6.2*** was created using the commands below:

```
\kdechart[basic][
    xlength=10,ylength=6,
    bandwidth=5,
    fillcolor=darkred,
    linecolor=darkred,
    axesunits=yes,
    xunit=Height][data={\trees}]
```

7.4 Multiple data series

Just like histograms, KDE charts can compare multiple distributions. However, because KDE uses transparency and curves rather than stacked bars, it is often easier to visually compare overlapping data. Using `filltransparency` is crucial here to ensure all curves remain visible where they overlap.

The example in ***Chart 6.3*** uses the same production time data as the stacked histogram in ***Chart 1.2***.

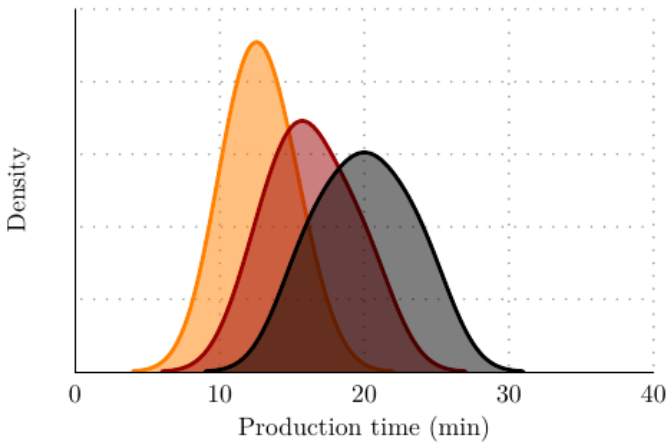

Chart 7.3 Multiple KDE plots comparing production times.

The ***Chart 6.3*** was created using the commands below:

```
\definepalette[autumn][orange, darkred, brown]

\def\hours{{10, 11, 11, 12, 12, 12, 13, 13, 14, 14, 15, 16},
           {12, 13, 14, 15, 15, 16, 16, 17, 18, 19, 20, 21},
           {15, 16, 17, 18, 19, 20, 20, 21, 22, 23, 24, 25}}

\kdechart[basic][
    xlength=12,ylength=6,
    fillpalette=autumn,
    linepalette=autumn,
    filltransparency=0.5,
    bandwidth=2,
    linewidth=1.5,
    grid=on,
    axesunits=yes,
    xunit={Production time},
    yunit={Density}][data={\hours}]
```

8 Violin charts

A violin plot is a method of plotting numeric data that can be considered a hybrid of a box plot and a kernel density plot. While a box plot only shows summary statistics, the violin plot reveals the full distribution of the data. (Waskom, 2024) The "violin" shape is created by a mirrored Kernel Density Estimation plotted along the data axis. This chart is particularly useful for comparing the probability density of data at different values and visualizing the distribution shape (peaks, valleys, and skewness) alongside statistical summaries.
The subtypes of the violin chart are:

- basic (with options to create without boxplot and multiple plots)

8.1 Violin chart parameters

The following parameters (and their default values) can be set in the violin chart.

To work with the axes:

xlength: length of x-axis
ylength: length of y-axis
padding: additional space around the minimum and maximum data values on the y-axis
distance: distance between individual violins when multiple series are plotted

To set the violin shape (density):

violinwidth: relative width of the violin shape – controls how "fat" the violin appears
resolution: number of points used to calculate the smooth density curve
bandwidth: controls the smoothness of the density curve
fillcolor: fill color of the violin shape
filltransparency: transparency of the violin fill
stroke: yes/no
strokecolor: color of the violin outline
strokewidth: thickness of the violin outline

To set the internal boxplot:

showboxplot: yes/no
boxwidth: width of the internal box relative to the violin
boxcolor: color of the box and whiskers
mediancolor: color of the dot/line representing the median
medianwidth: size of the median marker

To set the grid and labels:

grid: on/off
xaxislabels: yes/no

yaxislabels: yes/no
axesunits: yes/no
xunit: label for the x-axis
yunit: label for the y-axis

8.2 Basic violin with boxplot

The standard violin chart displays the density distribution with an overlay of statistical information (min, max, quartiles, and median). The example in ***Table 7.1*** shows the test scores of students in "Class A". The table lists the individual scores, which are then visualized in ***Chart 7.1***. The internal black box represents the interquartile range (IQR), and the white dot represents the median.

Student	Class A
1	15
2	18
3	19
4	20
5	20
6	21
7	21
8	22
9	23
10	25
11	28
12	30

Table 8.1 Distribution of scores in Class A.

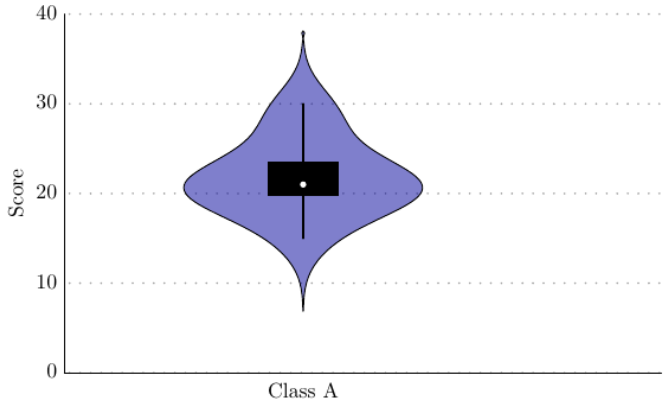

Chart 8.1 Violin chart with internal boxplot.

The ***Chart 7.1*** was created using the following commands:

```
\violinchart[basic][
    xlength=8,ylength=6,
    fillcolor=darkblue,
    filltransparency=0.6,
    violinwidth=1.0,
    showboxplot=yes,
    boxwidth=0.3,
    boxcolor=black,
    mediancolor=white,
    axesunits=yes,
    yunit={Score},
    grid=on,
    xaxislabels=yes,][
    xlabels={Class A},
    data={15, 18, 19, 20, 20, 21, 21, 22, 23, 25, 28, 30}]
```

8.3 Pure density shape

In some cases, the user may wish to visualize only the shape of the distribution without the statistical overlay. This can be achieved by setting `showboxplot=no`. The example in ***Table 7.2*** and ***Chart 7.2*** shows the distribution of values in "Sample X" using only the density curve with a stroked outline.

#	**Sample X**
1	45
2	46
3	47
4	50
5	50
6	52
7	55
8	60
9	62
10	45
11	48
12	50

Table 8.2 Values measured in Sample X.

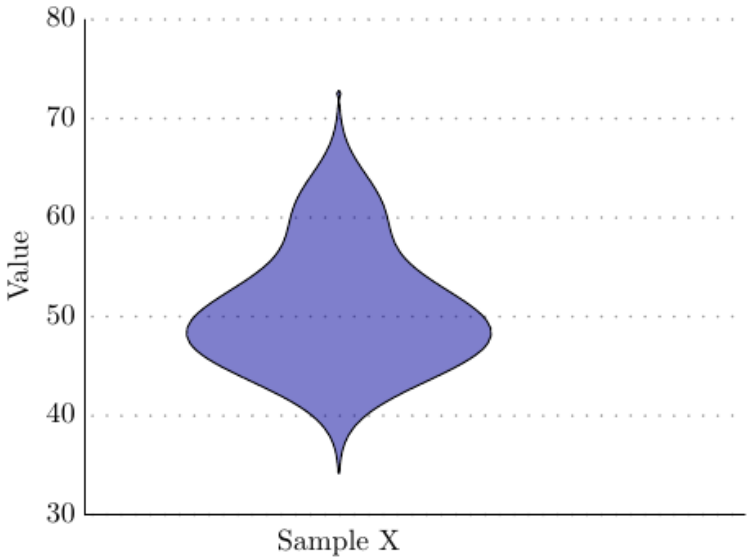

Chart 8.2 Violin chart showing only density (no boxplot).

The ***Chart 7.2*** was created using the commands below:

```
\violinchart[basic][
    xlength=8,ylength=6,
    fillcolor=darkblue,
    filltransparency=0.7,
    violinwidth=1.2,
    showboxplot=no,
    stroke=yes,
    axesunits=yes,
    yunit={Value},
    grid=on,
    xaxislabels=yes,][
    xlabels={Sample X},
    data={45, 46, 47, 50, 50, 52, 55, 60, 62, 45, 48, 50}]
```

8.4 Multiple data series

Violin charts are excellent for comparing distributions across multiple groups side-by-side. By plotting them next to each other, it is easy to compare the frequency of data at different levels.

8.4.1 Comparing shapes (without boxplot)

The following example compares the speed performance of three different models (A, B, and C). The data is listed in ***Table 7.3*** and visualized in ***Chart 7.3***. Note that `showboxplot=no` allows for a cleaner comparison of the distribution shapes.

Model A	Model B	Model C
10	18	20
12	18	21
11	19	22
13	11	23
12	11	24
14	12	25
15	15	26
10	15	27

Table 8.3 Speed measurements for three models.

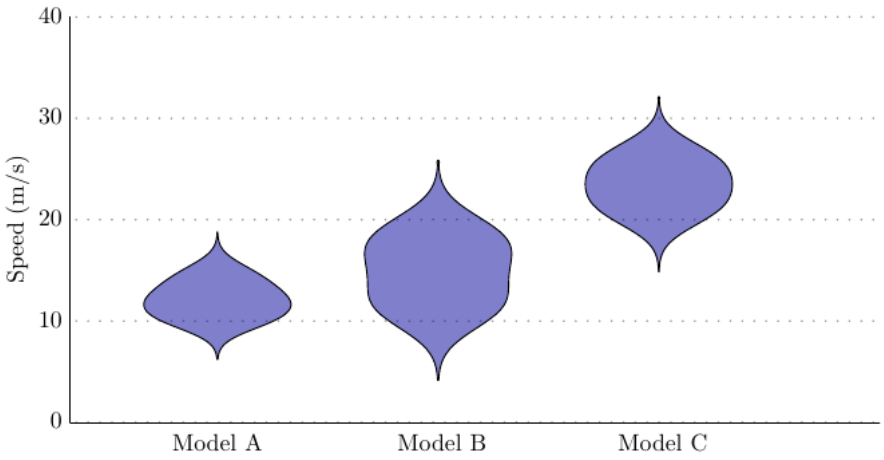

Chart 8.3 Multiple violin plots comparing distributions.

The ***Chart 7.3*** was created using the commands below:

```
\violinchart[basic][
    xlength=12,ylength=6,
    distance=1.5,
    violinwidth=1.0,
    fillcolor=darkblue,
    filltransparency=0.5,
    showboxplot=no,
    axesunits=yes,
    yunit={Speed (m/s)},
    grid=on,
    xaxislabels=yes,][
    xlabels={Model A, Model B, Model C},
    data={{10, 12, 11, 13, 12, 14, 15, 10},
          {18, 18, 19, 11, 11, 12, 15, 15},
          {20, 21, 22, 23, 24, 25, 26, 27}}]
```

8.4.2 Comparing distributions with statistics

For a more detailed comparison, you can include the boxplots within the violins. ***Table 7.4*** contains temperature data measured at three different times of day (Morning, Noon, Evening). ***Chart 7.4*** visualizes this data, allowing us to see both the spread (via the violin shape) and the median/quartiles (via the boxplot).

Morning	Noon	Evening
12	22	18
13	24	17
11	25	19
14	23	16
12	26	20
13	28	15
10	21	18
12	24	17

Table 8.4 Temperature measurements at different times of day.

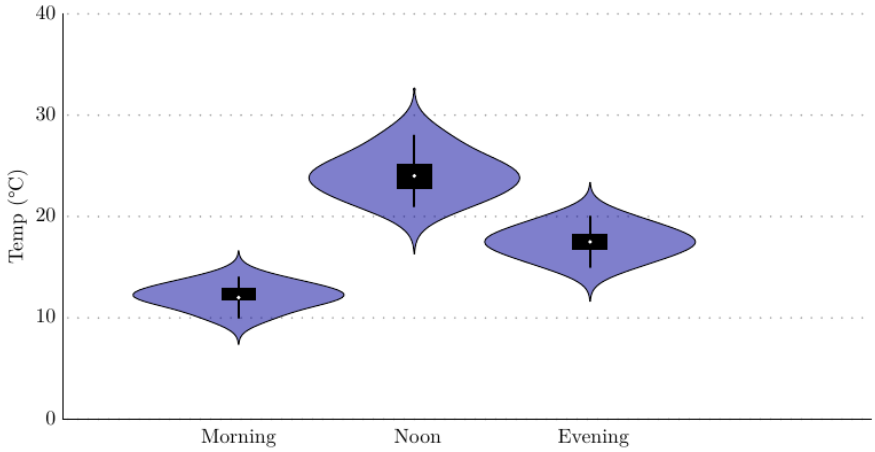

Chart 8.4 Full comparison with statistics (boxplot overlay).

The ***Chart 7.4*** was created using the commands below:

```
\violinchart[basic][
  xlength=14,ylength=7,
  violinwidth=1.2,
  fillcolor=darkblue,
  filltransparency=0.4,
  showboxplot=yes,
  boxwidth=0.2,
  medianwidth=2,
  axesunits=yes,
  yunit={Temp (°C)},
  xaxislabels=yes,][
  xlabels={Morning, Noon, Evening},
  data={{12, 13, 11, 14, 12, 13, 10, 12},
        {22, 24, 25, 23, 26, 28, 21, 24},
        {18, 17, 19, 16, 20, 15, 18, 17}}]
```

9 Heatmap charts

A heatmap is a data visualization technique that shows magnitude of a phenomenon as color in two dimensions. (Wilkinson, 2009) The variation in color may be by intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space. In this module, the heatmap represents values in a matrix where the x-axis and y-axis represent categories, and the color intensity of the cell represents the value.

9.1 Heatmap chart parameters

The following parameters (and their default values) can be set in the heatmap chart.

To work with cells and dimensions:

cellwidth: width of a single cell
cellheight: height of a single cell

To set the colors and visual style:

colorscheme: determines the color gradient used for the map.
steps: if defined (integer > 1), the color gradient is divided into discrete steps instead of a smooth transition. This is useful for categorizing data.
steps: if defined (integer > 1), the color gradient is divided into discrete steps instead of a smooth transition.
contour: yes/no
contourcolor: color of the cell borders.
showcellvalues: yes/no

To set the labels and legend:

xaxislabels: yes/no
yaxislabels: yes/no
xlabelsrotation: yes/no. Rotates the labels on the x-axis by 90 degrees.
legend: yes/no
legenddistance: distance between the chart and the legend.

9.2 Basic heatmap

The basic heatmap visualizes a 2D matrix of data. The module automatically calculates the minimum and maximum values in the dataset to normalize the color intensity. The example in ***Chart 8.1*** visualizes sales intensity in a cafe during the week.

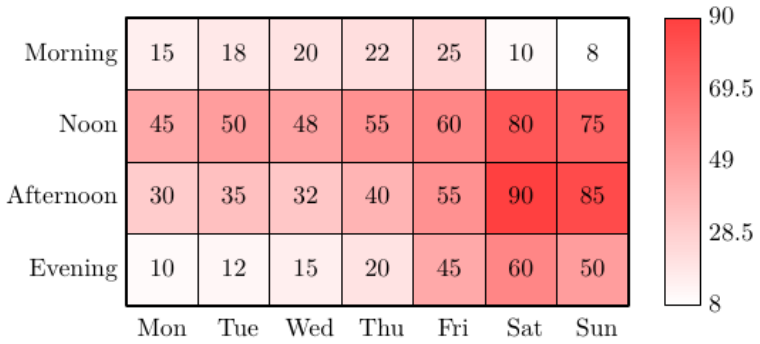

Chart 9.1 Heatmap showing sales intensity.

The ***Chart 8.1*** was created using the following commands:

```
\heatmapchart[basic][
    cellwidth=1,
    cellheight=1,
    colorscheme=red,
    showcellvalues=yes,
    xaxislabels=yes,
    yaxislabels=yes,
    legend=yes,][
    xlabels={Mon, Tue, Wed, Thu, Fri, Sat, Sun},
    ylabels={Morning, Noon, Afternoon, Evening},
    data={{15, 18, 20, 22, 25, 10, 8},
          {45, 50, 48, 55, 60, 80, 75},
          {30, 35, 32, 40, 55, 90, 85},
          {10, 12, 15, 20, 45, 60, 50}}]
```

9.3 Heatmap with discrete steps

By default, the heatmap uses a continuous color gradient. However, using the `steps` parameter allows you to categorize values into distinct intensity levels. The example in ***Chart 8.2*** shows server CPU load grouped into 5 intensity levels using the green color scheme.

Chart 9.2 Heatmap with discrete color steps.

The ***Chart 8.2*** was created using the following commands:

```
\heatmapchart[basic][
    colorscheme=green,
    showcellvalues=yes,
    xaxislabels=yes,
    yaxislabels=yes,
    xlabelsrotation=yes,
    steps=20,][
    xlabels={Server 1, Server 2, Server 3, Server 4},
    ylabels={DB, Web App, Storage, Backup},
    data={{85, 45, 20, 15},
          {30, 60, 25, 10},
          {10, 15, 95, 20},
          {5,  5,  10, 80}}]
```

9.4 Correlation matrix (Rotated labels)

Heatmaps are frequently used in statistics to visualize correlation matrices. In these cases, axis labels can be long. The `xlabelsrotation` parameter allows these labels to be displayed vertically. The example in ***Chart 8.3*** uses the blue color scheme.

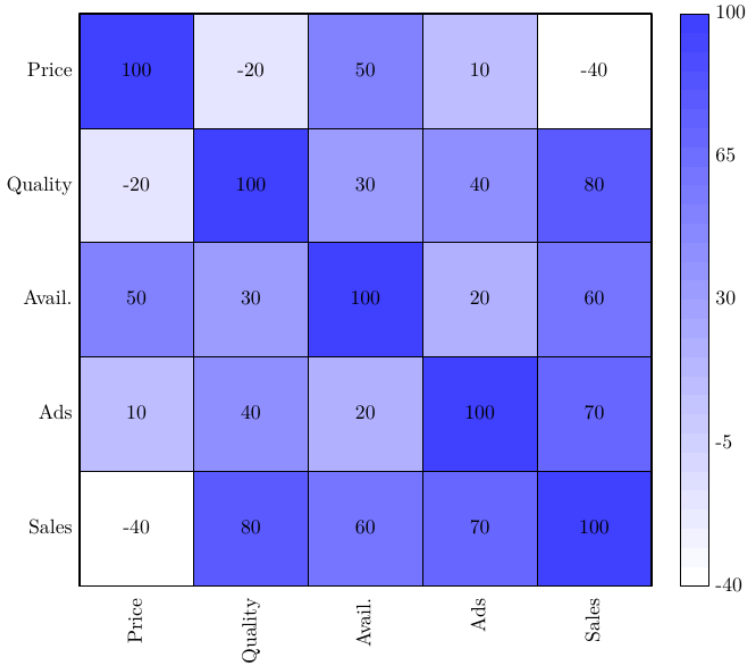

Chart 9.3 Correlation matrix with rotated labels.

The ***Chart 8.3*** was created using the following commands:

```
\heatmapchart[basic][
    colorscheme=blue,
    showcellvalues=yes,
    xaxislabels=yes,
    yaxislabels=yes,
    xlabelsrotation=yes,
    steps=30,][
    xlabels={Price, Quality, Avail., Ads, Sales},
    ylabels={Price, Quality, Avail., Ads, Sales},
    data={{100, -20, 50,  10, -40},
          {-20, 100, 30,  40,  80},
          { 50,  30, 100, 20,  60},
          { 10,  40, 20, 100,  70},
          {-40,  80, 60,  70, 100}}]
```

10 Polar charts

A polar chart is a graph used to plot data on a circular coordinate system. (IBM, 2012) Unlike Cartesian charts, which use X and Y coordinates, polar charts define the position of a point using an angle and a distance from the center. (Sukumar, 2025) This type of chart is particularly useful for displaying scientific data, such as microphone sensitivity patterns, cyclical time data, or directional measurements. The subtypes of the polar chart are:

- line,
- scatter.

10.1 Polar chart parameters

The following parameters (and their default values) can be set in the polar chart.

To work with dimensions and scale:

ylength: defines the visual radius of the chart in cm
maxradius: the data value that corresponds to the outer edge of the chart. If not set, it is calculated automatically from the maximum data value.
rotate: rotates the entire chart by a specific number of degrees

To work with the grid:

gridcolor: color of the grid lines
gridwidth: thickness of the grid lines
gridsteps: number of concentric circles to draw for the value axis
anglesteps: number of radial spokes to draw for the angle axis

To set the visual style:

linecolor: color of the data line
linewidth: thickness of the data line
closed: yes/no. Determines if the last point should be connected to the first point
fillcolor: color used to fill the polygon formed by the line
filltransparency: transparency of the fill

To set the dots:

dots: yes/no
dotscolor: color of the markers
dotswidth: size of the markers

10.2 Polar line chart

The polar line chart connects data points with a curve. It is commonly used to visualize shapes or patterns, such as the directional characteristics of antennas or microphones. The example in ***Chart 9.1*** shows a cardioid pattern. The data represents pairs of {*angle*, *value*}.

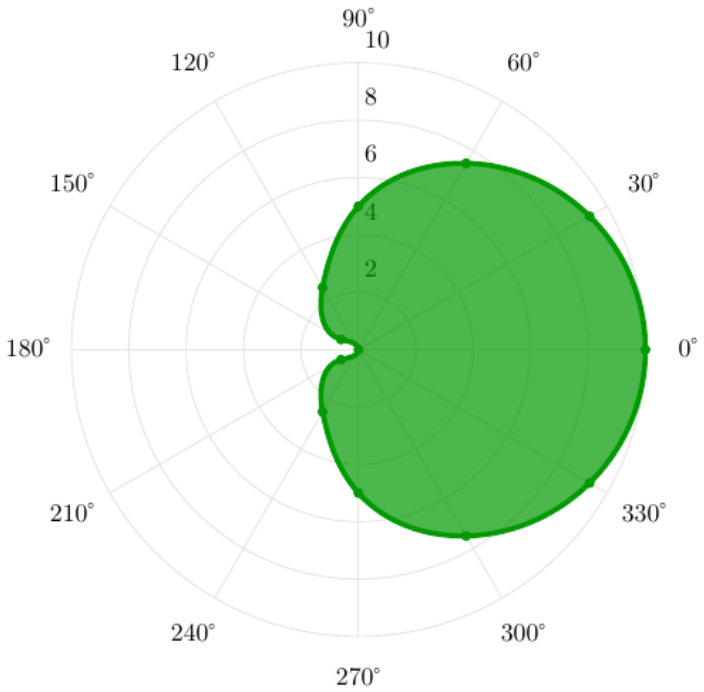

Chart 10.1 Polar line chart showing a cardioid pattern.

The ***Chart 9.1*** was created using the following commands:

```
\def\polardata{{0, 10}, {30, 9.3}, {60, 7.5}, {90, 5},
               {120, 2.5}, {150, 0.7}, {180, 0},
               {210, 0.7}, {240, 2.5}, {270, 5},
               {300, 7.5}, {330, 9.3}}

\polarchart[line][
    ylength=4,
    gridsteps=5,
    maxradius=10,
    linecolor=red,
    linewidth=2,
    fillcolor=red,
    filltransparency=0.7,
    dots=yes,
    dotscolor=red,
    closed=yes,][data={\polardata}]
```

10.2.1 Open curves (Spirals)

By default, the polar line chart closes the shape (connects the last point to the first). However, for data that represents a progression or a spiral, you can set `closed=no`. The example in ***Chart 9.2*** demonstrates an Archimedian spiral where the radius increases linearly with the angle.

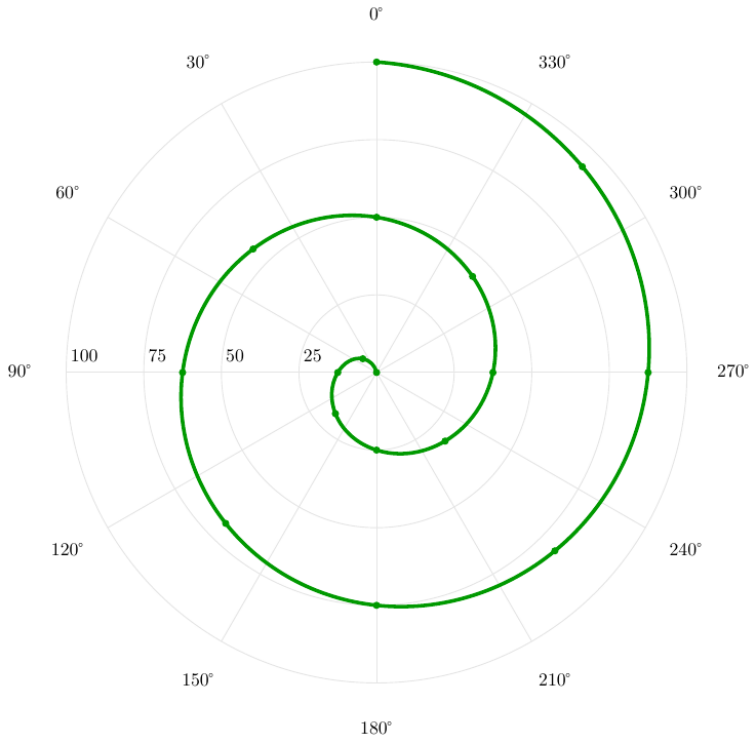

Chart 10.2 Polar chart with `closed=no` (Spiral).

The ***Chart 9.2*** was created using the following commands. Note the use of `rotate` to orient the chart:

```
\def\polardata{{0, 0}, {45, 6.25}, {90, 12.5}, {135, 18.75}, {180, 25},
               {225, 31.25}, {270, 37.5}, {315, 43.75}, {360, 50},
               {405, 56.25}, {450, 62.5}, {495, 68.75}, {540, 75},
               {585, 81.25}, {630, 87.5}, {675, 93.75}, {720, 100}}

\polarchart[line][
    ylength=6,
    gridsteps=4,
    maxradius=100,
    linecolor=darkgreen,
    linewidth=2,
    dots=yes,
    dotscolor=darkgreen,
    closed=no,
    filltransparency=1,
    rotate=90][data={\polardata}]
```

10.3 Polar scatter chart

The polar scatter chart displays data points without connecting lines. This is useful for visualizing events that have a directional and magnitude component but no inherent sequence. The example in ***Chart 9.3*** shows random measurements at various angles.

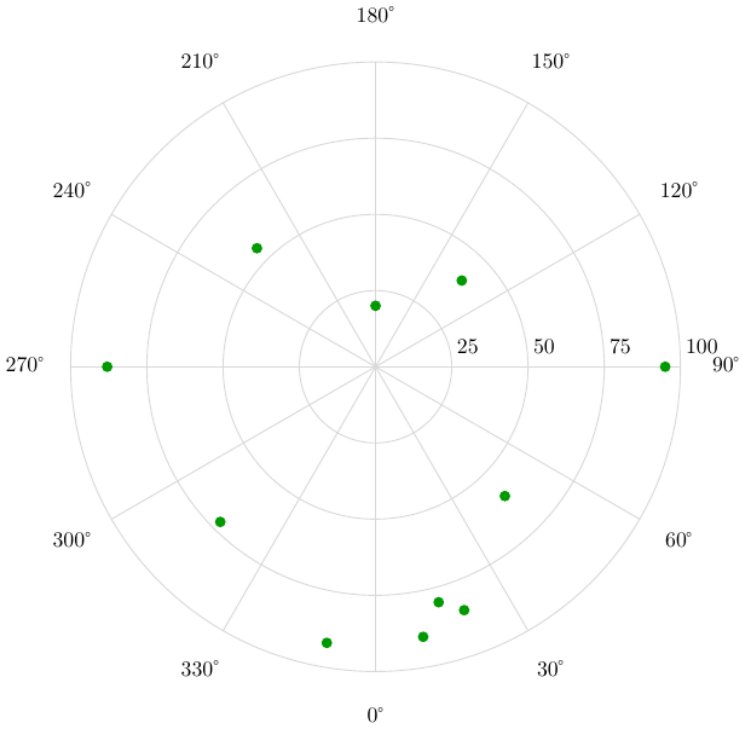

Chart 10.3 Polar scatter chart.

The ***Chart 9.3*** was created using the following commands:

```
\polarchart[scatter][
    ylength=5,
    gridcolor=lightgray,
    dotscolor=blue,
    dotswidth=5,
    maxradius=100,
    rotate=270,][
    data={
    {15, 80}, {45, 60}, {90, 95}, {135, 40},
    {180, 20}, {225, 55}, {270, 88}, {315, 72},
    {10, 90}, {20, 85}, {350, 92}}]
```

11 Sunburst charts

A sunburst chart is a graphical representation used to visualize hierarchical data structures. It consists of an inner circle surrounded by rings of deeper hierarchy levels. The angle of each segment is either proportional to a value or divided equally under its parent node.
Unlike a standard pie chart, which shows a single layer of data, a sunburst chart shows how a whole breaks down into parts, and those parts into smaller sub-parts. It is particularly effective for visualizing genealogy, file system directories, or organizational structures.

11.1 Sunburst chart parameters

The following parameters (and their default values) can be set in the sunburst chart.

To work with dimensions:

diameter: total diameter of the chart in cm
centerhole: diameter of the empty space in the center in cm. If set to 0, the root node creates a full circle in the center. If larger than 0, the chart looks like a multi-level doughnut chart.

To set the visual style:

linewidth: thickness of the white separators between segments
linecolor: color of the separators
fillpalette: palette of colors used to color the segments
filltransparency: transparency of the segment fill colors

To set the labels:

labels: yes/no
labelscale: scaling factor for the text font size
labelcolor: color of the text labels

11.2 Data Input Structure

Unlike other charts in this module that use matrix data, the Sunburst chart uses a ****parent-child relationship**** structure defined by three specific lists. This method allows for defining complex hierarchies without a strict grid.
The data must be provided using three keys in the parameters:

- **names**: A list of unique identifiers (names) for every node.
- **parents**: A list indicating the parent of the corresponding node in the **names** list. The root node (the center) must have an empty parent.
- **values**: A list of numerical values for the segments.

Important: The order of items in these three lists must correspond exactly. The n-th item in `parents` and `values` belongs to the n-th item in `names`.

Logic of Values:

If a `parent` has a value of **0**, the module automatically calculates its value as the sum of its `children`'s values. If a value is provided explicitly, it determines the size of that segment relative to its siblings.

11.3 Basic Sunburst (Genealogy)

The basic sunburst chart places the root element in the center and children in the surrounding rings. The example in ***Chart 10.1*** shows a simple family tree (Genealogy).

Understanding the Data:

In the code below:

- **Eve** is the root `parent` is empty.
- Cain and Seth are *children* of **Eve**.
- Enos and Noam are *children* of Seth.

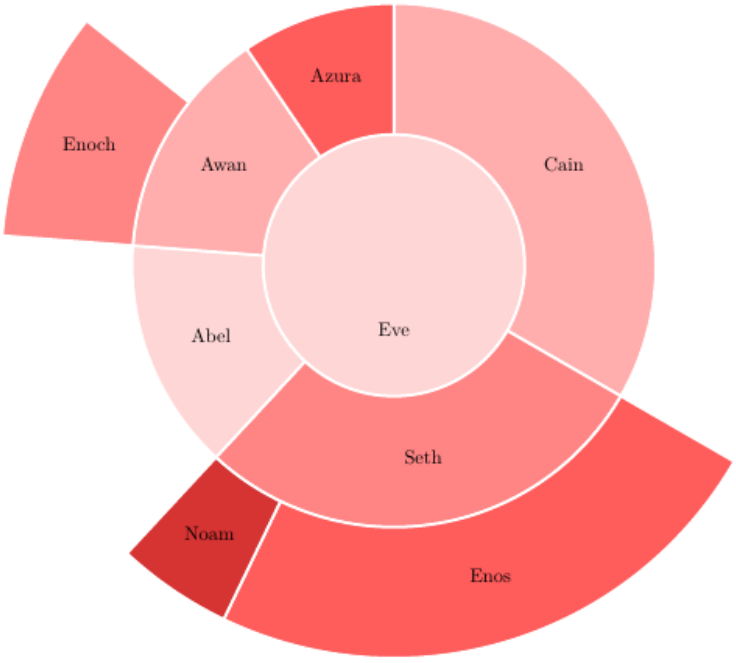

Chart 11.1 Sunburst chart showing genealogy (Root in center).

The ***Chart 10.1*** was created using the following commands:

```
\sunburstchart[basic][
    diameter=10,
    centerhole=0,
    fillpalette=redpalette,
    labels=yes][
    % 1. Identifiers
    names={Eve, Cain, Seth, Enos, Noam, Abel, Awan, Enoch, Azura},

    % 2. Hierarchy (Who is the parent?)
    % Note: The first item (Eve) has no parent (empty)
    parents={, Eve, Eve, Seth, Seth, Eve, Eve, Awan, Eve},

    % 3. Values
    values={10, 14, 12, 10, 2, 6, 6, 4, 4}]
```

11.4 Sunburst as a Donut Chart

If the hierarchy has only one level of depth (Root + direct children) and the `centerhole` is sufficiently large, the Sunburst chart functions exactly like a Donut chart.

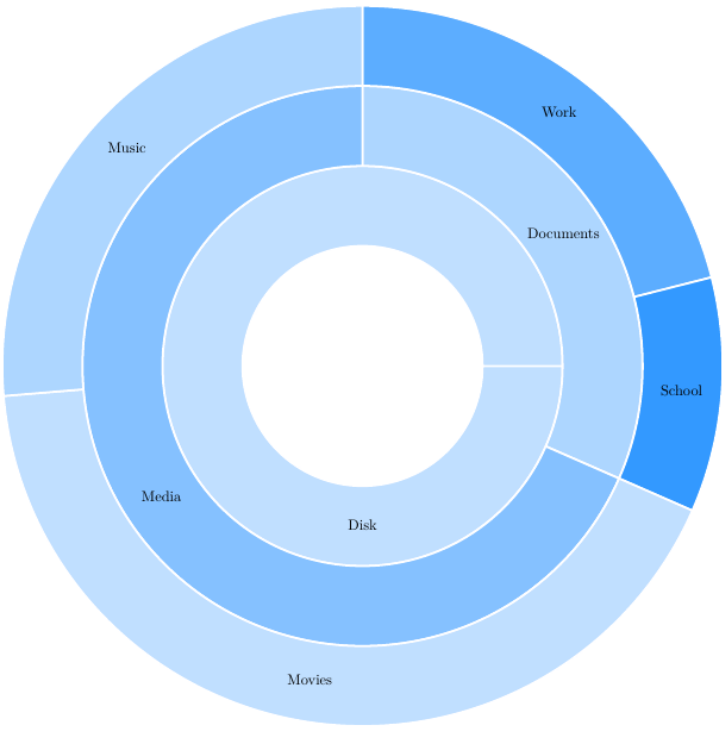

Chart 11.2 Sunburst chart behaving as a Donut chart (Disk Usage).

The ***Chart 10.3*** was created using the following commands:

```
\def\disknames{Disk, Documents, Media, Work, School, Movies, Music}
\def\diskparents{, Disk, Disk, Documents, Documents, Media, Media}
\def\diskvalues{0, 0, 0, 40, 20, 80, 50}

\sunburstchart[basic][
    diameter=12,
    centerhole=4,
    linewidth=1,
    linecolor=white,
    fillpalette=bluepalette,
    filltransparency=0.8,
    labels=yes,
    labelscale=0.7][
    names={\disknames},
    parents={\diskparents},
    values={\diskvalues}]
```

References

GeeksForGeeks Box Plot. In *Machine Learning Tutorial* [on-line]. 2010 [cit. 2025-10-29]. Available at: https://www.geeksforgeeks.org/machine-learning/box-plot/.

IBM Polar charts. In *GDDM-PGF V2R1.3 Application Programming Guide* [on-line]. 2012 [cit. 2025-12-1]. Available at: https://www.ibm.com/docs/en/gddm?topic=type-polar-charts/.

Kocurová, T., Kašparová, A. *t-statistical-charts.mkiv* [on-line]. Ver. 1. 2020a. Available at: https://www.thala.cz/statcharts/t-statistical-charts.mkiv.

Kocurová, T., Kašparová, A. *t-statistical-charts.lua* [on-line]. Ver. 1. 2020b. Available at: https://www.thala.cz/statcharts/t-statistical-charts.lua.

Kocurová, T., Kašparová, A., Hála, T. *Drawing statistical charts* [on-line]. Brno : [s. n.], 2020. [cit. 2025-10-3]. 116 s. Available at: https://www.thala.cz/statcharts/statistical-charts.pdf.

Kumari, J. Kernel Density Estimation (KDE) Plots. In *Data Analysis* [on-line]. 2025 [cit. 2025-11-28]. Available at: https://www.analyticsvidhya.com/blog/2025/05/kde-plot/.

Scott, D. W. Histogram. In *Wiley Interdisciplinary Reviews: Computational Statistics* [on-line]. 2010 [cit. 2025-10-27]. Available at: http://www.tug.org/texlive/doc/texlive-en/texliveen.html.

Smith, G. A. Understanding error bars in charts. In *Decoded* [on-line]. 2025 [cit. 2025-11-02]. Available at: https://www.pewresearch.org/decoded/2025/09/16/understanding-error-bars-in-charts/.

Sukumar, H. Polar charts 101. In *An in-depth guide* [on-line]. 2025 [cit. 2025-12-1]. Available at: https://inforiver.com/insights/polar-charts-101/.

Waskom, M. seaborn.kdeplot. In *seaborn* [on-line]. 2024 [cit. 2025-11-28]. Available at: https://seaborn.pydata.org/generated/seaborn.kdeplot.html.

Waskom, M. seaborn.violinplot. In *violinplot* [on-line]. 2024 [cit. 2025-11-21]. Available at: https://seaborn.pydata.org/generated/seaborn.violinplot.html.

Wilkinson, L. – Friendly, M. *The History of the Cluster Heat Map.* 63(2). vyd. [B.m.] : The American Statistician, 2009. 179–184 s. ISBN DOI: 10.1198/tas.2009.0033.

Yi, M. A Complete Guide to Stacked Bar Charts. In *Atlassian* [on-line]. 2025 [cit. 2025-10-27]. Available at: https://www.atlassian.com/data/charts/stacked-bar-chart-complete-guide.