ggVisIntreval: An R package for visualizing interval-valued data using **ggplot2**

Bo-Syue Jiang[1], Han-Ming Wu[2*]

[1] Department of Statistics, National Taipei University, New Taipei City, Taiwan
[2] Department of Statistics, National Chengchi University, Taipei City, Taiwan

[*] wuhm@g.nccu.edu.tw

Abstract

Exploratory data analysis (EDA) remains indispensable for revealing a dataset's salient features prior to formal modeling. Classic EDA relies on graphical summaries such as boxplots, histograms, and scatterplots, supplemented by dimension-reduction tools and interactive displays. Yet modern data are increasingly high-volume and structurally rich: single observations may be recorded not as scalars but as intervals, histograms, or full empirical distributions, collectively known as symbolic data. Conventional graphics cannot natively accommodate these objects, creating a gap between contemporary data structures and available visualization tools. To bridge this gap, we introduce **ggVisInterval**, an R package for visualizing interval-valued symbolic data. The package delivers seamless integration with the **ggplot2** workflow. We detail the package design, explain its extensibility, and illustrate its practical value through two real-world symbolic datasets. The results demonstrate that **ggVisInterval** enables analysts to uncover structure and anomalies in interval-valued data that would remain hidden with traditional scalar-oriented graphics, thereby enhancing the EDA toolkit for modern applied statistics and data science. The **ggVisInterval** package is currently available on the Comprehensive R Archive Network (CRAN).

Introduction

Exploratory data analysis (EDA), pioneered by [1], seeks to reveal a data set's main features before formal modeling. It relies on graphical summaries such as scatterplots, boxplots, histograms, and descriptive statistics such as the mean, median, and variance [2,3]. Projection-pursuit graphics [4] and modern dimension-reduction (DR) techniques [5] further assist in exploring multivariate structures. Compared with classical model-based inference, graphical EDA excels at condensing large, heterogeneous data sets and exposing patterns, trends, anomalies, and relationships. Its role is especially prominent in the early stages of data mining. With the rise of data science and big data, interest in EDA has resurged, and numerous R packages such as **DataExplorer** [6], **GGally** [7], **SmartEDA** [8], and **tableone** [9], among others, are now available on the Comprehensive R Archive Network (CRAN) [10].

In conventional analysis, each observation is a point in $\mathbb{R}^p$, where p denotes the number of variables. By contrast, *symbolic data* represent each statistical unit as richer objects such as intervals, histograms, or, more generally, Cartesian products of distributions in $\mathbb{R}^p$. The field of symbolic data analysis (SDA) was formalized by [11, 12] and later surveyed by [13] and [14]. Symbolic variables conveniently describe groups of individuals or abstract concepts and often arise by aggregating massive data sets into interval- or histogram-valued summaries, either to reduce size or to address new scientific questions.

Traditional EDA graphics target point-valued observations and therefore cannot be applied directly to symbolic data. Although some DR techniques allow symbolic objects to be explored in lower-dimensional spaces, there is a clear need for dedicated, visually oriented EDA tools tailored to symbolic data.

A number of R packages for symbolic data analysis are available on CRAN, as shown in Table 1. These packages implement statistical and machine-learning methods, primarily for interval-valued data, including dimension reduction, clustering, and regression. Some of them also provide limited graphical tools, such as scatterplots, histograms, and radar plots, built with the base R graphics system and **ggplot2** [15]. However, no R package has yet been developed specifically for exploratory symbolic data analysis and visualization. This gap motivated us to create **ggVisInterval**, a new package built on **ggplot2** that leverages its advanced graphical capabilities. Our current research focuses on EDA for interval data, where each variable is represented by an interval.

The article is structured as follows. The following section details the design of **ggVisInterval** and the manipulation of interval-valued symbolic data. We then review the descriptive statistics required for our graphical routines, followed by demonstrations of the implemented plots with accompanying R code. Extensions and customizations—including visualization of interval data in a Principal Component Analysis (PCA) subspace—are also discussed. The final section concludes the paper.

Package design and data manipulation

The ggVisInterval package design

The design of **ggVisInterval** consists of three parts. The first part focuses on data manipulation, such as reading from and writing to files, as well as aggregation. The second part computes descriptive statistics for interval-valued univariate and bivariate variables, including means, variances, and covariances. The third part implements basic and advanced graphical techniques for interval-valued univariate, bivariate, and multivariate variables, such as boxplots, histograms, scatterplots, radar plots, image plots, and so on. Figure 1 illustrates the design structure of **ggVisInterval**.

As the **ggVisInterval** package is based on **ggplot2**, we would like to provide a brief description of **ggplot2**. The concept behind **ggplot2** consists of three fundamental parts for a plot:

$$\text{Plot} = \text{Data} + \text{Aesthetics} + \text{Geometry}$$

Aesthetics are used to specify variables x and y of a dataframe `Data`. Additionally, they can customize the color, shape, or size of points, as well as the height of bars. Geometry defines the type of graphics, such as histograms, boxplots, line plots, density plots, dot plots, and so on. The general syntax for constructing a plot with **ggplot2** is as follows.

```
ggplot(data = <DATA> ) +
<GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),
stat = <STAT> , position = <POSITION> ) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

Both `ggplot(data = <DATA>)` and `<GEOM_FUNCTION>` are required. Below is an example of a scatterplot of `Sepal.Length` and `Sepal.Width` from the `iris` dataset.

Package	SO*	Description	Plots	Data Analysis Methods
GraphPCA	H	Graphical Tools of Histogram PCA	scatterplot	PCA
HistDat	H	Summary Statistics for Histogram/Count Data	-	basic statistics
HistDAWass	H	Histogram-Valued Data Analysis	boxplot, CDF, histogram, matrix of histograms, QF	basic statistics, Batch SOM, Fuzzy c-means, hierarchical clustering, K-means, PCA, regression, time series
HistogramTools	H	Utility Functions for R Histograms	ECDF plots	data manipulation, distance, quantile.
IntervalQuestionStat	I	Tools to Deal with Interval-Valued Responses in Questionnaires	scatterplot	arithmetic operations, Cronbach's α, distance, transformation
intkrige	I	A Numerical Implementation of Interval-Valued Kriging	interval plot, variograms	distance, kriging
iRegression	I	Regression Methods for Interval-Valued Variables	-	regression
MAINT.Data	I	Model and Analyse Interval Data	outliers plot, parallel coordinates plot, scatterplot	clustering, discriminant analysis, LRT, MANOVA, mixture model estimation, MLE
mdsOpt	I	Searching for Optimal MDS Procedure for Metric and Interval-Valued Data	scatterplot	MDS
psda	P	Polygonal Symbolic Data Analysis	scatterplot	basic statistics, regression
RSDA	I	R to Symbolic Data Analysis	histogram, radar, scatterplot	basic statistics, distance, K-means, KNN, MCFA, PCA, regression, RF, SVM.
symbolicDA	I	Analysis of Symbolic Data	3D interval plot, radar	decision tree, distance, dynamical clustering, HINoV, KDA, MDS, PCA, RF, SOM.

*Symbolic object: I for the interval-valued data; H for the histogram-valued data; P for the polygonal-valued data.

Table 1. The main features and the plots provided by the various R packages available on CRAN for symbolic data.

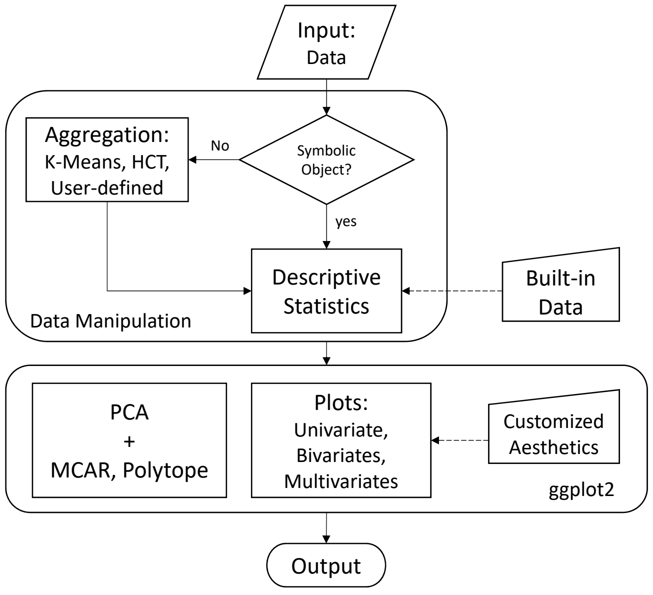


Fig 1. The design structure of the **ggVisInterval** package.

```
library(ggplot2)
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
geom_point() +
scale_color_manual(values = c("red", "blue", "yellow"))
```

Note that data used in **ggplot2** must belong to either the `data.frame` or `tibble` classes. Classes containing interval-valued data, such as `symbolic_interval` and `symbolic_tbl` defined in the **RSDA** package [16], are currently unsupported as they haven't been implemented in **ggplot2**. To address this limitation, we developed an R function called `ggInterval_<GRAPH_TYPE>`, which serves as the primary syntax of **ggVisInterval** and replaces the standard `ggplot` function. This design follows the common pattern used by several **ggplot2** extensions, including **ggtern** [17], **gganatogram** [18], **ggstatsplot** [19], among others. Below is the general syntax of **ggVisInterval** for visualizing interval-valued data.

```
ggInterval_<GRAPH_TYPE>(data = <SYMBOLIC_DATA>,
mapping = aes(<MAPPINGS>), ...) +
geom_<LAYER_TYPE>(...) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

The `<LAYER_TYPE>` includes `geom_text`, `geom_line`, `geom_point`, and `geom_area`. The following section provides some code examples. Unlike **ggplot2**, which requires data in a specific format, interval-valued data in SDA-related R packages can have various formats. To facilitate future maintenance and extension, we use the `symbolic_interval` and `symbolic_tbl` class objects from **RSDA** for `data = <SYMBOLIC_DATA>` in our package.

Data	Dimension	Reference	Application
`facedata`	27×6	[20] [21]	PCA
`oils`	8×4	[22] [23]	PCA
`mushroom`	23×3	[24] [25]	Regression
`Cardiological`	11×3	[26] [27]	Regression
`AbaloneIdt`	24×7	[28]	Dissimilarity
`Environment`	14×17	[29]	EDA

Table 2. Summaries of the built-in interval-valued data sets in the **ggVisInterval** package.

Built-in interval-valued data sets

Several widely used interval-valued datasets are included in this package as built-in datasets, as shown in Table 2. Two of them will be used for illustration in this study: `facedata` and `Environment`. The face recognition dataset [20, 21, 30] provides six face measurements of nine men (denoted by `Subjects`), all with three replicates, for a total of 27 observations. The six measurements designed to identify each face are expressed as the number of pixels in an image. Specifically, they are: the distance spanned by the eyes (denoted by $X_1 = \text{AD}$), the distance between the eyes ($X_2 = \text{BC}$), the distance from the outer right eye to the upper middle lip between the nose and mouth ($X_3 = \text{AH}$), the corresponding distance for the left eye ($X_4 = \text{DH}$), the distance from the upper middle lip to the outside of the mouth on the right side ($X_5 = \text{EH}$), and the distance to the left side of the mouth ($X_6 = \text{GH}$). These measurements were taken from a sequence of more than 1,000 images and cover a wide range of values, making them interval variables.

```
> library(ggVisInterval)
> data(facedata)
> facedata
# A tibble: 27 x 6
AD              BC                AH                DH
*       <symblc_n>      <symblc_n>        <symblc_n>        <symblc_n>
1 [155.00 : 157.00] [58.00 : 61.01] [100.45 : 103.28] [105.00 : 107.30]
2 [154.00 : 160.01] [57.00 : 64.00] [101.98 : 105.55] [104.35 : 107.30]
3 [154.01 : 161.00] [57.00 : 63.00]  [99.36 : 105.65] [101.04 : 109.04]
4 [168.86 : 172.84] [58.55 : 63.39] [102.83 : 106.53] [122.38 : 124.52]
5 [169.85 : 175.03] [60.21 : 64.38] [102.94 : 108.71] [120.24 : 124.52]
6 [168.76 : 175.15] [61.40 : 63.51] [104.35 : 107.45] [120.93 : 125.18]
7 [155.26 : 160.45] [53.15 : 60.21]   [95.88 : 98.49]   [91.68 : 94.37]
8 [156.26 : 161.31] [51.09 : 60.07]   [95.77 : 99.36]   [91.21 : 96.83]
9 [154.47 : 160.31] [55.08 : 59.03]   [93.54 : 98.98]   [90.43 : 96.43]
10 [164.00 : 168.00] [55.01 : 60.03] [120.28 : 123.04] [117.52 : 121.02]
# ... with 17 more rows, and 2 more variables: EH <symblc_n>, GH <symblc_n>
# i Use ‘print(n = ...)‘ to see more rows, and ‘colnames()‘ to see all variable names
> class(facedata)
[1] "symbolic_tbl" "tbl_df"       "tbl"          "data.frame"
> Subjects <- substr(rownames(facedata), 1, 3)
> Subjects
[1] "FRA" "FRA" "FRA" "HUS" "HUS" "HUS" "INC" "INC" "INC" "ISA" "ISA" "ISA"
[13] "JPL" "JPL" "JPL" "KHA" "KHA" "KHA" "LOT" "LOT" "LOT" "PHI" "PHI" "PHI"
[25] "ROM" "ROM" "ROM"
```

When applied to an interval variable, the `summary` function returns summary statistics for both the lower and upper bounds of the intervals, similar to its operation on numerical variables.

```
> summary(facedata)
$symbolic_interval
AD               BC                 AH
Min.    [149.34 : 155.32] [50.36 : 55.23]   [93.54 : 98.49]
1st Qu. [154.56 : 158.91] [53.60 : 59.08] [102.88 : 106.99]
Median  [163.00 : 167.07] [57.00 : 63.00] [115.26 : 119.60]
Mean    [162.90 : 162.90] [60.00 : 60.00] [113.17 : 113.17]
3rd Qu. [167.13 : 171.19] [61.22 : 65.04] [117.91 : 121.60]
Max.    [169.85 : 175.15] [66.03 : 69.01] [123.75 : 127.29]
Std.        [6.82 : 6.82]   [4.42 : 4.42]     [9.08 : 9.08]
DH               EH               GH
Min.      [90.43 : 94.37] [49.41 : 54.64] [48.27 : 50.61]
1st Qu. [105.18 : 111.07] [54.65 : 58.49] [51.60 : 56.03]
Median  [114.28 : 117.41] [56.73 : 61.72] [55.32 : 60.46]
Mean    [113.20 : 113.20] [59.85 : 59.85] [57.69 : 57.69]
3rd Qu. [117.10 : 121.72] [60.96 : 65.80] [58.52 : 63.84]
Max.    [124.08 : 127.78] [63.89 : 69.07] [64.20 : 67.80]
Std.        [9.48 : 9.48]   [4.04 : 4.04]   [4.63 : 4.63]
```

The Environment questionnaire data [31] comprises 14 objects with 17 variables, four of which are modal multi-valued while the remaining are interval-valued. The modal multi-valued variables will be used to illustrate the radar plot.

```
> data(Environment)
> dim(Environment)
[1] 14 17
> class(Environment)
[1] "symbolic_tbl" "tbl_df"       "tbl"          "data.frame"
> summary(Environment)
$symbolic_interval
CONTROL                SATISFY           INDIVIDUAL
Min.    [-723.25 : -339.65] [-570.57 : -259.45] [-826.25 : -484.48]
1st Qu.   [-270.31 : 83.66]  [-355.04 : -60.17]  [-508.97 : -88.07]
Median   [-189.18 : 188.39]  [-176.68 : 189.71]  [-202.13 : 132.72]
Mean        [-9.75 : -9.75]      [10.97 : 10.97]   [-31.16 : -31.16]
3rd Qu.   [-28.26 : 307.38]    [20.92 : 328.60]    [120.46 : 394.34]
Max.      [133.40 : 478.04]   [342.58 : 705.03]   [484.35 : 817.59]
Std.      [209.53 : 209.53]   [287.41 : 287.41]   [377.18 : 377.18]
...
$symbolic_modal
URBANICITY INCOMELEVEL EDUCATION REGIONDEVELOPME
[1,] 6: 0.15    25: 0.32    1: 0.37   4: 0.54
[2,] 4: 0.44    50: 0.20    3: 0.34   3: 0.23
[3,] 5: 0.10    75: 0.25    5: 0.16   2: 0.15
[4,] 1: 0.19    90: 0.13    6: 0.13   1: 0.07
[5,] 3: 0.07    100: 0.10
[6,] 2: 0.04
```

Aggregate conventional data table to interval-valued data table

The aggregation of numerical datasets serves as a potential source of interval-valued data. Using this approach, data are categorized into groups based on specific criteria, with intervals constructed by summarizing the minimum and maximum values within each group. The primary advantage of interval-valued symbolic data is their ability to reduce large datasets to a manageable size while retaining maximal information from the original dataset. We implemented an `R` function, `classic2sym`, that aggregates `data.frame` objects into `symbolic_tbl` objects. Below, we demonstrate these aggregation approaches using the Breast Cancer Wisconsin (Diagnostic) dataset obtained from Kaggle at `https://www.kaggle.com/uciml/breast-cancer-wisconsin-data?select=data.csv`. The dataset contains information for 569 patients with 32 variables (ID, diagnosis, and 30 real-valued features) describing characteristics of cell nuclei present in digitized images of fine needle aspirates (FNA) of breast masses.

```
> breastData <- read.csv("data.csv")
> colnames(breastData)
[1] "id"                      "diagnosis"
...
[31] "symmetry_worst"          "fractal_dimension_worst"
[33] "X"
```

Aggregation by clustering algorithms

Clustering algorithms represent a natural approach for data aggregation. Among these, `classic2sym` implements both K-means and hierarchical clustering via the argument `groupby = c("kmeans", "hclust", "customize")`, returning a list of components including clustering results and interval-valued data. Each cluster generates an interval-valued object. The interval-valued data can be accessed using `$intervalData`. Note that the clustering algorithm processes only numerical variables. Consequently, categorical variables are summarized by their relative frequencies within each cluster.

```
> breastData <- dplyr::select(breastData, -id)
> breastData.sym <- classic2sym(breastData, groupby = "kmeans", k = 5)
> breastData.sym.i <- breastData.sym$intervalData
> as.data.frame(head(breastData.sym.i[, 1:4], 5))
diagnosis     radius_mean    texture_mean    perimeter_mean
1 B:0.04 M:0.96 [13.81 : 19.59] [11.89 : 39.28]  [91.56 : 132.40]
2 B:0.68 M:0.32 [11.84 : 16.30] [10.89 : 30.72]  [77.93 : 109.80]
3 B:0.00 M:1.00 [20.73 : 28.11] [17.25 : 31.12] [135.70 : 188.50]
4 B:0.98 M:0.02  [6.98 : 13.05]  [9.71 : 33.81]   [43.79 : 85.09]
5 B:0.00 M:1.00 [15.50 : 24.25] [10.38 : 32.47] [102.90 : 166.20]
```

Aggregation by a categorical variable

A categorical variable in the data may also be used to conduct the aggregation. The number of observations (objects) for the converted interval-valued data is the number of categories for that variable. One can also possible aggregate observations based on user-defined categories.

```
> breastData.sym <- classic2sym(breastData, groupby = "diagnosis")
> breastData.sym.i <- breastData.sym$intervalData
> head(breastData.sym.i[, 1:4])
radius_mean    texture_mean   perimeter_mean           area_mean
```

(a) Univariate statistics

According to [32], [12] and [33]	
Mean	$\bar{X} = \frac{1}{n}\sum_{i=1}^{n}\frac{a_i + b_i}{2}$
Variance	$S^2 = \frac{1}{3n}\sum_{i=1}^{n}(a_i^2 + a_i b_i + b_i^2) - \frac{1}{4n^2}\left[\sum_{i=1}^{n}(a_i + b_i)\right]^2$

(b) Bivariate statistics

According to [32]	
Covariance	$C_{BG}(X_1, X_2) = \frac{1}{4n}\sum_{i=1}^{n}(b_{i1} + a_{i1})(b_{i2} + a_{i2}) - \frac{1}{4n^2}\left[\sum_{i=1}^{n}(b_{i1} + a_{i1})\right]\left[\sum_{i=1}^{n}(b_{i2} + a_{i2})\right]$
Correlation	$R_{BG}(X_1, X_2) = \frac{C_{BG}(X_1, X_2)}{S(X_1)S(X_2)}$
According to [12]	
Covariance	$C_{BD}(X_1, X_2) = \frac{1}{3n}\sum_{i=1}^{n} G_1 G_2 \left[Q_1 Q_2\right]^{1/2}$, where for $j = 1, 2$, $Q_j = (a_{ij} - \bar{X}_j)^2 + (a_{ij} - \bar{X}_j)(b_{ij} - \bar{X}_j) + (b_{ij} - \bar{X}_j)^2$, $G_j = \begin{cases} -1, & \text{if } \xi_{ij}^c \leq \bar{X}_j, \\ 1, & \text{if } \xi_{ij}^c > \bar{X}_j, \end{cases}$ where $\xi_{ij}^c = (a_{ij} + b_{ij})/2$.
Correlation	$R_{BD}(X_1, X_2) = \frac{C_{BD}(X_1, X_2)}{S(X_1)S(X_2)}$
According to [33]	
Covariance	$C_B(X_1, X_2) = \frac{1}{6n}\sum_{i=1}^{n}[2(a_{i1} - \bar{X}_1)(a_{i2} - \bar{X}_2) + (a_{i1} - \bar{X}_1)(b_{i2} - \bar{X}_2) + (b_{i1} - \bar{X}_1)(a_{i2} - \bar{X}_2) + 2(b_{i1} - \bar{X}_1)(b_{i2} - \bar{X}_2)]$.
Correlation	$R_B(X_1, X_2) = \frac{C_B(X_1, X_2)}{S(X_1)S(X_2)}$

Table 3. Univariate and bivariate statistics for interval-valued data.

```
1  [6.98 : 17.85]   [9.71 : 33.81] [43.79 : 114.60]   [143.50 : 992.10]
2 [10.95 : 28.11] [10.38 : 39.28] [71.90 : 188.50] [361.60 : 2,501.00]
```

Alternatively, users may generate interval-valued data tables using the `classic.to.sym` function from the **RSDA** package.

Reading and writing interval-valued data to and from R

To handle interval-valued data input and output, **ggVisInterval** employs the R functions `read.sym.table` and `write.sym.table` from **RSDA** package. These functions support reading and writing symbolic data in CSV format using a standardized structure. This design ensures compatibility with other SDA packages and facilitates quick adoption of **ggVisInterval**. For implementation details, consult the documentation for `read.sym.table` and `write.sym.table` in **RSDA**.

Descriptive statistics for interval-valued data

EDA primarily consists of numerical data summaries (e.g., measures of central tendency and variation) along with graphical representations. For instance, the five-number summary of numerical data (minimum, 25th percentile, median, 75th percentile, and maximum) is used to construct boxplots. Consequently, basic descriptive statistics for symbolic data should be computed before applying graphing techniques. Furthermore, to standardize data or perform multivariate analyses like PCA on interval-valued data, we must first calculate means, variances, and covariances. The SDA literature includes several algorithms for computing descriptive statistics for interval-valued data.

Denote the ith observation of a univariate interval-valued variable X by $[a_i, b_i]$, $i = 1, \cdots, n$, and the ith observation of bivariate interval-valued variables (X_1, X_2) by $([a_{i1}, b_{i1}], [a_{i2}, b_{i2}])$, $i = 1, \cdots, n$. We have implemented basic descriptive statistics for univariate and bivariate interval-valued variables following [34]. The corresponding formulas appear in Table 3. The following code demonstrates how to obtain these descriptive statistics.

```
> mean(facedata)
# A tibble: 1 x 6
AD    BC    AH    DH    EH    GH
<dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  163.  60.0  113.  113.  59.8  57.7
> sd(facedata)
# A tibble: 1 x 6
AD    BC    AH    DH    EH    GH
<dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  6.82  4.42  9.08  9.48  4.04  4.63
>
> cov(facedata$AD, facedata$BC, method = "BD")
[1] 21.47015
> cor(facedata$AD, facedata$BC, method = "BD")
[1] 0.7132073
```

Standardization of interval-valued data

Standardizing each variable in the dataset to have zero mean and unit variance is often crucial for data analysis. This process prevents variables with larger measurement scales from dominating those with smaller scales. In this package, the standardization approach for interval-valued variables follows a general principle: applying identical transformations separately to both lower and upper bounds of all intervals. This method ensures all point values between the bounds receive the same linear standardization.

```
> scale(facedata)$intervalData
AD                BC               AH               DH
FRA1 [-1.10 : -0.82]  [-0.39 : 0.20] [-1.38 : -1.07] [-0.85 : -0.61]
FRA2 [-1.24 : -0.40]  [-0.59 : 0.79] [-1.21 : -0.83] [-0.92 : -0.61]
FRA3 [-1.23 : -0.26]  [-0.59 : 0.59] [-1.50 : -0.82] [-1.26 : -0.43]
HUS1   [0.83 : 1.38]  [-0.28 : 0.67] [-1.12 : -0.72]   [0.95 : 1.17]
HUS2   [0.96 : 1.68]   [0.04 : 0.86] [-1.11 : -0.48]   [0.73 : 1.17]
HUS3   [0.81 : 1.70]   [0.28 : 0.69] [-0.96 : -0.62]   [0.80 : 1.24]
INC1 [-1.06 : -0.34]  [-1.34 : 0.04] [-1.88 : -1.59] [-2.23 : -1.95]
...
```

No. variates	Plots	Syntax
Univariate	index plot	`ggInterval_index`
	boxplot	`ggInterval_boxplot`
	histogram	`ggInterval_hist`
	line plot	`ggInterval_index`
	center-range plot	`ggInterval_centerRange`
	min-max plot	`ggInterval_minmax`
Bivariate	2D scatterplot	`ggInterval_scatter`
	2D histogram	`ggInterval_2Dhist`
Multivariate	scatterplot matrices	`ggInterval_scaMatrix`
	2D histogram matrices	`ggInterval_2DhistMatrix`
	image plot	`ggInterval_indexImage`
	radar plot	`ggInterval_radar`
	quantile plot	`ggInterval_radar`

Table 4. The graphing techniques provided by **ggVisInterval**.

Graphics for interval-valued symbolic data

ggVisInterval extends and implements traditional graphical techniques for interval-valued data, which can be classified into univariate, bivariate, and multivariate plots as shown in Table 4. Note that numerous variants of these traditional graphical techniques exist in the literature. In this study, we focus on introducing fundamental statistical graphics that accurately and effectively represent symbolic data for exploratory analysis. Future versions may incorporate more advanced graphical methods. Additionally, we have developed two specialized plot types for interval-valued data: the min-max plot and the center-range plot, designed to highlight their unique characteristics.

Univariate plots

Univariate plots characterize the location, dispersion, and distribution of observations for a single variable. For interval-valued observations, we have implemented index plots, boxplots, histograms, line plots, min–max plots, and center–range plots.

The index plot An index plot (also called a run-sequence plot, (x_i vs i)) displays the values of a variable against their corresponding observation numbers (row indices) in the data set. It enables users to scan large amounts of data quickly to detect anomalies and unusual observations. For an interval-valued variable, the index plot consists of segments or bars whose endpoints represent the lower and upper bounds of each interval, plotted against the observation numbers. This visualization facilitates inspection of the distributions of both interval ranges and centers. The following example, using `ggInterval_index`, illustrates four kinds of index plots for the variable AD in the face data set. Figure 2(a) shows an index plot with a segment representation, where the x-axis indicates the range of AD measurements and the y-axis the observation index. Figure 2(b) presents the same plot with coloured bars. Figure 2(c) depicts a variant that replaces bars with an image strip whose sequential colours encode interval ranges; we refer to this as an *index image*. Figure 2(d) displays index images using equi-width colour strips. As the plots reveal, the three faces recorded for each subject cluster tightly within that subject. In particular, subjects LOT, KHA, INC, and FRA exhibit lower-than-average AD values.

```
> # Figure 2(a)
> f.nr <- nrow(facedata)
```

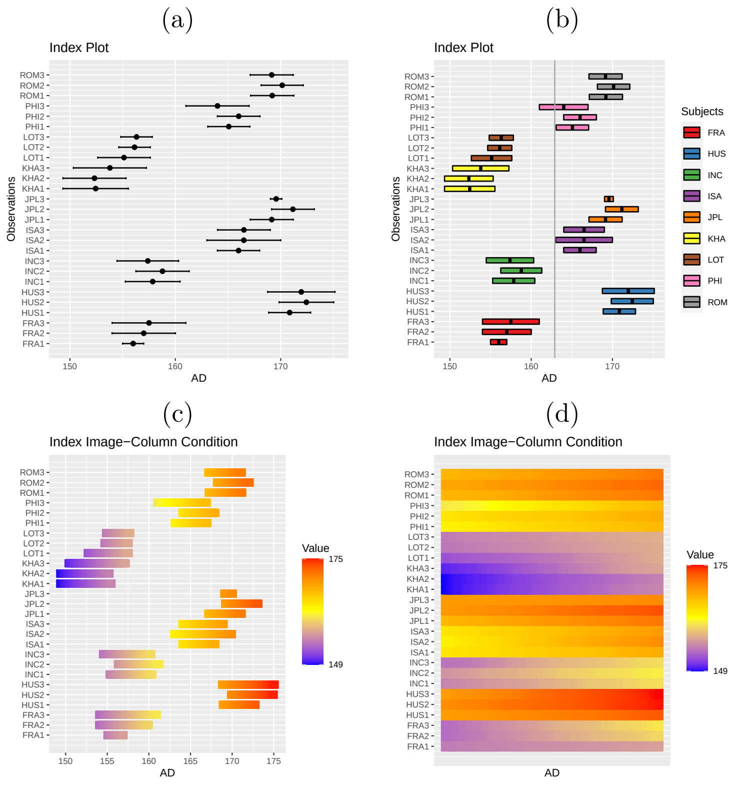


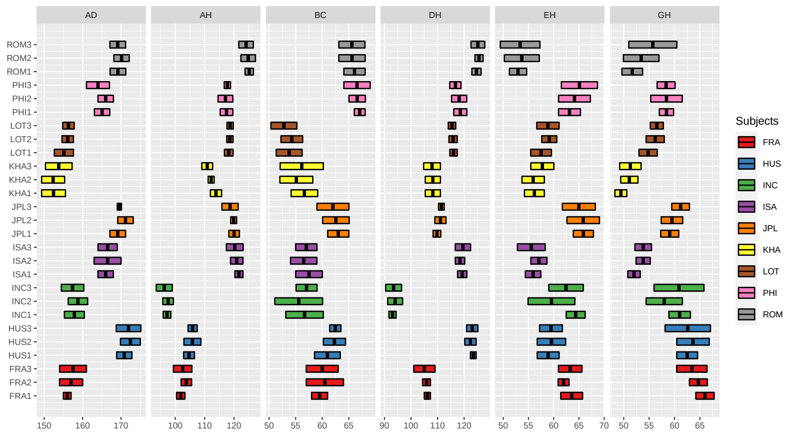
Fig 2. Four types of index plot for the variable AD of the face recognition data set. (a) Index plot. (b) Index plot with color bars. (c) Index image. (d) Index image with equi-width strips.

```
> f.nc <- ncol(facedata)
> ggInterval_index(facedata, aes(x = AD)) +
+   scale_y_continuous(breaks = 1:f.nr, labels = rownames(facedata))
>
> # Figure 2(b)
> ggInterval_index(facedata, aes(x = AD, fill = Subjects)) +
+   scale_fill_brewer(palette = "Set1") +
+   geom_vline(xintercept = mean(facedata$AD), color="darkgray") +
+   scale_y_continuous(breaks = 1:f.nr, labels = rownames(facedata)) +
+   labs(fill = "Subjects")
```

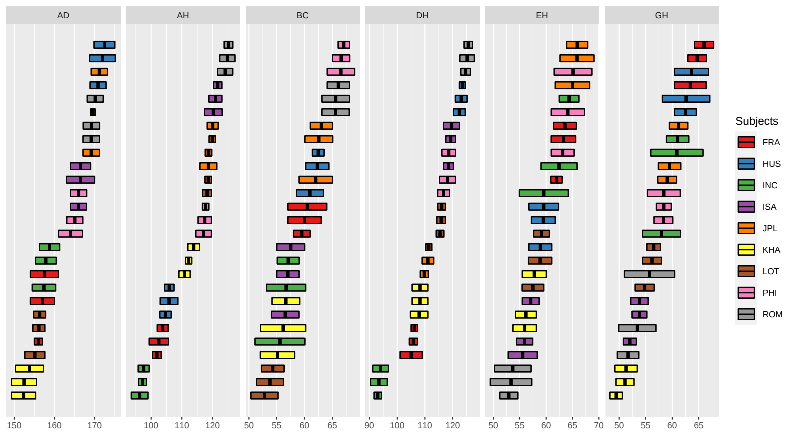
Figure 3(a) displays the index plots for all six variables in the face-recognition data set. Given the expected symmetry of facial characteristics, we observe that two individuals—HUS and KHA—exhibit markedly different AH and DH measurements. As Figures 3(b) and (c) illustrate, reordering the observations by specific statistics, such as the centers or ranges of the intervals, can reveal additional structure. The following code chunk generates these graphs. When the argument `plotAll = TRUE` is supplied, `ggInterval_index` produces an index plot for every variable in the data set. The same interface applies to the other graphing functions: `ggInterval_boxplot`, `ggInterval_centerRange`, `ggInterval_hist`, `ggInterval_indexImage`, and `ggInterval_minmax`.

```
> # Figure 3(a)
> ggInterval_index(facedata, aes(fill = Subjects), plotAll = T) +
+   scale_fill_brewer(palette = "Set1") +
+   labs(x = "", y = "", fill = "Subjects")
>
>
> # Figure 3(b)
> library(dplyr)
> dd <- 0
> for(i in 1:dim(facedata)[2]){
+   d <- data.frame(min = facedata[[i]]$min,
+                   max = facedata[[i]]$max,
+                   myname = Subjects,
+                   mysort = (facedata[[i]]$min+facedata[[i]]$max)/2)
+   temp <- dplyr::arrange(d, mysort)
+   dd <- cbind(dd, temp)
+ }
>
> dd <- dd[,-1]
> newd <- classic2sym(dd, groupby = "customize",
+                     minData = cbind(dd$min, dd$min.1, dd$min.2,
+                                     dd$min.3, dd$min.4, dd$min.5),
+                     maxData = cbind(dd$max, dd$max.1, dd$max.2,
+                                     dd$max.3, dd$max.4, dd$max.5))
> face.sort <- newd$intervalData
> colnames(face.sort) <- colnames(facedata)
> rownames(face.sort) <- rownames(facedata)
> myname <- dd[,c(3,7,11,15,19,23)]
> ggInterval_index(face.sort, aes(fill = unlist(myname)), plotAll = T) +
+   scale_fill_brewer(palette="Set1") +
+   labs(x = "", y = "", fill = "Subjects") +
+   scale_y_continuous(breaks = NULL, labels = NULL)
```

(a)



(b)



(c)

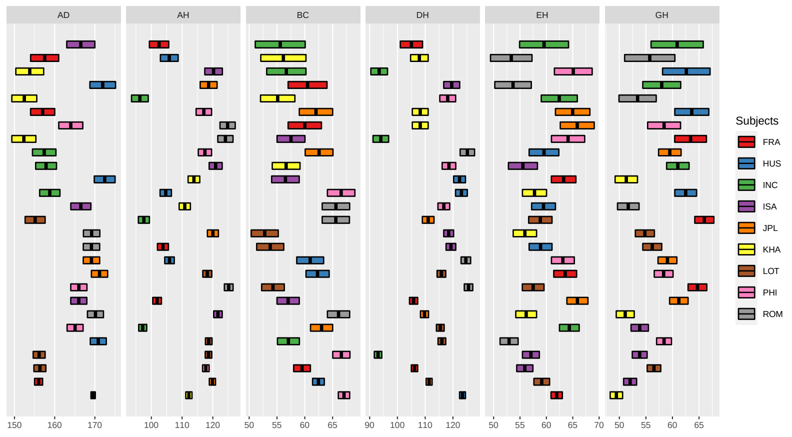


Fig 3. Index plots for face recognition data with different ordering of observations.

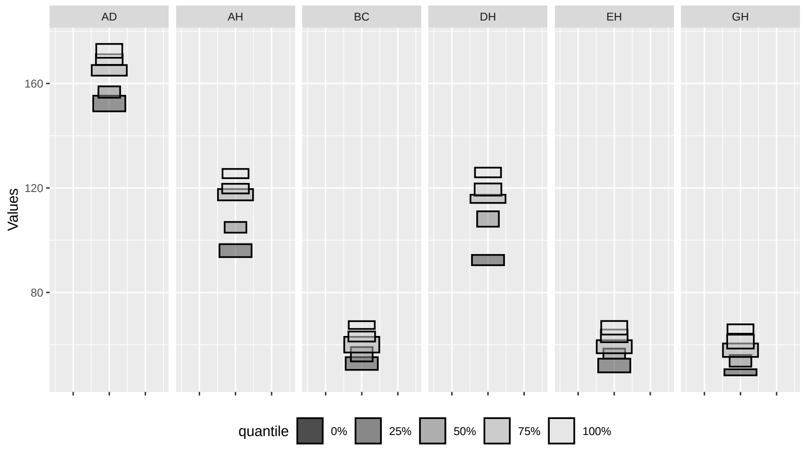


Fig 4. The side-by-side boxplot for all six variables of the face recognition data.

The boxplot Figure 4 displays side-by-side box plots for all six variables in the face data set. For each quantile box, the top (bottom) hinge represents the corresponding quantile of the upper (lower) bounds of the intervals. In addition to providing a quick visual summary of the data, box plots are particularly useful for comparing the distributions of multiple variables. The `ggInterval_boxplot` call below generates the interval box plots shown in Figure 4. Among the six measurements, AD exhibits the widest intervals, whereas BC shows the narrowest. Because facial features are generally symmetrical, we expect AH and DH, as well as EH and GH, to display similar distributions. Moreover, the overlapping quantile boxes of BC, EH, and GH suggest approximately bell-shaped distributions.

```
> # Figure 4
> ggInterval_boxplot(facedata, plotAll = T) +
+   theme(legend.position = "bottom", axis.text.x = element_blank())
```

The histogram A histogram is a graphical representation in which data are grouped into contiguous numeric ranges, each corresponding to a vertical bar. Counts (or frequencies) within each range are indicated by the height of the bar. Histograms are well suited for visualizing the distribution of numerical data. A univariate histogram of the interval data is constructed as follows.

1. Let $I = [\min_i a_i, \max_i b_i], i = 1, 2, \cdots, n$ be the interval that spans all the observed values of X.
2. Partition I into r subintervals $I_g = [\xi_{g-1}, \xi_g)$ of equal width $\|I_g\|, g = 1, 2, \cdots, r$.
3. The observed frequency for the histogram subinterval $I_g = [\xi_{g-1}, \xi_g)$ is defined as:

$$f_g = \sum_i \frac{\| \; [a_i, b_i] \cap I_g \; \|}{\| \; [a_i, b_i] \; \|}, \; g = 1, \cdots, r. \tag{1}$$

4. The histogram of an interval-valued variable X is then obtained by $\{(I_g, f_g), g = 1, \cdots, r\}$.

For the steps above, the subintervals I_g need not have equal widths. All interval boundaries can be used when computing the breaks ξ_g. More precisely, we pool a_i and

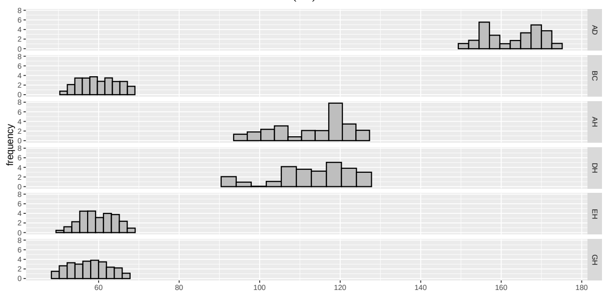
b_i into a single vector and sort it into $(x^{(1)}, x^{(2)}, \cdots, x^{(2n)})$, where $x^{(j)} \leq x^{(j+1)}$ for $j = 1, 2, \cdots, 2n-1$. The frequency for the subinterval $I'_g = [x^{(j)}, x^{(j+1)}]$ is then obtained from Equation (1). This construction yields a *non-equidistant-bin* histogram. The function `ggInterval_hist` implements histograms for interval variables. By specifying `method = "unequal-bin"`, one obtains the non-equidistant-bin version. Figures 5(a) and (b) display the histograms of all six variables in the face data, using equidistant and non-equidistant bins, respectively. Both plots convey the location and scale of each variable's distribution, but the non-equidistant-bin histogram reveals finer distributional details. The information conveyed is consistent with Figure 4: variables AD, AH, and DH exhibit bimodal distributions.

```
> # Figure 5(a)
> hist.obj.equal <- ggInterval_hist(facedata, plotAll = T, bins = 10)
> hist.obj.equal$plot
> hist.obj.equal$‘Table AD‘
Interval Observed.Frequency Relative.Frequency
1  [149.34:151.92]           1.077477         0.03990656
2   [151.92:154.5]           1.753623         0.06494900
3   [154.5:157.08]           5.525627         0.20465284
4  [157.08:159.66]           2.806570         0.10394703
5  [159.66:162.24]           1.042880         0.03862518
6  [162.24:164.83]           1.703999         0.06311106
7  [164.83:167.41]           3.304380         0.12238443
8  [167.41:169.99]           4.948225         0.18326760
9  [169.99:172.57]           3.738883         0.13847713
10 [172.57:175.15]           1.098337         0.04067915
>
> # Figure 5(b)
> ggInterval_hist(facedata, plotAll = T, method = "unequal-bin")$plot
```

The line plot In a line plot, data are displayed along an ordered sequence or number line (usually on the x-axis), showing how the values change over time and whether the points appear random or follow any patterns. A time plot is simply a line plot with time on the horizontal axis. For illustration, we use stock price data. A time-series plot can reveal important information about how a financial instrument's price evolves before any statistical modeling is applied. The following code reads the stock prices of three companies from Yahoo Finance—International Business Machines Corporation (IBM), Microsoft Corporation (MSFT), and Apple Inc. (AAPL). The historical data set contains open, high, low, close, adjusted close prices, and trading volumes. To illustrate an interval time-series plot, we select the low and high prices from 2021-01-15 to 2021-02-08.

```
> library(data.table)
> download.url <- "https://query1.finance.yahoo.com/v7/finance/download/"
> url.option <- "?period1=1609459200&period2=1640995200&interval=1d&events=
history&includeAdjustedClose=true"
>
> ibm <- fread(paste0(download.url, "IBM", url.option))
Downloaded 18703 bytes...> msft <- fread(paste0(download.url, "MSFT", url.option))
Downloaded 18941 bytes...> aapl <- fread(paste0(download.url, "AAPL", url.option))
Downloaded 19020 bytes...>
> from <- 10
```

(a)



(b)

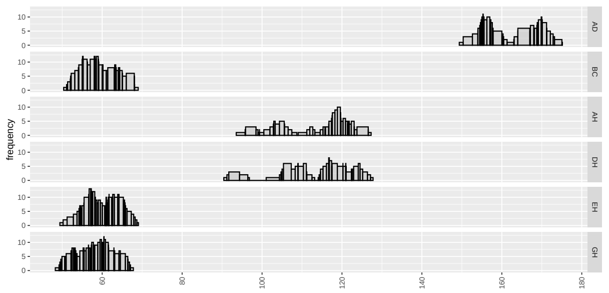


Fig 5. Histograms of the face recognition data. (a) Equidistant-bin. (b) Non-equidistant-bin.

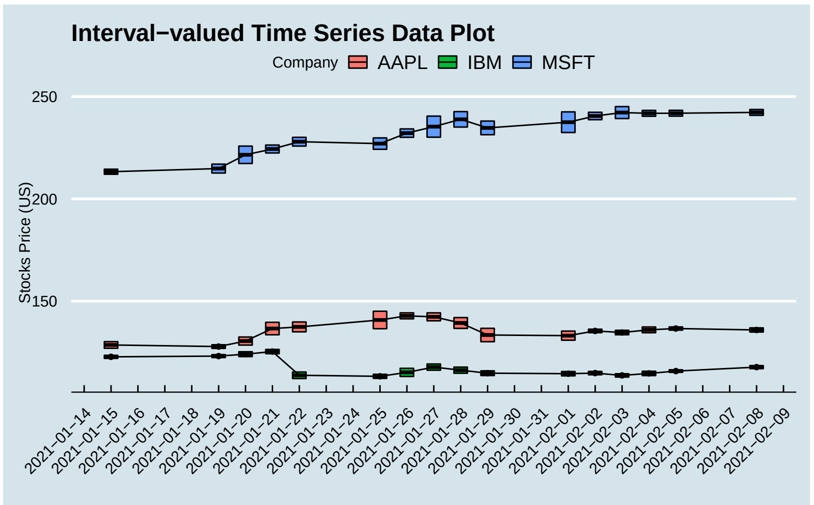


Fig 6. Interval-valued time series.

```
> to <- 25
> Company <- rep(c("IBM", "MSFT", "AAPL"), each = to - from + 1)
> DLH <- c("Date", "Low", "High")
> stock.data <- rbind(ibm[from:to, ..DLH], msft[from:to, ..DLH], aapl[from:to, ..DLH])
>
> head(stock.data)
Date       Low      High
1: 2021-01-15 122.0554 123.5564
2: 2021-01-19 122.4570 123.8910
3: 2021-01-20 122.9063 125.2964
4: 2021-01-21 124.3308 126.4245
5: 2021-01-22 112.1989 115.3920
6: 2021-01-25 112.2849 114.2830
```

The interval time-series plot shown in Figure 6 is produced using the functions `ggInterval_index` and `geom_line`, combined with an economist theme. For each company, a line connects the midpoints of the high and low prices at every time point. During this period, MSFT's high/low price range exceeds those of the other two companies and generally rises, indicating an upward trend. A clear downturn in IBM's stock price is evident around 2021-01-21, with no obvious pattern in the fluctuations thereafter. Between 2021-01-21 and 2021-02-01, the stock-price variability of all three companies appears to increase slightly. Note that if interval-valued time-series data are obtained by aggregating observed values within fixed time intervals (e.g., days or months) sequentially over time, the above code for generating the line plot is still applicable.

```
> # Figure 6
> library(ggthemes)
> stock.data.LH.i <- classic2sym(stock.data, groupby = "customize",
+                                minData = stock.data$Low,
+                                maxData = stock.data$High)$intervalData
>
> ggInterval_index(stock.data.LH.i, aes(y = V1, x = stock.data$Date,
+                                       group = Company, fill = Company)) +
+   geom_line() +
+   scale_x_date(date_breaks = "1 day") +
+   labs(title = "Interval-valued Time Series Data Plot",
+        x = "", y = "Stocks Price (US)", fill = "Company") +
+   theme_economist() +
+   theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
```

The min-max plot The min–max plot is a two-dimensional scatterplot in which the x-axis represents the minima and the y-axis represents the maxima of the intervals. Each minimum–maximum pair is connected by a segment, and a 45-degree diagonal reference line is superimposed. Variations within intervals are thus easily visualized. In the code chunk below, `ggInterval_minmax` produces the min–max plots for the face data shown in Figure 7. We observe that variables EH, BC, and GH exhibit relatively large interval variability. In addition, the argument `scaleXY = "global"` is provided to ensure that the x- and y-axis limits are identical across all min–max plots.

```
> # Figure 7
> mm.plot <- function(x){
+   plot.var <<- x
+   ggInterval_minmax(facedata, aes(facedata[[plot.var]], size = 2)) +
```

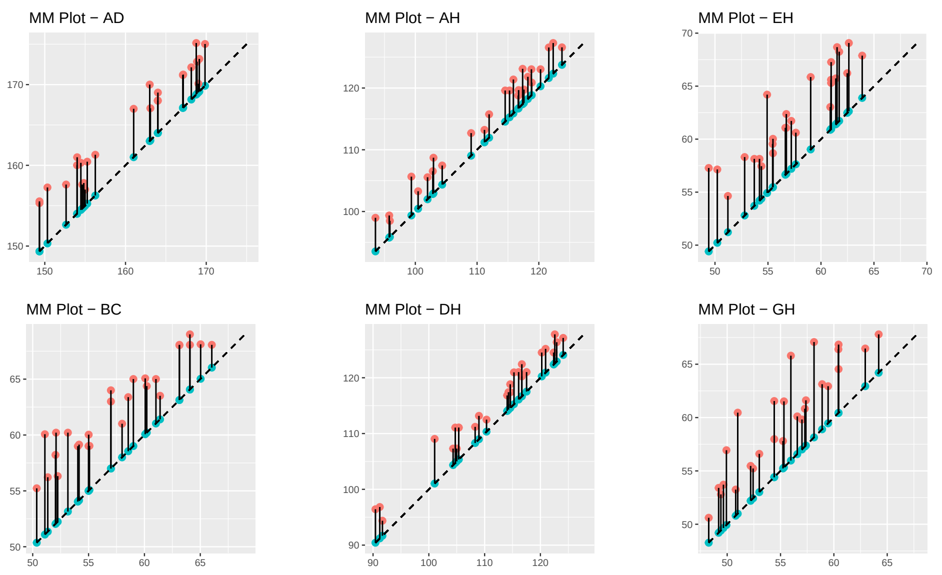


Fig 7. The min-max plot for the six variables of the face recognition data.

```
+   theme(legend.position = "none") +
+   coord_fixed(ratio = 1)
+ }
> mm.plot.list <- lapply(1:6, mm.plot)
> library(gridExtra)
> marrangeGrob(mm.plot.list, nrow = 2, ncol = 3, top = "")
```

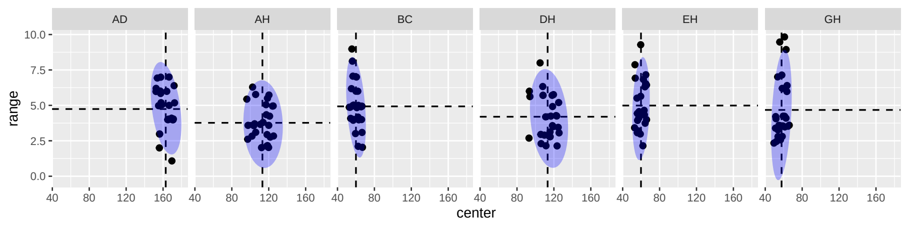
The center-range plot As an alternative, the literature often represents intervals by their center and range. A center–range plot is a two-dimensional scatterplot in which the x-axis shows the centers and the y-axis shows the interval ranges. Figure 8(a) and (b) display center–range plots for the six variables in the face-recognition data, on the original and standardized scales, respectively, created with the `ggInterval_centerRange` function. The `stat_ellipse` layer overlays bivariate-normal data ellipses. This visualization helps researchers understand the relationship between centers and ranges. After standardization, AH and DH exhibit relatively weak center–range correlations, whereas variable BC shows a pronounced negative correlation.

```
> # Figure 8(a)
> ggInterval_centerRange(facedata, aes(size = 1.5), plotAll = T) +
+   stat_ellipse(geom = "polygon", fill = "blue", alpha = 0.3)
>
> # Figure 8(b)
> facedata.scale <- scale(facedata)$intervalData
> ggInterval_centerRange(facedata.scale, aes(size = 1.5), plotAll = T) +
+   stat_ellipse(geom = "polygon", fill = "blue", alpha = 0.3)
```

Bivariate plots

Bivariate plots provide a means of assessing how variables relate to each other when they are measured on the same sample of subjects. Several basic characteristics of this

(a)



(b)

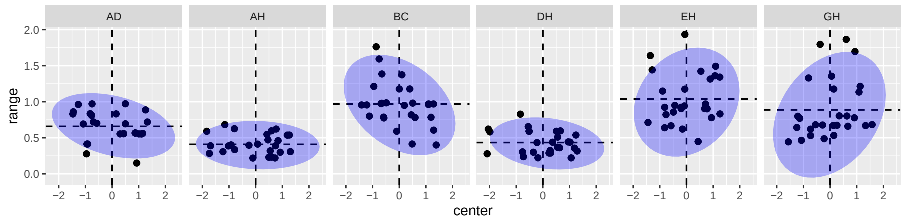


Fig 8. The center-range plot for the six variables of the face recognition data. (a) Original scale data. (b) The standardized data.

relationship are of interest, its form, its strength, and its dependence on external factors. We have implemented 2D scatter plots, 2D histograms, and time-series plots.

The 2D scatterplot The function `ggInterval_scatter` implements a two-dimensional scatterplot for interval-valued observations. The following code chunk generates a scatterplot of variables AD and BC from the face-recognition data, as shown in Figure 9(a). With `scale_fill_brewer`, rectangles representing the same subject are filled with identical colors. Variables AD and BC exhibit a positive correlation, and two main clusters can be distinguished: one in which both measurements are relatively large and another in which they are relatively small.

```
> # Figure 9(a)
> ggInterval_scatter(facedata, aes(x = BC, y = AD, fill = Subjects),
+                            col = "black") +
+   scale_fill_brewer(palette = "Set1") +
+   labs(fill = "Subjects")
```

The 2D histogram In a similar manner to Equation (1), we can get the joint histogram of (X_1, X_2) as follows.

1. Let $I_j = [\min_i a_{ij}, \max_i b_{ij}], i = 1, 2, \cdots, n$ be the interval that spans all the observed values of $X_j,\ j = 1, 2$.
2. Partition I_j into r_j subintervals $I_{j,g_j} = [\xi_{j,g_j-1}, \xi_{j,g_j})$ of equal width $\|I_{j,g_j}\|, g_j = 1, 2, \cdots, r_j,\ j = 1, 2$.
3. Define the joint histogram for X_1 and X_2 by plotting $\{R_{g_1g_2}, p_{g_1g_2}\}$ over the rectangles $R_{g_1g_2} = \{[\xi_{1,g_1-1}, \xi_{1,g_1}) \times [\xi_{2,g_2-1}, \xi_{2,g_2})\}$, $g_1 = 1, \cdots, r_1, g_2 = 1, \cdots, r_2$, where
$$p_{g_1g_2} = \frac{1}{n}\sum_{i=1}^{n}\frac{\|Z_i \cap R_{g_1g_2}\|}{\|Z_i\|}$$
is the probability an arbitrary individual description vector lies in the rectangle $R_{g_1g_2}$ and $Z_i = [a_{i1}, b_{i1}) \times [a_{i2}, b_{i2})$.

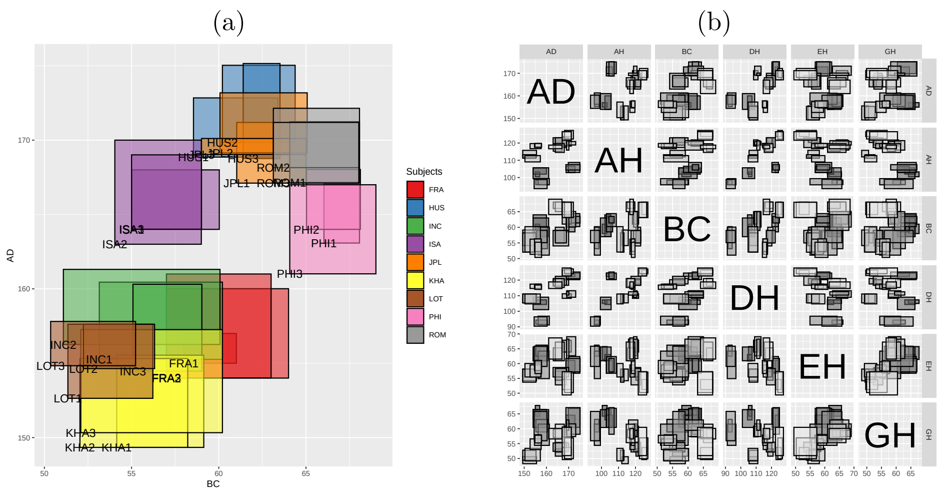


Fig 9. (a) The scatterplot of the variables AD against BC and (b) the scatterplot matrix of the face recognition data.

If the volume on the rectangle $R_{g_1 g_2}$ of the joint histogram represents the probability, its height would be

$$f_{g_1 g_2} = [(\xi_{1,g_1} - \xi_{1,g_1-1}) \times (\xi_{2,g_2} - \xi_{2,g_2-1}]^{-1} p_{g_1 g_2}.$$

That is, $f_{g_1 g_2}$ is the number of observations that fall within the rectangle $R_{g_1 g_2}$ and is not necessarily an integer (except in the special case of classical data). The relative frequency with which an arbitrary description vector lies in $R_{g_1 g_2}$ is therefore $p_{g_1 g_2} = f_{g_1 g_2}/n$. The function `ggInterval_2Dhist` in the following code chunk produces the 2D histogram of variables AD and BC with 10×10 bins, as shown in Figure 10(a). Each cell is color-coded, and its frequency is displayed as a number. The plot is rendered using the **ggplot2** layer functions `geom_text` and `scale_fill_distiller`. The 2D histogram in Figure 10(a) reveals the same pattern observed in the 2D scatterplot of variables AD and BC.

```
> # Figure 10(a)
> face.2dh <- ggInterval_2Dhist(facedata, aes(x = BC, y = AD, col = "white"),
+                               xBins = 10, yBins = 10, addFreq = F)
>
> face.2dh$plot +
+   labs(fill = 'Frequency', title = "") +
+   #coord_fixed(ratio = 1) +
+   scale_fill_distiller(palette = "Blues", direction = 1) +
+   geom_text(data = . %>% dplyr::filter(round(.data$freq, 2) != 0),
+             aes(x = (x1 + x2) / 2, y = (y1 + y2) / 2,
+                 label = round(freq, 2) %>% as.character %>%
+                   gsub(pattern = "^0", replacement = "")))
```

Multivariate plots

To visualize multivariate data sets, we have implemented scatterplot matrices, 2D histogram matrices, radar plots, and image plots.

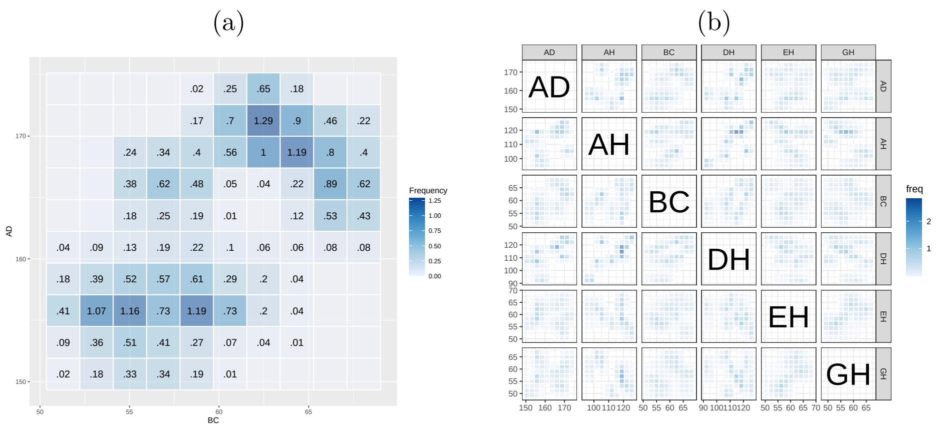


Fig 10. (a) The 2D histogram of the variables AD against BC of the face recognition data. (b) The 2D histogram matrix of the face recognition data.

The scatterplot matrix A scatterplot matrix is a common tool for visualizing multivariate data in two dimensions. It presents the bivariate relationships among all variable pairs in a grid of scatterplots. Each panel displays the relationship between a specific pair of variables, enabling users to examine many associations within a single chart. As shown in Figure 9(b), the scatterplot matrix for the six variables in the face data set is produced by the following code.

```
> # Figure 9(b)
> ggInterval_scaMatrix(facedata)
```

The interval-valued observations are represented by grayscale rectangles. Owing to facial symmetry, the pairs AH, DH and EH, GH are strongly positively correlated. In addition, the nine subjects can be readily distinguished by variables AD and AH. By contrast, negative correlations are observed between the pairs AH, GH and DH, EH, which measure complementary distances on opposite sides of a person's face.

The 2D histogram matrix A 2D histogram matrix visualizes bivariate relationships between variables through a grid of 2D histograms, analogous to a scatterplot matrix. As shown in Figure 10(b), the 2D histogram matrix for the six variables in the face data is generated by the function `ggInterval_2DhistMatrix`, which yields results similar to those produced by scatterplot matrices. The logical arguments `addFreq` and `removeZero` control whether frequencies are displayed in the rectangles and whether cells with zero frequency are removed.

```
> # Figure 10(b)
> ggInterval_2DhistMatrix(facedata, aes(col = "white"), xBins = 10, yBins = 10,
+                             removeZero = T, addFreq = F) +
+   scale_fill_distiller(palette = "Blues", direction = 1)
```

The image plot The image plot, also called a data image or heatmap, is a dimension-free method for visualizing data tables. Each column corresponds to the index image of a specific variable. Figures 11(a) and (b) present image plots for the face data, produced with `ggInterval_indexImage` and a predefined sequential colour spectrum. The two heatmaps differ in their display modes, column versus matrix, controlled by the logical argument `column_condition`.

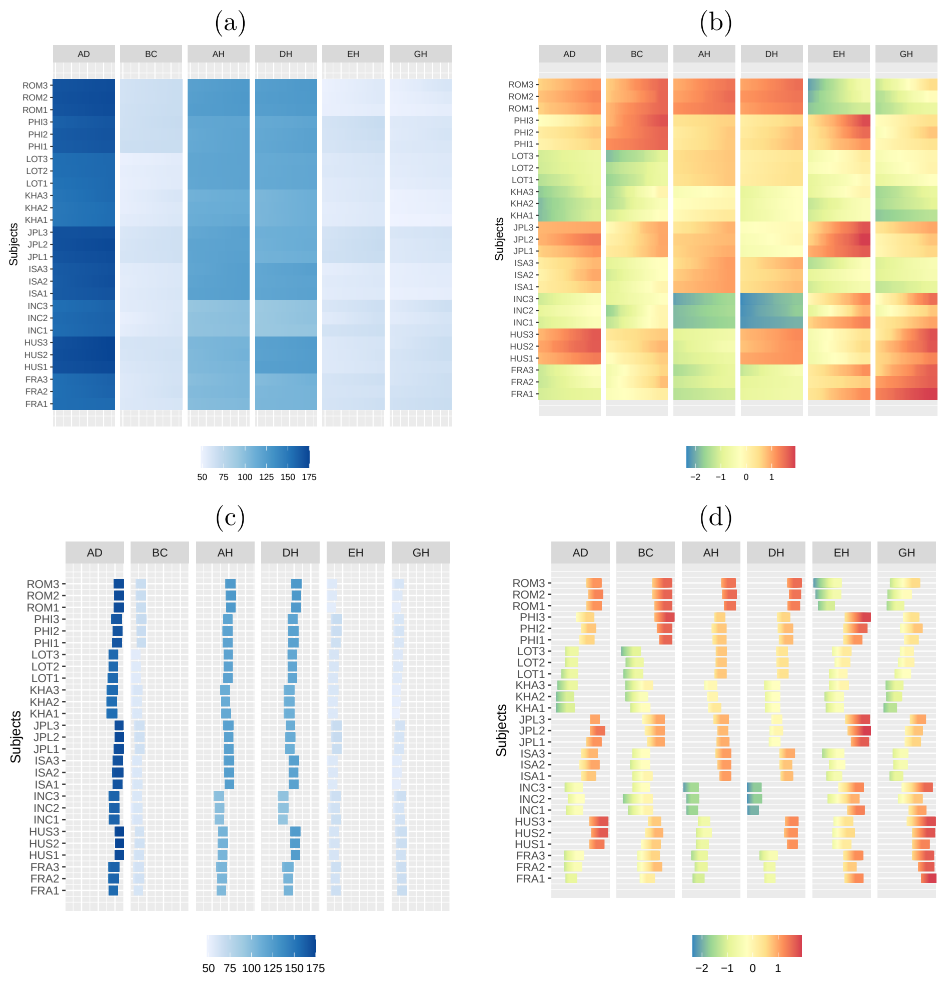


Fig 11. The image plot for the face recognition data with (a) matrix and (b) column display conditions. The non-equi-width image strips representation of the image plot with (c) matrix and (d) column display conditions.

```
> # Figure 11(a): matrix Condition
> ggInterval_indexImage(facedata, plotAll = T, full_strip = T,
+                       column_condition = F) +
+   scale_colour_distiller(palette = "Blues", direction = 1) +
+   labs(x = "Subjects") +
+   theme(axis.title.x=element_blank(), axis.text.x=element_blank(),
+         axis.ticks.x=element_blank())
>
> # Figure 11(b): columns Condition
> ggInterval_indexImage(facedata, plotAll = T, full_strip = T) +
+   scale_colour_distiller(palette = "Spectral") +
+   labs(x = "Subjects")
```

Using the column display condition, the entire colour spectrum spans the full range of values for each variable independently. This option is appropriate when the variables differ greatly in scale or are measured in different units, because it makes their distributions easier to compare visually. Under the matrix display condition, however, the colour

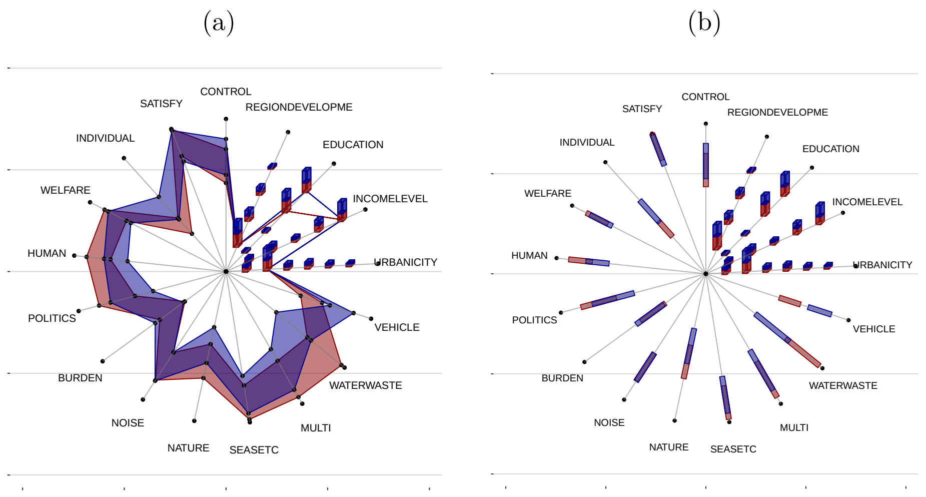


Fig 12. The radar plots of the Environmental questionnaire data. (a) The polygon representation. (b) The rectangle representation.

spectrum represents the range of the entire data matrix. The choice of display condition thus mimics a data transformation; for example, the column display condition is equivalent to standardising each variable.

Figure 11(c) and (d) present alternative versions of the image plots in Figure 11(a) and (b), where the colour strips correspond to interval ranges of varying widths. This behaviour is controlled by the logical argument `full_strip = F`. These image plots convey information similar to that in the index plots in Figure 3. Because heatmaps rely heavily on colour to encode numerical values, effective ordering of rows and columns is crucial for revealing structural patterns. We therefore recommend applying appropriate seriation and/or clustering algorithms to the rows and columns of the table before drawing an image plot [35, 36].

The radar plot A radar plot, also called a spider or star plot, displays multivariate data in a two-dimensional chart with radial axes emanating from a common center, one axis per variable (quantitative or qualitative). Consequently, radar plots help identify variables that take high, low, or similar values and highlight potential outliers. Extending them to interval-valued and/or modal multivalued data is straightforward. For interval-valued data, the minima and maxima of each variable are plotted on their respective axes, and all minima (and, separately, all maxima) are connected to form filled polygons. Modal multivalued variables can be shown with stacked bars, whose segments represent the percentage associated with each category.

Figure 12(a) shows the radar plot for the 4th and 6th symbolic objects in the environmental questionnaire data, created with the function `ggInterval_radar` and the argument `plotPartial = c(4, 6)`. Four probabilistic variables are displayed as stacked bars in the chart. By default, the `type` argument produces a polygon representation for interval variables. When `type = "rect"` is specified, intervals are depicted as rectangles, as in Figure 12(b); this representation is also known as the three-dimensional zoom star [31]. Grid lines connecting the axes serve as guides and can be toggled with the logical argument `base_circle`. Setting `inOneFig = TRUE` displays all observations in a single chart; otherwise, each observation appears in a separate radar plot.

```
> # Figure 12(a)
> library(ggthemes)
```

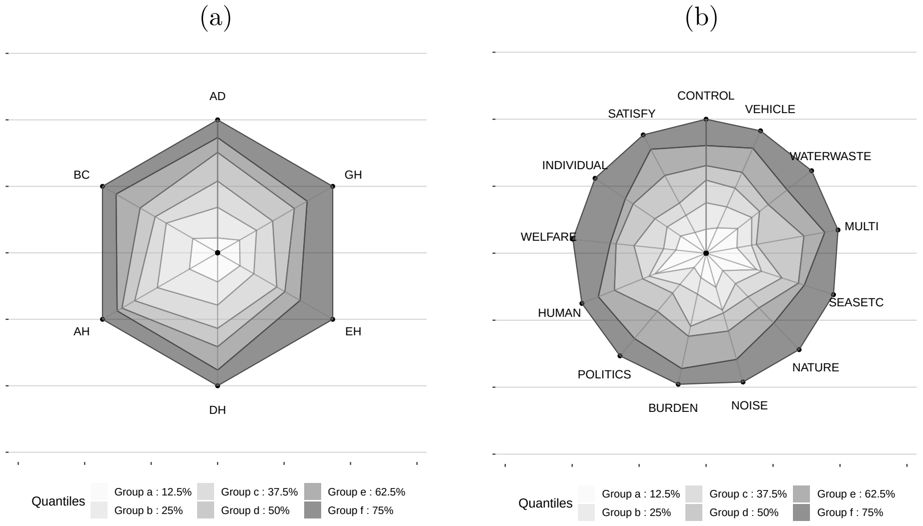


Fig 13. The quantile plots for the face recognition data and the Environmental questionnaire data (interval-valued variables only).

```
> ggInterval_radar(Environment, plotPartial = c(4, 6),
+                  showLegend = F, base_circle = F, base_lty = 1,
+                  addText = F, addText_modal = F) +
+   scale_fill_manual(values = c("darkred", "darkblue")) +
+   scale_color_manual(values = c("darkred", "darkblue")) +
+   labs(title = "") +
+   theme_hc()
```

The quantiles plot The quantile plot in our implementation is obtained directly from the radar plot by calling `ggInterval_radar` with `type = "quantile"` and specifying the desired number of quantiles via `quantileNum`. Figures 13(a) and (b) display quantile plots for the face-recognition data and the environmental-questionnaire data, respectively, considering only interval-valued variables. The distribution of each variable can be readily visualized and compared. For example, AH in the face data is left-skewed, whereas POLITICS in the environmental-questionnaire data is likewise left-skewed.

```
> # Figure 13(a)
> p <- ggInterval_radar(facedata, plotPartial = 1, base_circle = F,
+                  base_lty = 1, type = "quantile", quantileNum = 5,
+                  showLegend = T, Drift = 0) +
+   scale_fill_brewer(palette = "Greys") +
+   labs(title = "", fill = "Quantiles") +
+   theme_hc()
```

Generalization and other functions

Working with ggplot2 for customized plots

Package **ggVisInterval** extends **ggplot2**, so most features available in **ggplot2** can also be applied within **ggVisInterval** to create more advanced graphics. The example code below demonstrates how to create a customized index plot for the variable AD in the face data,

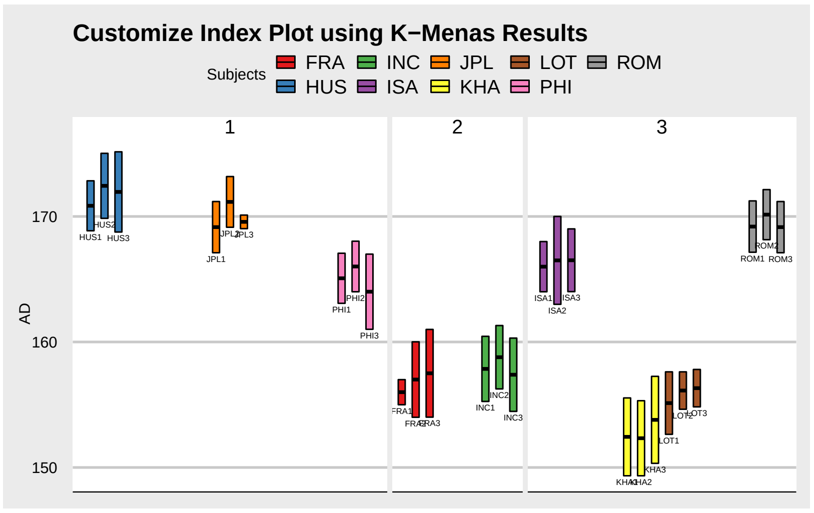


Fig 14. A customized index plot for the variable AD of face recognition data based on the K-means results.

based on K-means results, using `ggInterval_index`. Figure 14 shows the resulting plot, which employs `geom_text`, `labs`, `scale_fill_brewer`, `facet_grid`, and `theme`.

```
> # Figure 14
> set.seed(1234567890)
> facedata.tmp <- facedata
> facedata.tmp$cluster <- sym.kmeans(facedata.tmp, k = 3)$cluster
>
> ggInterval_index(facedata.tmp, aes(y = AD, fill = Subjects)) +
+   geom_text(aes(x = 1:f.nr, y = .data$AD$min,
+                 label = rownames(facedata), vjust = 1.5), size = 2) +
+   scale_fill_brewer(palette = "Set1") +
+   labs(title = "Customize Index Plot using K-Menas Results", fill = "Subjects") +
+   facet_grid(cols = vars(facedata.tmp$cluster),
+              scales = "free_x", space = "free_x") +
+   theme_economist_white() +
+   theme(axis.title.x=element_blank(), axis.text.x=element_blank(),
+               axis.ticks.x=element_blank())
```

The previous section presented examples that showcase the advanced, aesthetically pleasing features of **ggplot2**, which in turn enhance **ggVisInterval**. A few remarks follow.

Geometries The common visual representations provided by **ggplot2** geometries—such as `geom_area`, `geom_line`, `geom_point`, `geom_segment`, and `geom_text`—can be readily applied to graphics created with **ggVisInterval**.

Scale The **ggVisInterval** package supports the natural use of `scale_*` functions from **ggplot2**, including `scale_(x|y)_*` and `scale_(fill|colour)_`.

Theme Themes from **ggplot2**, **ggthemes** [37], or **ggpubr** [38] can be applied to graphs created with **ggVisInterval**.

Facets The plot functions we developed such as `ggInterval_index`, `ggInterval_centerRange`, `ggInterval_minmax`, and `ggInterval_scatter`, are compatible with the faceting facilities of **ggplot2**. Faceting with either `facet_grid` or `facet_wrap` produces multiple subplots based on different subsets of the data. Specifically, `facet_grid` creates a matrix of panels determined by both row and column faceting variables, whereas `facet_wrap` arranges panels for the levels of a single faceting variable.

Although most additional layers or scales can be applied to a **ggVisInterval** graphic, some may conflict with native **ggplot2** behaviour. The complexity of interval data can pose challenges that are hard to accommodate with the default aesthetics. Examples include `geom_bar`, `geom_count`, `geom_density`, and `geom_dotplot`. For graphics affected by these issues, it is often preferable to create a new plot rather than add an extra layer.

Working with other SDA packages

Symbolic objects produced by SDA-related packages can be stored in various formats. To improve their compatibility with **ggVisInterval**, we offer a transformation function, `classic2sym`, that converts these objects into a form suitable for visualization with **ggVisInterval**. We illustrate this capability using two SDA packages: **HistDAWass** [39] and **MAINT.Data** [40].

HistDAWass The built-in `BLOOD` data set in the **HistDAWass** package is histogram-valued. It comprises 14 groups of patients described by three variables. Using the `get.MatH.stats` function, we obtain the minima and maxima of each histogram. We then convert these values into intervals with the `classic2sym` function, preparing them for use in **ggVisInterval**.

```
> library(HistDAWass)
> BLOOD <- HistDAWass::BLOOD
> blood.min <- get.MatH.stats(BLOOD, stat = "min")
> blood.max <- get.MatH.stats(BLOOD, stat = "max")
> blood <- data.frame(blood.min, blood.max)
> blood.sym <- classic2sym(blood,
+                          groupby = "customize",
+                          minData = blood[, 2:4],
+                          maxData = blood[, 6:8])
> blood.i <- blood.sym$intervalData
> colnames(blood.i) <- get.MatH.main.info(BLOOD)$varnames
> head(as.data.frame(blood.i), 5)
Cholesterol      Hemoglobin      Hematocrit
1  [80.00 : 240.00] [12.00 : 15.00] [35.00 : 47.00]
2  [80.00 : 240.00] [10.50 : 14.00] [31.00 : 44.00]
3  [95.00 : 245.00] [10.50 : 14.00] [31.00 : 43.50]
4 [105.00 : 260.00] [10.50 : 14.00] [31.00 : 42.50]
5 [115.00 : 260.00] [10.80 : 13.60] [31.00 : 42.50]
```

MAINT.Data Interval-valued data can also be expressed through midpoints and ranges. A case in point is the built-in `Abalone` data set in the **MAINT.Data** package. This data set comprises 24 observations and seven interval-valued variables, stored as midpoints and log ranges. These intervals were obtained by aggregating the Abalone data set (UCI Machine Learning Repository) by sex and age. Using `classic2sym`, we convert the midpoints and ranges back into intervals defined by their lower and upper bounds.

```
> library(MAINT.Data)
> AbaloneIdt <- MAINT.Data::AbaloneIdt
> a.range <- exp(AbaloneIdt@LogR)
> a.midp <- AbaloneIdt@MidP
> abalone <- data.frame(a.midp - a.range / 2, a.midp + a.range / 2)
> abalone.sym <- classic2sym(abalone,
+                           groupby = "customize",
+                           minData = abalone[, 1:7],
+                           maxData = abalone[, 8:14])
> colnames(abalone.sym$intervalData) <- AbaloneIdt@VarNames
> abalone.i <- abalone.sym$intervalData %>%
+   cbind(Aba.obs = AbaloneIdt@ObsNames) %>%
+   column_to_rownames(var = "Aba.obs")
> head(abalone.i[, 1:4], 5)
Length      Diameter       Height  Whole_weight
F-10-12 [0.34 : 0.78] [0.26 : 0.63] [0.06 : 0.23] [0.21 : 2.66]
F-13-15 [0.39 : 0.82] [0.30 : 0.65] [0.10 : 0.25] [0.27 : 2.51]
F-16-18 [0.40 : 0.74] [0.32 : 0.60] [0.10 : 0.24] [0.35 : 2.20]
F-19-21 [0.49 : 0.72] [0.36 : 0.58] [0.12 : 0.21] [0.68 : 2.12]
F-23-24 [0.45 : 0.80] [0.38 : 0.63] [0.14 : 0.22] [0.64 : 2.53]
```

Visualizing intervals in 2D Principal Component space

If the output of a statistical or machine-learning method is interval-valued, **ggVisInterval** provides a convenient way to visualize it. Principal component analysis (PCA) for interval data is one such example [20, 23]. By applying PCA, users can explore data structure by projecting interval-valued observations into a two- or three-dimensional dimension-reduced (DR) subspace. Several approaches have been proposed to visualise interval objects in lower-dimensional spaces. Our study implements two of them: the maximum covering area rectangle (MCAR) and the polytope representation. Below, we show the projected intervals of the face data on the first DR plane. The interval PCA is performed with the function `sym.pca` from the **RSDA** package.

The maximum covering area rectangle The maximum covering area rectangle (MCAR; [41]) is widely used to graphically represent interval objects in a dimension-reduced (DR) subspace because of its simplicity. It represents each observation as a k-dimensional hyperrectangle (typically $k = 2$) whose sides are parallel to the coordinate axes. The principal drawback of MCAR is that these hyperrectangles are oversized in the DR space relative to the original objects in $\mathbb{R}^p$; consequently, a hyperrectangle may encompass data points that were not part of the original observation. Figure 15(a) displays the two-dimensional projection of the face data obtained by the vertices method of PCA under the MCAR representation.

```
> # Figure 15(a)
> pca.results <- RSDA::sym.pca(facedata, method = "tops")
> rownames(pca.results$Sym.Components) <- rownames(facedata)
> ggInterval_scatter(pca.results$Sym.Components, aes(Dim.1, Dim.2,
+                    fill = as.factor(Subjects)), color = "black") +
+   coord_fixed(ratio = 1) +
+   labs(x = "PCA-1", y = "PCA-2", fill = "Subjects")
```

The polytopes representation The symbolic covariance method of interval PCA with a polytope representation was introduced by [21]. The vertices of the interval-valued

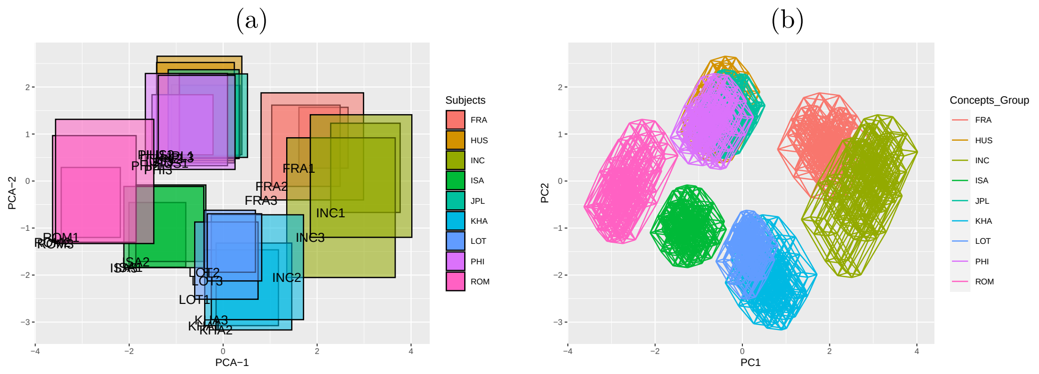


Fig 15. A 2D projection of interval PCA to the recognition face: (a) the vertices method with MCAR representation, and (b) the symbolic covariance method with the polytopes representation.

data are projected onto a lower-dimensional subspace, and their edges are reconstructed in that subspace by connecting the projected vertices, mirroring their configuration in the original space $\mathbb{R}^p$. The polytope representation refines the MCAR approach, yielding a faithful projection of the observed interval data. Figure 15(b) displays the two-dimensional projection of the face data obtained with the symbolic covariance PCA method under the polytope representation; it reproduces Figure 4(a) of [21].

```
> # Figure 15(b)
> ggInterval_PCA(facedata, poly = T, concepts_group = as.factor(Subjects)) +
+   coord_fixed(ratio = 1)
```

Conclusion and future development

EDA and data-visualization procedures help users discover hidden patterns in data. Without graphical exploration, analyzing data solely with numerical statistics can be misleading. As noted in [42], 'finding ways to visualize data sets can be as important as ways to analyze them," and, as stressed in [43], 'EDA is often neglected by people who are eager to jump to more sophisticated analyses. It should have an important place." Consequently, developing effective EDA and visualization tools for symbolic data is essential.

In this study, we introduced the **ggVisInterval** package, which complements **ggplot2** for exploratory symbolic data analysis and focuses on interval-valued data. Among SDA-related R packages, its functionality is unique. We demonstrated its capabilities with several widely used interval-valued data sets. Future versions will incorporate additional multivariate graphics and methodological enhancements, such as revised 3D scatterplots and interactive tools. We believe **ggVisInterval** will prove valuable to the SDA community and broaden participation within statistics and data science.

Traditional graphical EDA techniques often fail when confronted with big data because of their scale. Statisticians must now analyze rapidly gathered data from diverse sources and of complex types. New, more efficient statistical and graphical methods are therefore required. SDA can address this challenge by aggregating massive data sets into interval-, histogram-, or distribution-valued forms of manageable size, after which **ggVisInterval** can be applied for swift initial exploration.

Supporting information

S1 File. The ggVisInterval package. The **ggVisInterval** package is currently available on the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=ggVisInterval`.

Acknowledgments

The authors are grateful to the editor, associate editor, and referees for their insightful comments and suggestions. This research received support from the National Science and Technology Council, Taiwan, R.O.C. (MOST 105-2118-M-032-009)

Author Contributions

Conceptualization: Han-Ming Wu

Data curation: Bo-Syue Jiang

Investigation: Han-Ming Wu

Methodology: Bo-Syue Jiang, Han-Ming Wu

Software: Bo-Syue Jiang

Supervision: Han-Ming Wu

Writing – original draft: Bo-Syue Jiang, Han-Ming Wu

Writing – review & editing: Han-Ming Wu

References

1. Tukey JW. Exploratory Data Analysis. Reading, Massachusetts: Addison-Wesley; 1977.
2. Gelman A, Buja A. Exploratory Data Analysis for Complex Models. J Comput Graph Stat. 2004;13(4):755–787.
3. Unwin A. Graphical Data Analysis with R. New York: Chapman and Hall/CRC; 2015.
4. Cook D, Buja A, Cabrera J, Hurley C. Grand tour and projection pursuit. J Comput Graph Stat. 1995;4(3):155–172.
5. Camacho J, Rodríguez-Gómez RA, Saccenti E. Group-wise principal component analysis for exploratory data analysis. J Comput Graph Stat. 2017;26(3):501–512.
6. Cui B. DataExplorer: Automate Data Exploration and Treatment [Internet]. 2020. Available from: http://boxuancui.github.io/DataExplorer/.

7. Schloerke B, Cook D, Larmarange J, Briatte F, Marbach M, Thoen E, Elberg A, Crowley J. GGally: Extension to ggplot2 [Internet]. 2021. Available from: https://CRAN.R-project.org/package=GGally.

8. Dayanand Ubrangala R K, Prasad Kondapalli R, Putatunda S. SmartEDA: Summarize and Explore the Data [Internet]. 2021. Available from: https://daya6489.github.io/SmartEDA/.

9. Yoshida K, Bartel A. tableone: Create Table 1 to Describe Baseline Characteristics with or without Propensity Score Weights [Internet]. 2022. Available from: https://github.com/kaz-yos/tableone.

10. R Core Team. R: A Language and Environment for Statistical Computing [Internet]. Vienna, Austria: R Foundation for Statistical Computing; 2021. Available from: https://www.R-project.org/.

11. Billard L, Diday E. From The Statistics of Data to The Statistics of Knowledge: Symbolic Data Analysis. J Am Stat Assoc. 2003;98(462):470–487.

12. Billard L, Diday E. Symbolic Data Analysis: Conceptual Statistics and Data Mining. New Jersey: Wiley; 2007.

13. Noirhomme-Fraiture M, Brito P. Far Beyond the Classical Data Models: Symbolic Data Analysis. Stat Anal Data Min: The ASA Data Sci J. 2011;4(2):157–170.

14. Diday E. Thinking by Classes in Data Science: the Symbolic Data Analysis Paradigm. WIREs Comput Stat. 2016;8(5):172–205.

15. Wickham H. ggplot2: Elegant Graphics for Data Analysis. New York: Springer-Verlag; 2016. ISBN 978-3-319-24277-4. Available from: https://ggplot2.tidyverse.org.

16. Rodriguez O. RSDA: R to Symbolic Data Analysis [Internet]. 2022. Available from: http://www.oldemarrodriguez.com.

17. Hamilton NE, Ferry M. ggtern: Ternary Diagrams Using ggplot2. J Stat Softw Code Snippets. 2018;87(3):1–17. doi:10.18637/jss.v087.c03.

18. Maag JL. gganatogram: An R Package for Modular Visualisation of Anatograms and Tissues Based on ggplot2. F1000Research. 2018;7:1576.

19. Patil I. Visualizations with Statistical Details: The 'ggstatsplot' Approach. J Open Source Softw. 2021;6(61):3167. doi:10.21105/joss.03167. Available from: https://doi.org/10.21105/joss.03167.

20. Douzal-Chouakria A, Billard L, Diday E. Principal Component Analysis for Interval-valued Observations. Stat Anal Data Min. 2011;4(2):229–246.

21. Le-Rademacher J, Billard L. Symbolic Covariance Principal Component Analysis and Visualization for Interval-valued Data. J Comput Graph Stat. 2012;21(2):413–432.

22. Lauro CN, Palumbo F. Principal Component Analysis of Interval Data: A Symbolic Data Analysis Approach. Comput Stat. 2000;15(1):73–87.

23. Lauro NC, Verde R, Irpino A. Principal component analysis of symbolic data described by intervals. In: Diday E, Noirhomme-Fraiture M, editors. Symbolic Data Analysis and the SODAS Software. 2008. pp. 279–312.

24. Neto EdAL, Cordeiro GM, de Carvalho FdA. Bivariate Symbolic Regression Models for Interval-valued Variables. J Stat Comput Simul. 2011;81(11):1727–1744.

25. Domingues MA, de Souza RM, Cysneiros FJA. A Robust Method for Linear Regression of Symbolic Interval Data. Pattern Recognit Lett. 2010;31(13):1991–1996.

26. Billard L, Diday E. Regression Analysis for Interval-valued Data. In: Data Analysis, Classification, and Related Methods. Springer; 2000. pp. 369–374.

27. Xu W. Symbolic Data Analysis: Interval-valued Data Regression. Ph.D. thesis, University of Georgia Athens, GA; 2010.

28. Malerba D, Esposito F, Gioviale V, Tamma V. Comparing Dissimilarity Measures for Symbolic Data Analysis. In: Proceedings of Exchange of Technology and Know-how and New Techniques and Technologies for Statistics; 2001. pp. 473–481.

29. Umbleja K, Ichino M, Yaguchi H. Improving Symbolic Data Visualization for Pattern Recognition and Knowledge Discovery. Visual Informatics. 2020;4(1):23–31.

30. Leroy B, Chouakria A, Herlin I, Diday E. Approche geometrique et classification pour la reconnaissance de visage. In: Congres de Reconnaissance des formes et Intelligence Artificielle; 1996. pp. 548–557.

31. Diday E, Noirhomme-Fraiture M. Symbolic Data Analysis and the SODAS Software. John Wiley & Sons; 2008.

32. Bertrand P, Goupil F. Descriptive Statistics for Symbolic Data. In: Bock HH, Diday E, editors. Analysis of Symbolic Data. Studies in Classification, Data Analysis, and Knowledge Organization. Berlin, Heidelberg: Springer; 2000. pp. 106–124.

33. Billard L. Sample Covariance Functions for Complex Quantitative Data. In: Proceedings of the World IASC Conference; Yokohama, Japan; 2008. pp. 157–163.

34. Irpino A, Verde R. Basic Statistics for Distributional Symbolic Variables: A New Metric-based Approach. Adv Data Anal Classif. 2015;9:143–175.

35. Hurley CB. Clustering Visualizations of Multidimensional Data. J Comput Graph Stat. 2004;13(4):788–806.

36. Wu HM, Tien YJ, Chen CH. GAP: A Graphical Environment for Matrix Visualization and Cluster Analysis. Comput Stat Data Anal. 2010;54(2):767–778.

37. Arnold JB. ggthemes: Extra Themes, Scales and Geoms for ggplot2 [Internet]. 2021. Available from: https://github.com/jrnold/ggthemes.

38. Kassambara A. ggpubr: ggplot2 Based Publication Ready Plots [Internet]. 2020. Available from: https://rpkgs.datanovia.com/ggpubr/.

39. Irpino A. HistDAWass: Histogram-Valued Data Analysis [Internet]. 2021. Available from: https://CRAN.R-project.org/package=HistDAWass.

40. Silva PD, Brito P. MAINT.Data: Model and Analyse Interval Data [Internet]. 2021. Available from: https://CRAN.R-project.org/package=MAINT.Data.

41. Cazes P, Chouakria A, Diday E, Schektman Y. Extension de l'analyse en composantes principales a des donnees de type intervalle. Rev Stat Appl. 1997;45(3):5–24.

42. Ripley BD. How Computing has Changed Statistics. In: Celebrating Statistics: Papers in Honour of Sir David Cox on His 80th Birthday; eds. Davison A, Dodge Y, Wermuth N; 2005. pp. 197–212.

43. Borcard D, Gillet F, Legendre P. Numerical Ecology with R. New York, NY: Springer; 2011.