MD-plot Documentation

Release 2019-06-21

Michael Thrun, Tino Gehlert

Jul 01, 2023

User Guide:

CHAPTER 1

Python Package

1.1 Installation

The latest stable release can be installed from PyPi:

```
pip install md_plot
```

The source code is hosted on GitHub at: https://github.com/TinoGehlert/md_plot

1.2 Dependencies

- Python 3.5+
- pandas: 0.24.2 or higher
- NumPy: 1.16.0 or higher
- scipy: 1.1.0 or higher
- matplotlib: 3.1.0 or higher
- plotnine: 0.5.1 or higher
- unidip: 0.1.1 or higher

Windows users of Anaconda distribution should update numpy, scipy and matplotlib via conda instead of pip.

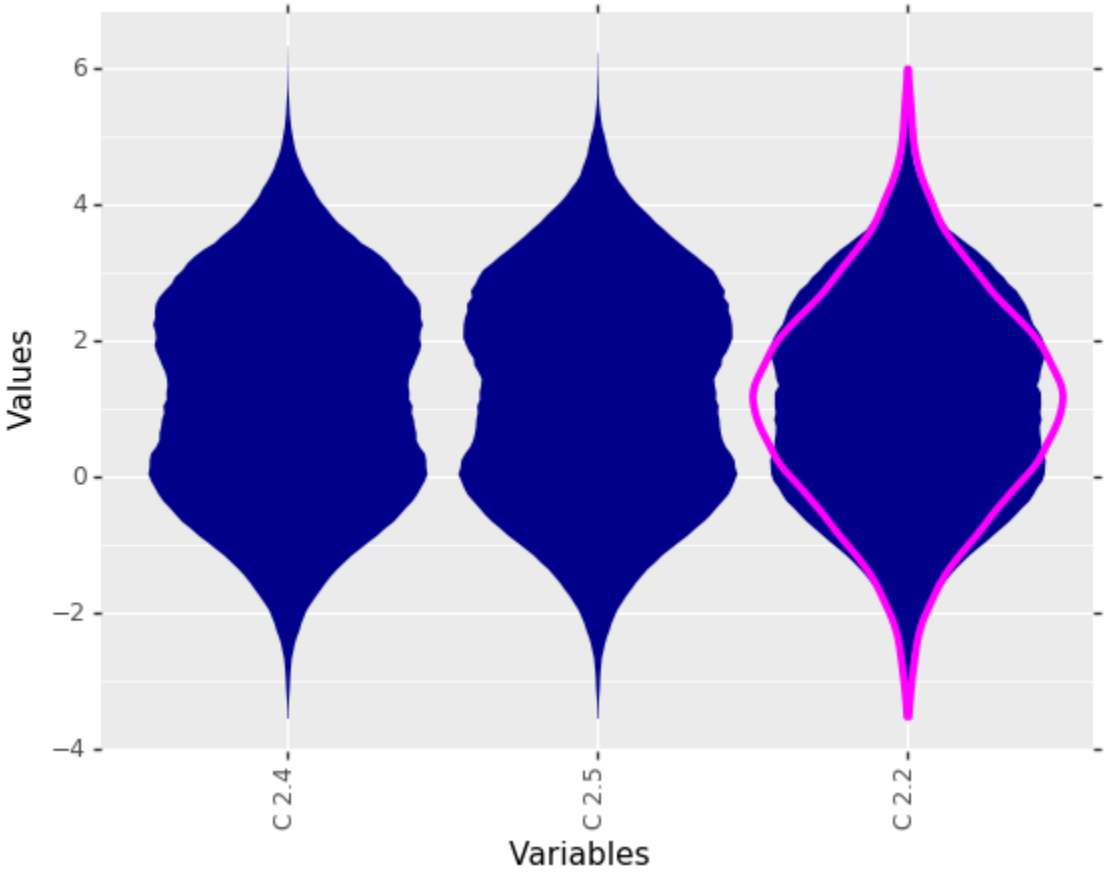
1.3 Basic Usage

```
from md_plot import MDplot, load_examples

dctExamples = load_examples()
```

(continues on next page)

(continued from previous page)

```
MDplot(dctExamples["BimodalArtificial"])
```



1.4 Future Improvements

In addition to bug fixing, these improvements to the md_plot pack-age are planned:

- Close the performance gap to the R version
- Reimplementation of dip-test based on diptest R package (uni-dip is using Monte Carlo simulations to compute the p-values, but this is not the default behavior of the diptest in R)

CHAPTER 2

Python Tutorial

2.1 Introduction

In data analysis of industrial topics such as the quality assurance of production facilities or the analysis of customer behavior, we constantly encounter univariate distributions of all kinds. From unimodal Gaussians to symmetric or skewed distributions to multimodal distributions It is of immense importance to determine the exact shape of the distribution in order to be able to select the correct further analysis steps and to be able to draw correct conclusions. Of special interest is the question, whether the given empirical distribution is composed of two or more distinct subsets of data points. Such subsets give hints to the existence of different states of the data producing process, such as, for example, healthy vs. sick patients or the existence of different diseases or treatments.

Conventional visualization methods of univariate probability density distributions have problems in the distinction of uniform versus multi-modal distributions and in visualizing capped skewed distributions correctly. With the mirrored density plot, a visualization method more suit-able for these applications was postulated [Thrun/Ultsch, 2019] in the programming language R, which is now extended to python.

The Python package *md_plot* is an implementation of the MD-Plot function of the R package *DataVisualizations* on CRAN [Thrun/Ultsch, 2018]. The use of the package is described in this technical report.

2.2 Basic Usage

2.2.1 Inbuild Samples

The md_plot package offers several samples to make getting started using the MD-Plot easier. To do this, call the function load_examples, which returns a dictionary. This dictionary contains several keys, which each provide access to a single pandas dataframe.

```
from md_plot import load_examples
dctExamples = load_examples()
```

Key	Type	Size	Value
BimodalArtificial	DataFrame	(31000, 3)	Column names: C 2.2, C 2.4, C 2.5
MTY_Clipped	DataFrame	(11194, 1)	Column names: MTY_Clipped
MuncipalIncomeTaxYield_IncomeTaxShare	DataFrame	(11194, 2)	Column names: ITS, MTY
SampleLogInome	DataFrame	(500, 1)	Column names: LogData
SkewedDistribution	DataFrame	(15000, 4)	Column names: C_0.95, C_0.6, C_1.1, C_1
StocksData2018Q1	DataFrame	(269, 51)	Column names: TotalRevenue, CostofRevenue, GrossProfit, SellingGeneral ...
UniformSample	DataFrame	(1000, 1)	Column names: UniformSample

Fig. 1: *Return of the load_examples function. The sample data is contained in a dictionary as pandas dataframes and can be used in the MDplot to reproduce the visualizations.*

These dataframes can now be used in the md_plot function.

2.2.2 Visualization

The MDplot function of the md_plot package accepts pandas series (vectors) and pandas dataframes (matrices) or anything that converts simply into these two data structures (for example, lists and numpy ar-rays) as input. Preprocessing depending on the analysis can be before done before visualization. For example the capping of values:

```
from md_plot import MDplot
dfMTY = dctExamples["MTY_Clipped"]
dfMTY = dfMTY[(dfMTY["MTY_Clipped"] >= 1800) & (dfMTY["MTY_Clipped"] <= 6000)]
MDplot(dfMTY)
```

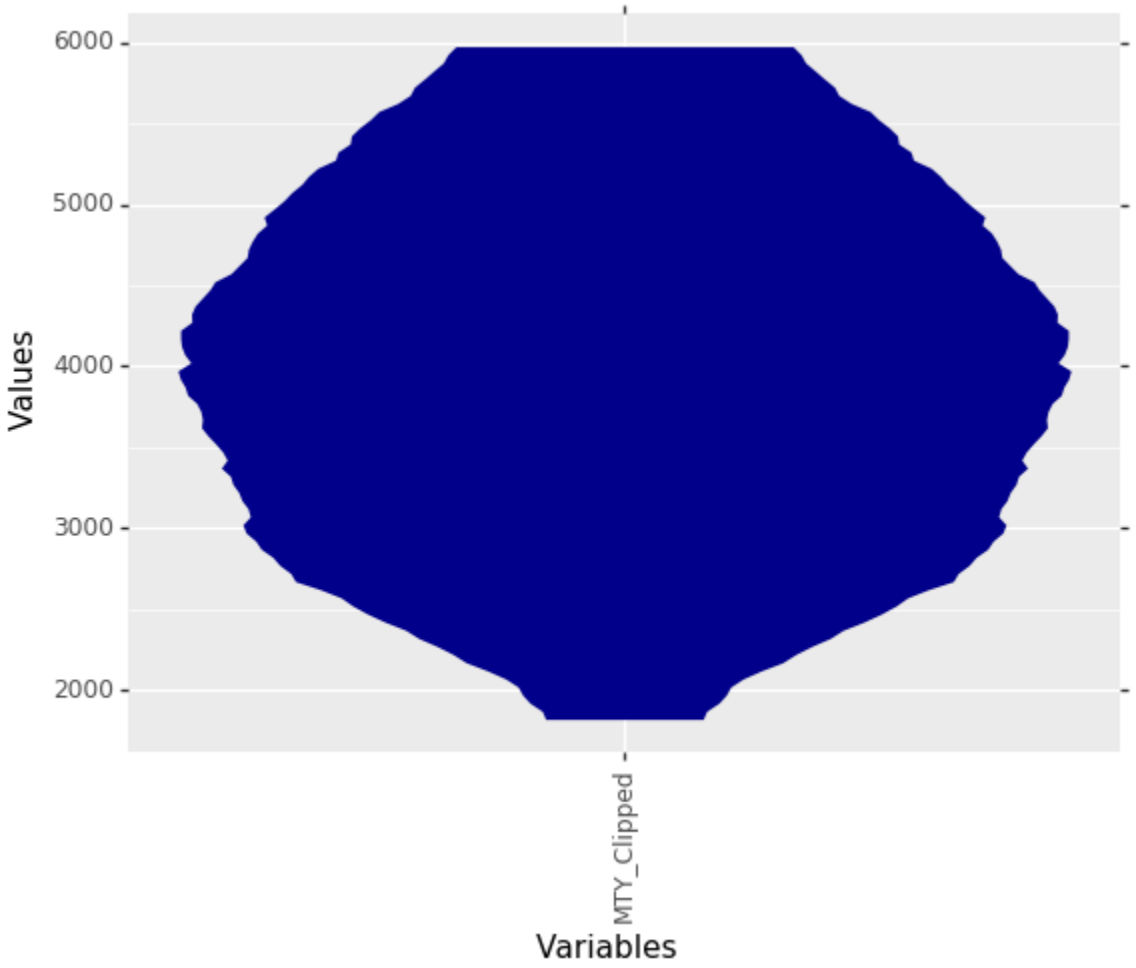


Fig. 2: *MD-Plot of capped municipality income tax yield (MTY) of Germany municipalities of 2015. Good to see are the clear capping limits at which the were cropped for the visualization.*

The function returns a ggplot object, but you can get additional in-formation by setting the *OnlyPlotOutput* parameter to False.

```
dctResult = MDplot(dctExamples['BimodalArtificial'], OnlyPlotOutput=False)
```

2.2.3 Working with long table format

The MDplot function accepts by default data in wide table format (each column represents one variable), but can also work with data in long table format (the values of all variables are contained in one column, a second column with identifiers exist) if the column names of the value column and the class column are provided.

```
MDplot(dctExamples["SkewedDistributionLongTable"], ValueColumn="value", ClassColumn=
↪"class")
```

2.2.4 Changing Layout

The layout of the ggplot object returned by the MDplot can be modi-fied by adding additional ggplot objects, e.g. a title.

```
import plotnine as p9
MDplot(dfMTY) + p9.labels.ggtitle('Capped MTY data') + p9.labels.ylab('PDE') + p9.
↪labels.xlab('Variables') + p9.theme_seaborn()
```

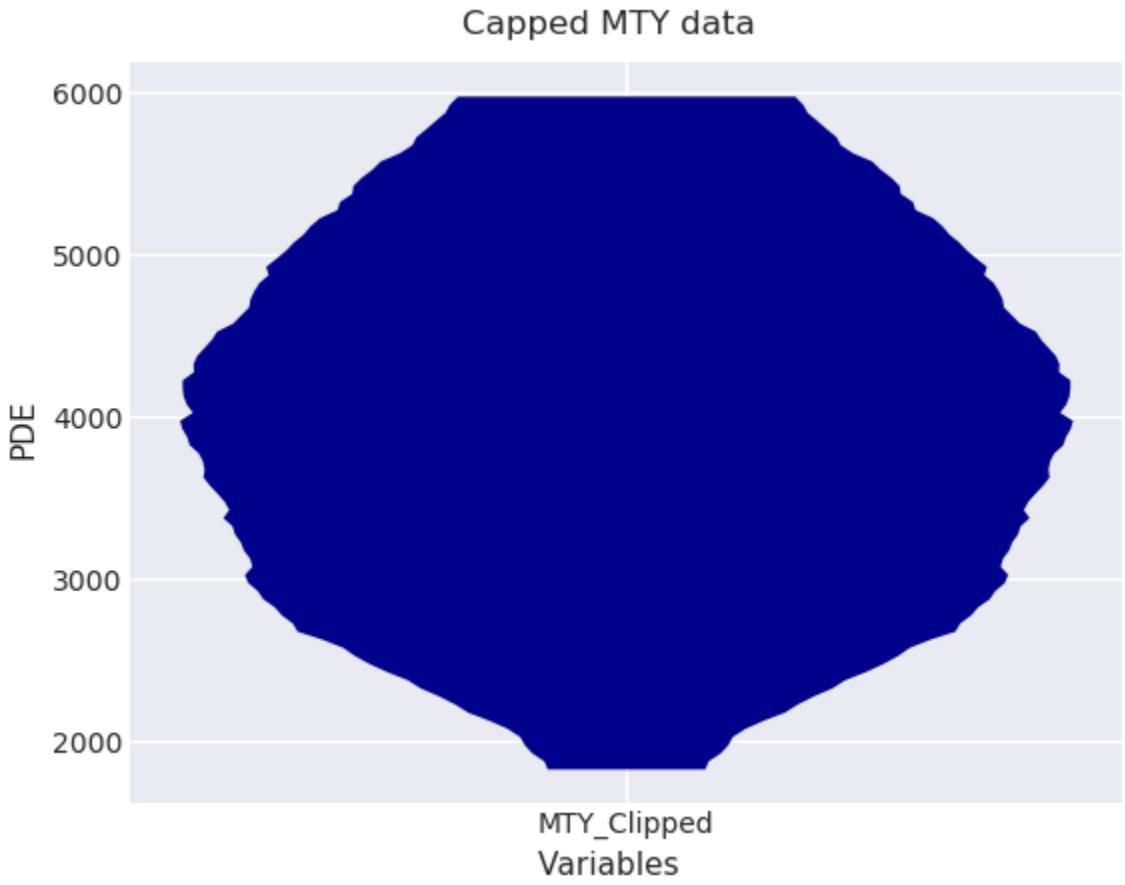


Fig. 3: *MD plot of capped MTY data with title, changed axis labels and in seaborn theme*

Further information on the layout design of ggplots in plotnine can be found in the official documentation at https://plotnine.readthedocs.io.

2.3 Advanced Usage

2.3.1 Draw a Gaussian distribution

The parameter *RobustGaussian* is used to activate or deactivate an overlay of a Gaussian distribution (this activated by default). The Gaussian distribution will only be drawn if several statistical tests have shown that the data is unimodal and not skewed. For changing the vis-ual appearance of the Gaussian distribution, the parameters *Gaussian-Color* and *GaussianLwd* (line width) are provided.

```
MDplot(dctExamples['BimodalArtificial'], RobustGaussian=False)
```

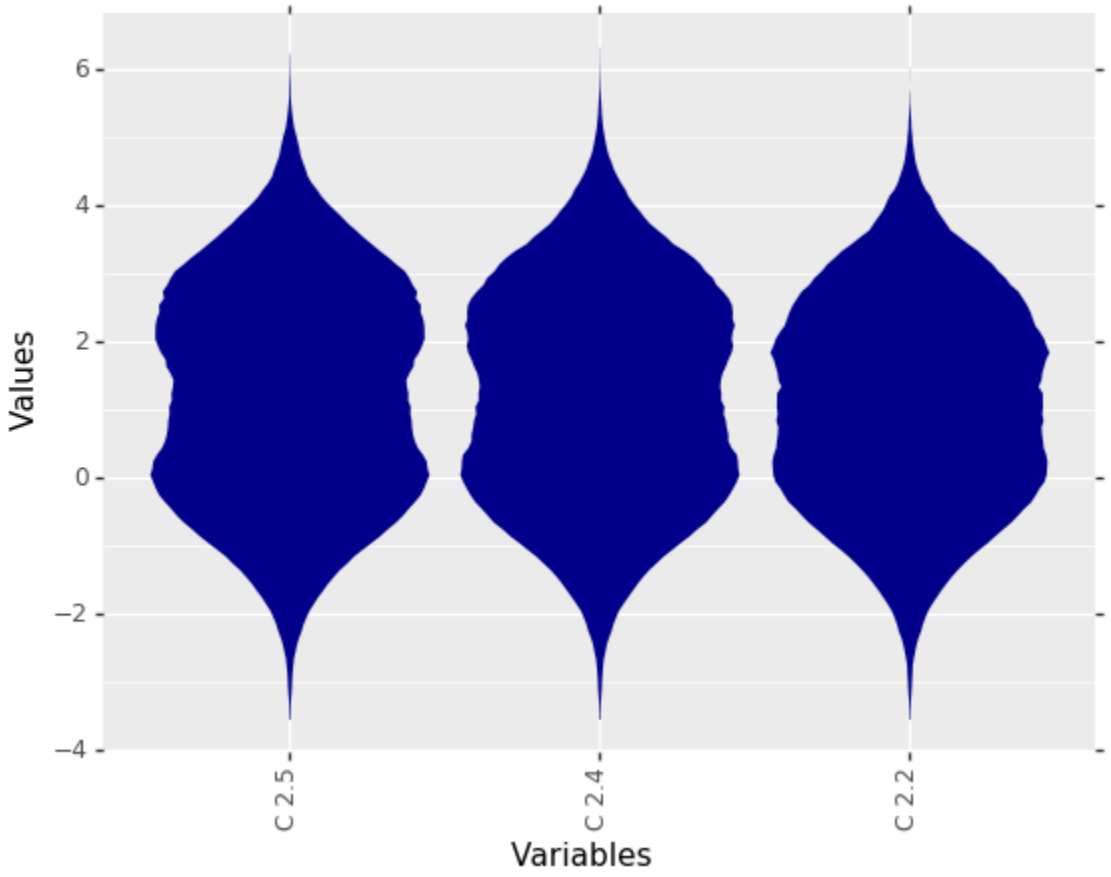


Fig. 4: *Visualization of bimodal data without drawn Gaussian distribution.*

```
MDplot(dctExamples['BimodalArtificial'], GaussianColor='green', GaussianLwd=2.5)
```

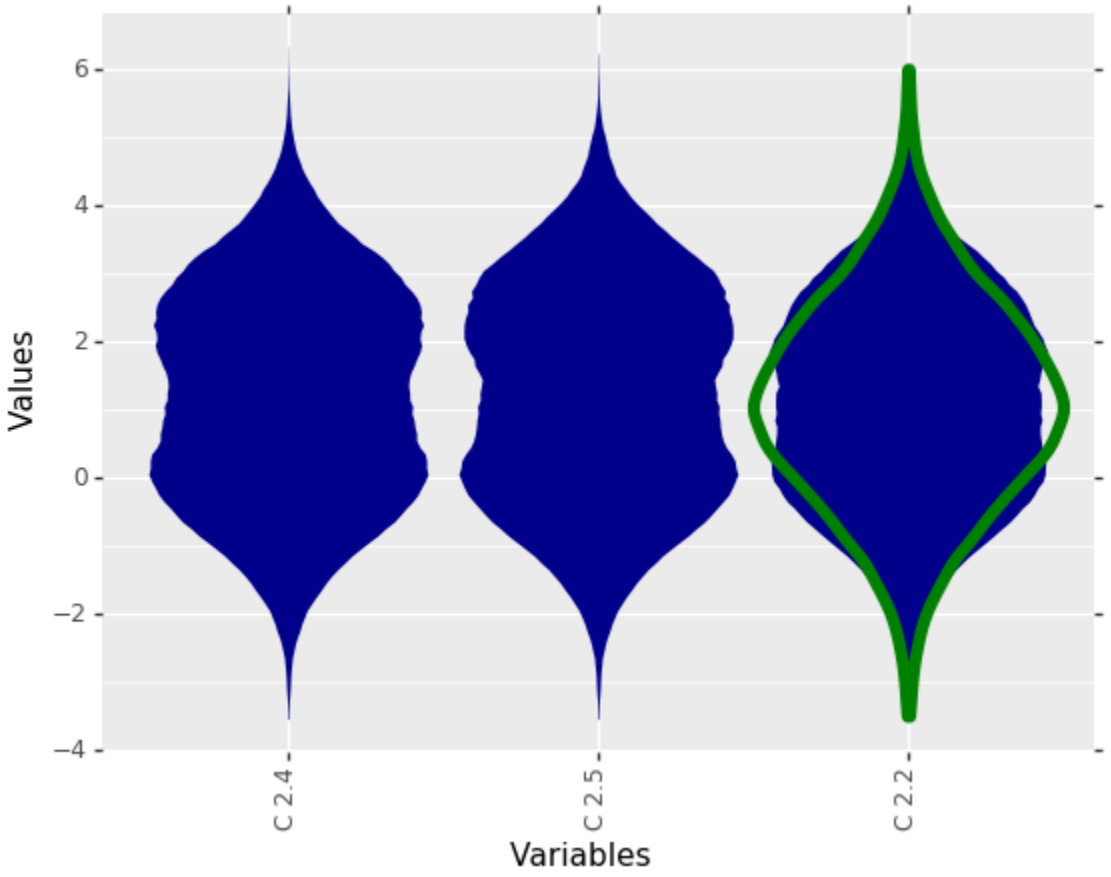


Fig. 5: *Visualization of bimodal data with a drawn Gaussian distribution.*

2.3.2 Draw a Box Plot

The parameters *BoxPlot* (deactivated by default) and *BoxColor* are used for plotting a boxplot over each MD-Plot.

```
MDplot(dctExamples['UniformSample'], BoxPlot = True)
```

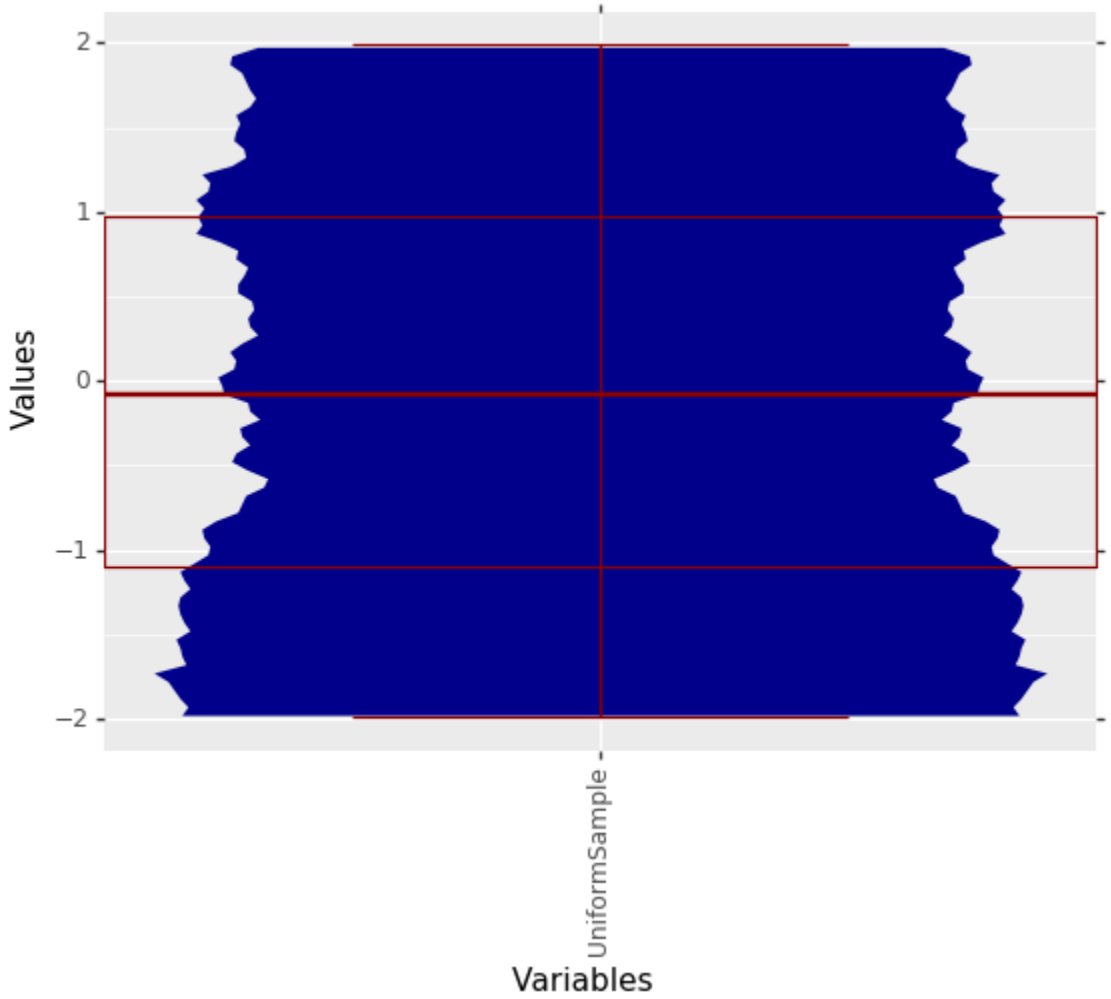


Fig. 6: *Uniformly distributed data with drawn Box Plot. The Box Plot alone would suggest a Gaussian distribution.*

2.3.3 Sampling

In order to avoid too long calculation durations, the MD-Plot deter-mines a uniformly distributed, random sample. This is controlled by the parameter *SampleSize* (default: 500000 elements / cells).

```
dfMTY = dctExamples["MTY_Clipped"]
dfMTY = dfMTY[(dfMTY["MTY_Clipped"] >= 1800) & (dfMTY["MTY_Clipped"] <= 6000)]
MDplot(dfMTY, SampleSize=5000)
```

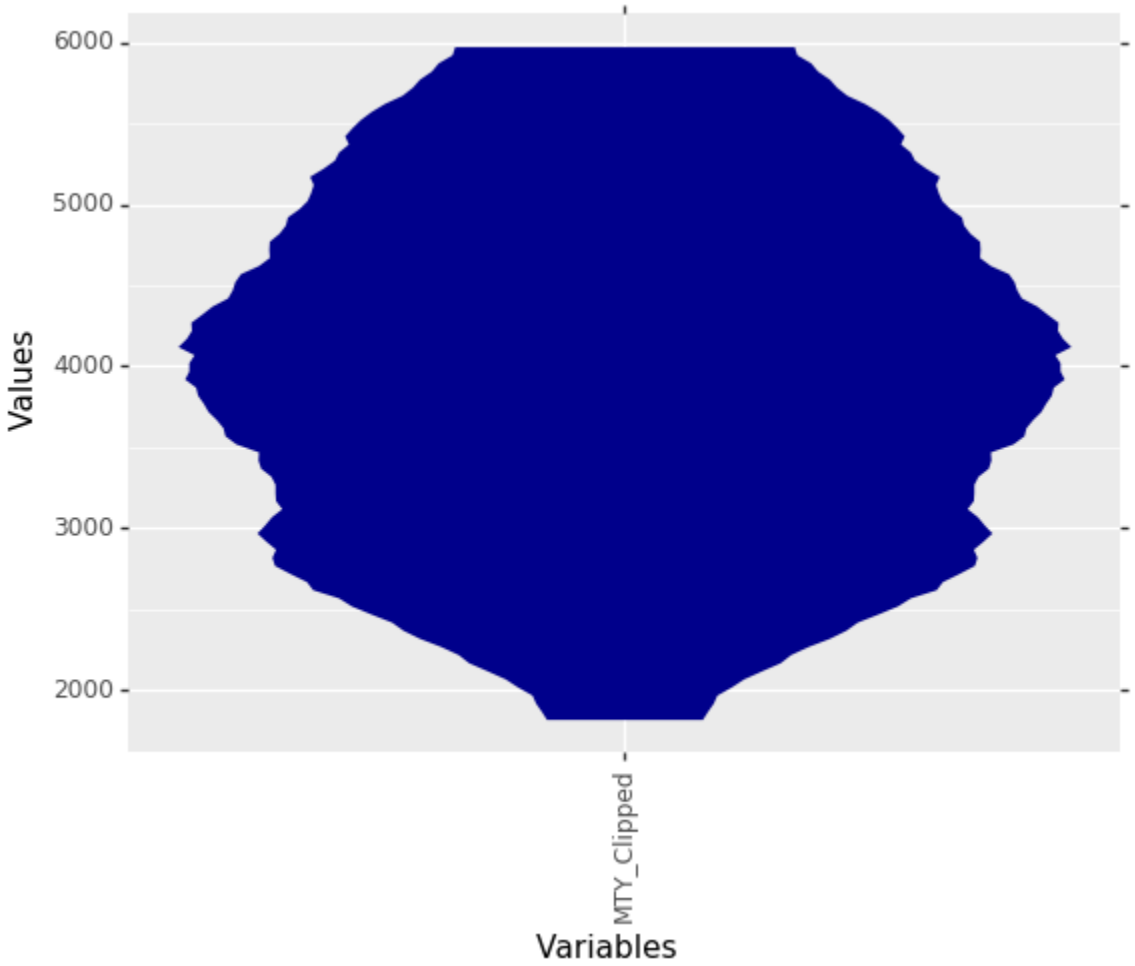


Fig. 7: *From 9467 to 5000 rows sampled MD-Plot of capped municipality income tax yield of Germany municipalities of 2015.*

2.3.4 Scaling

In order to visualize the shapes of all features with very different scales in a plot, the MD-Plot offers four different *Scaling* methods (Percentalize, CompleteRobust, Robust, Log).

```
MDplot(dctExamples['MuncipalIncomeTaxYield_IncomeTaxShare'])
```

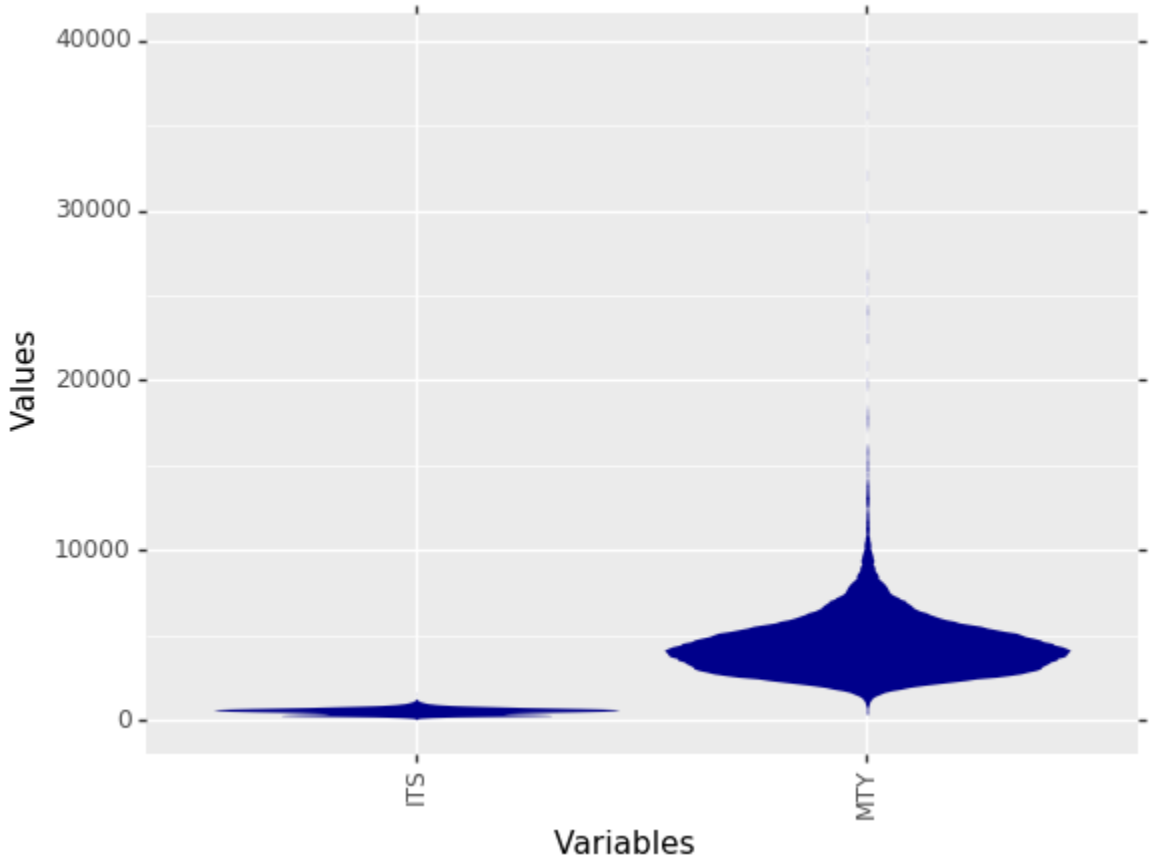


Fig. 8: *Visualization of two features with different value ranges. The comparison of the distributions is only possible to a limited extent.*

```
MDplot(dctExamples['MuncipalIncomeTaxYield_IncomeTaxShare'], Scaling='CompleteRobust')
```

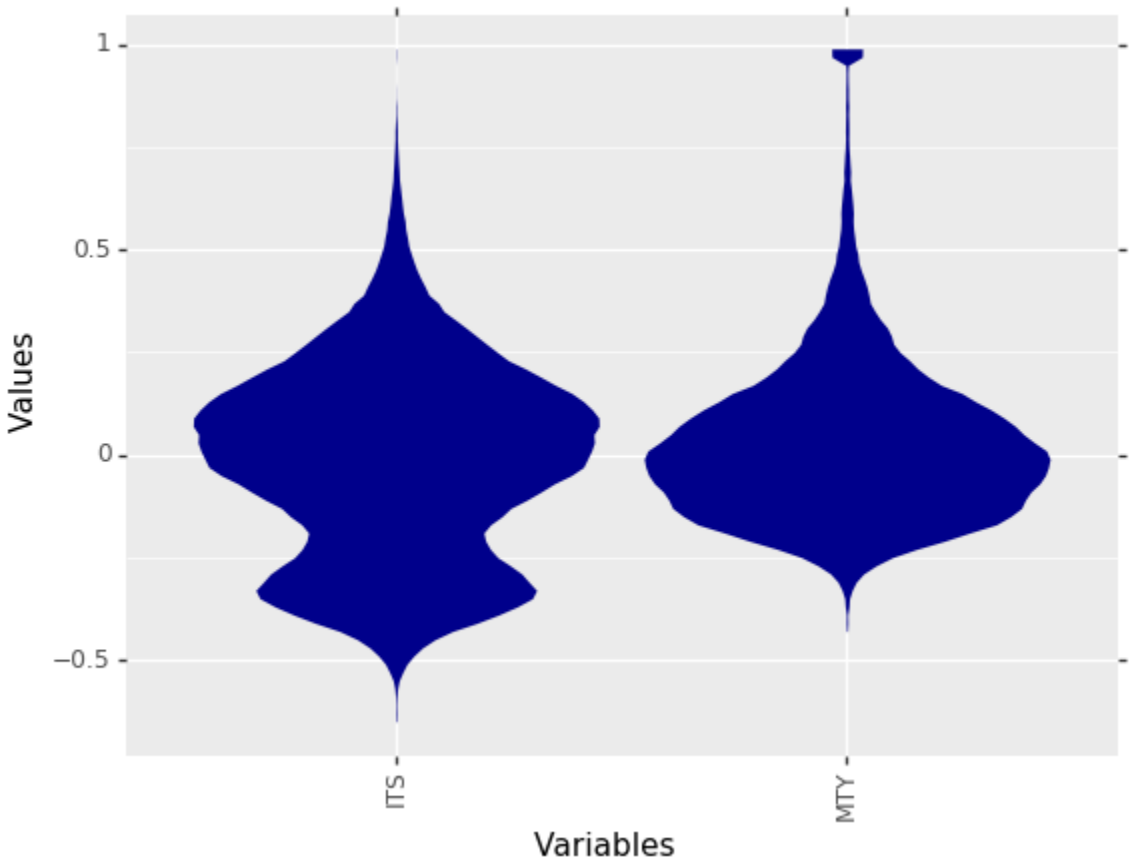


Fig. 9: *Scaling makes it easy to compare the distributions of features with different ranges of values.*

2.3.5 Ordering

The *Ordering* parameter controls the sequence of features displayed. For example, the ordering can be especially useful if one wants to sort the distribution gradually by skewness.

```
dfStocks = dctExamples["StocksData2018Q1"]
dfStocks = dfStocks[["TotalCashFlowFromOperatingActivities", "TreasuryStock",
↪"CapitalExpenditures", "InterestExpense", "Net-Income_y", "NetTangibleAssets",
↪"TotalAssets", "TotalLiabilities", "To-talStockholderEquity",
↪"TotalOperatingExpenses", "GrossProfit", "To-talRevenue"]]
dfStocks = dfStocks[(dfStocks >= -250000) & (dfStocks <= 1000000)]
MDplot(dfStocks, Ordering='Statistics')
```

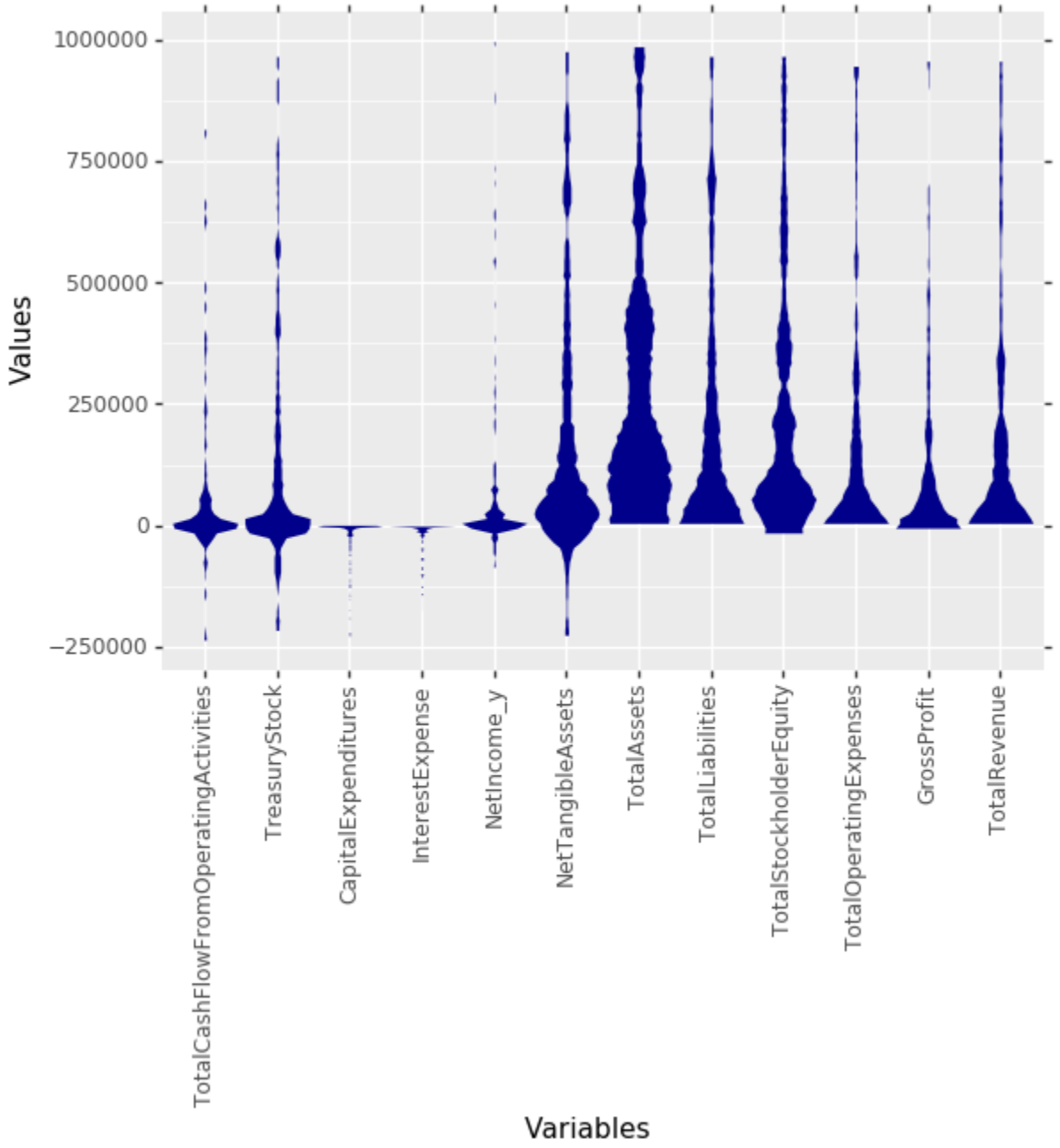


Fig. 10: *MD plots of selected features from 269 companies on the German stock market reporting quarter-ly financial statements by the Prime standard. The features are ordered by the effect strength of statistical tests about unimodality and skewness. This leads to an ordering from "Gaussian" fea-tures on the left to "Non-Gaussian" features on the right.*

```
MDplot(dfStocks, Ordering='Alphabetical')
```

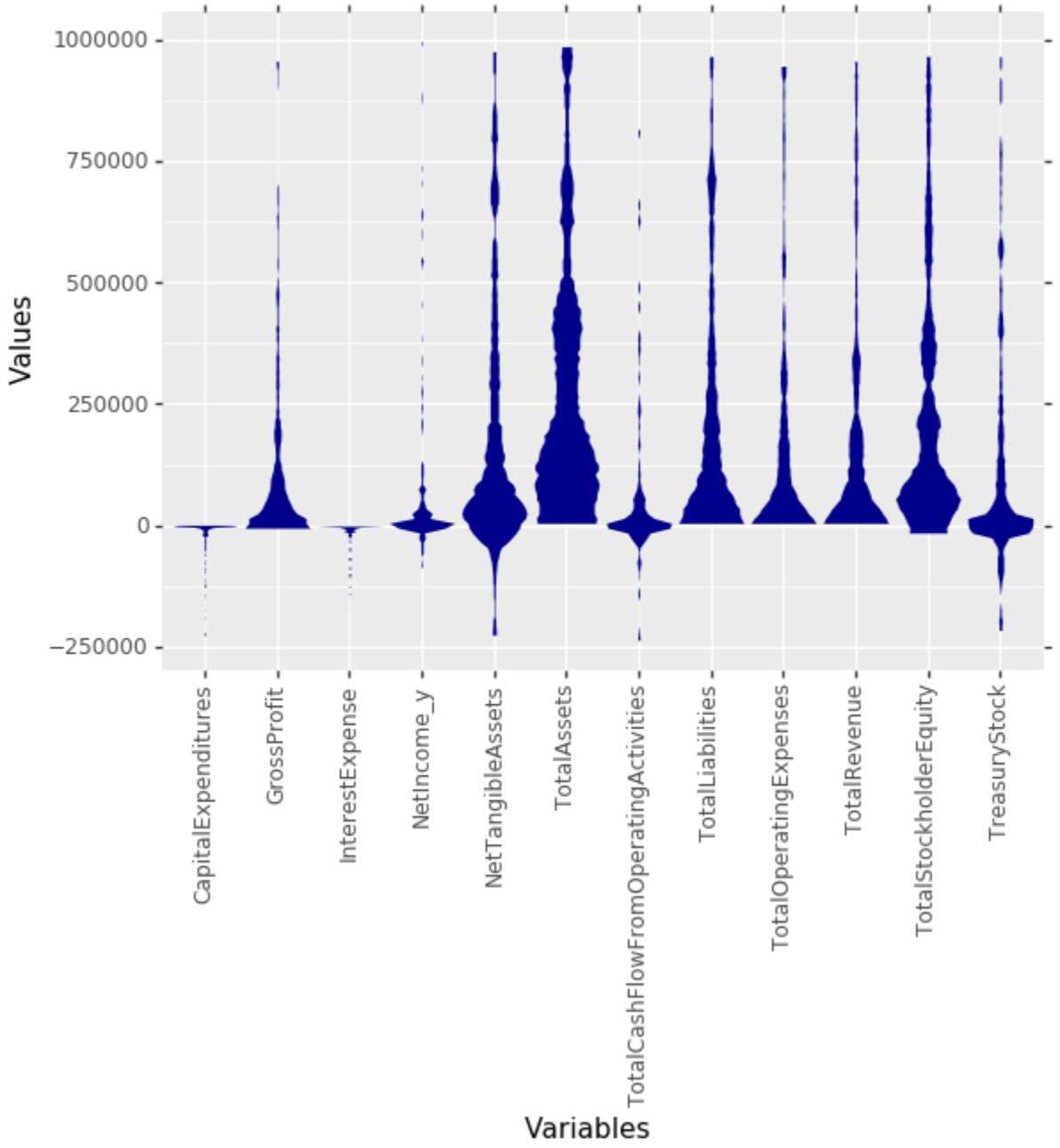


Fig. 11: *Same stock market features as in Fig. 10, but ordered alphabetical by their name.*

2.4 References

[Thrun/Ultsch, 2019] **Thrun, M. C., & Ultsch, A.:** Analyzing the Fine Structure of Distributions, Technical Report of the University of Marburg, 2019.

[Thrun/Ultsch, 2018] **Thrun, M. C., & Ultsch, A.:** https://cran.r-project.org/web/packages/DataVisualizations/index.html

CHAPTER 3

R Package

```
install.packages("DataVisualizations")

library(DataVisualizations)
```

The complete guide to DataVisualizations can be found in A Quick Tour in Data Visualizations

In this website only the Mirrored-Density plot (MD plot) is introduced. Please see for the introduction the R tutorial or the Python tutorial.

CHAPTER 4

R Tutorial

4.1 Introduction

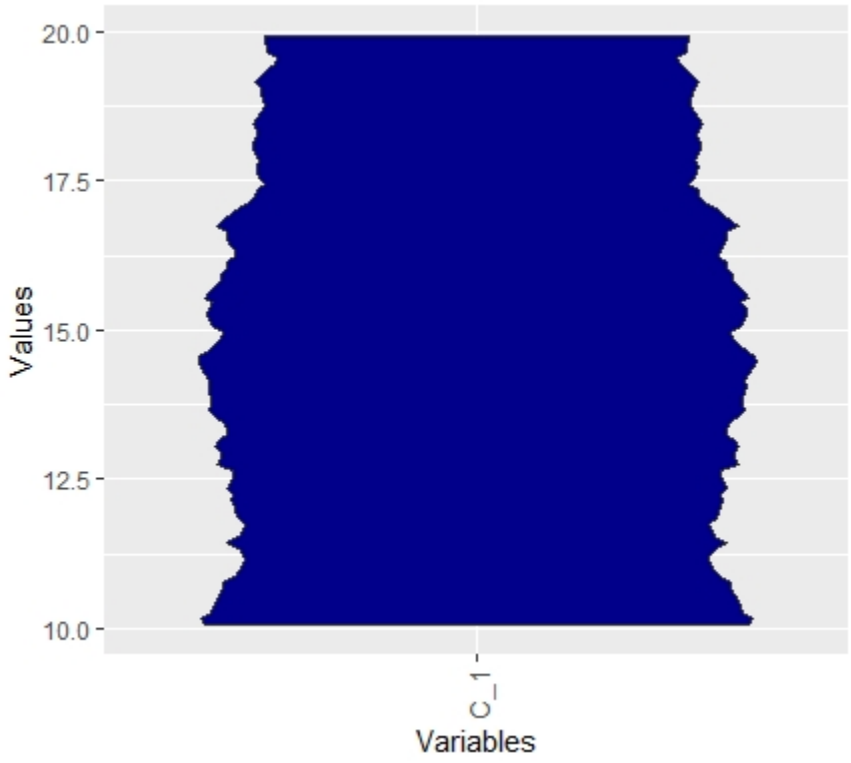
In data analysis of industrial topics such as the quality assurance of production facilities or the analysis of customer behavior, we constantly encounter univariate distributions of all kinds. From unimodal Gaussians to symmetric or skewed distributions to multimodal distributions It is of immense importance to determine the exact shape of the distribution in order to be able to select the correct further analysis steps and to be able to draw correct conclusions. Of special interest is the question, whether the given empirical distribution is composed of two or more distinct subsets of data points. Such subsets give hints to the existence of different states of the data producing process, such as, for example, healthy vs. sick patients or the existence of different diseases or treatments.

Conventional visualization methods of univariate probability density distributions have problems in the distinction of uniform versus multi-modal distributions and in visualizing capped skewed distributions correctly. With the mirrored density plot, a visualization method more suit-able for these applications was postulated [Thrun/Ultsch, 2019] and is currently under review in [Thrun et al., 2019].

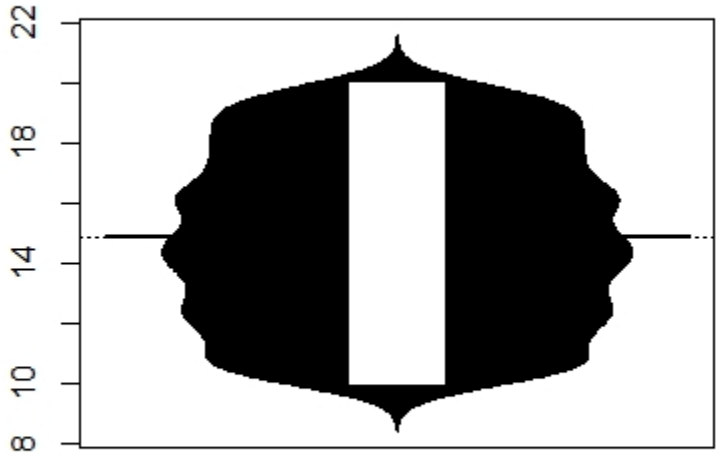
4.2 Basic Usage of Visualization

The Mirrored-Density plot (MD plot) is called by the function MDplot. The difference to the bean plot is visible: MD plot indicates a uniform distribution in a clearly capped range. The bean plot does not show the capped range and indicates multiomodality

```
library(DataVisualizations)
data=runif(1000,10,20)
MDplot(data)
```



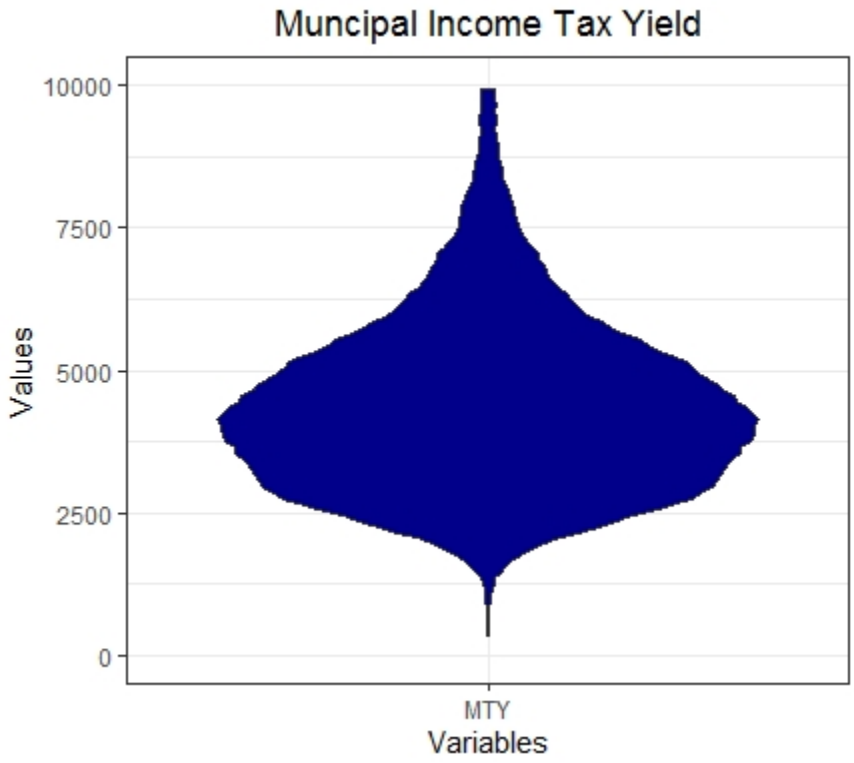

```
beanplot::beanplot(data)
```



4.3 Changing Layout

The MDplot uses the syntax of ggplot2. In this example we capped the values

```
library(DataVisualizations)
library(ggplot2)
data(MTY)
MDplot(MTY,Names = 'MTY')+ylim(c(0,10000))+ggtitle('Muncipal Income Tax Yield')+theme_
↪bw()+theme(plot.title = element_text(hjust = 0.5))
```



Of course, the ggplot2 internally also provides a similar visualization, but it does not estimate the probability density function (pdf) sufficiently. This is outlined in the next example.

4.4 Distribution Analysis of Stocks Data

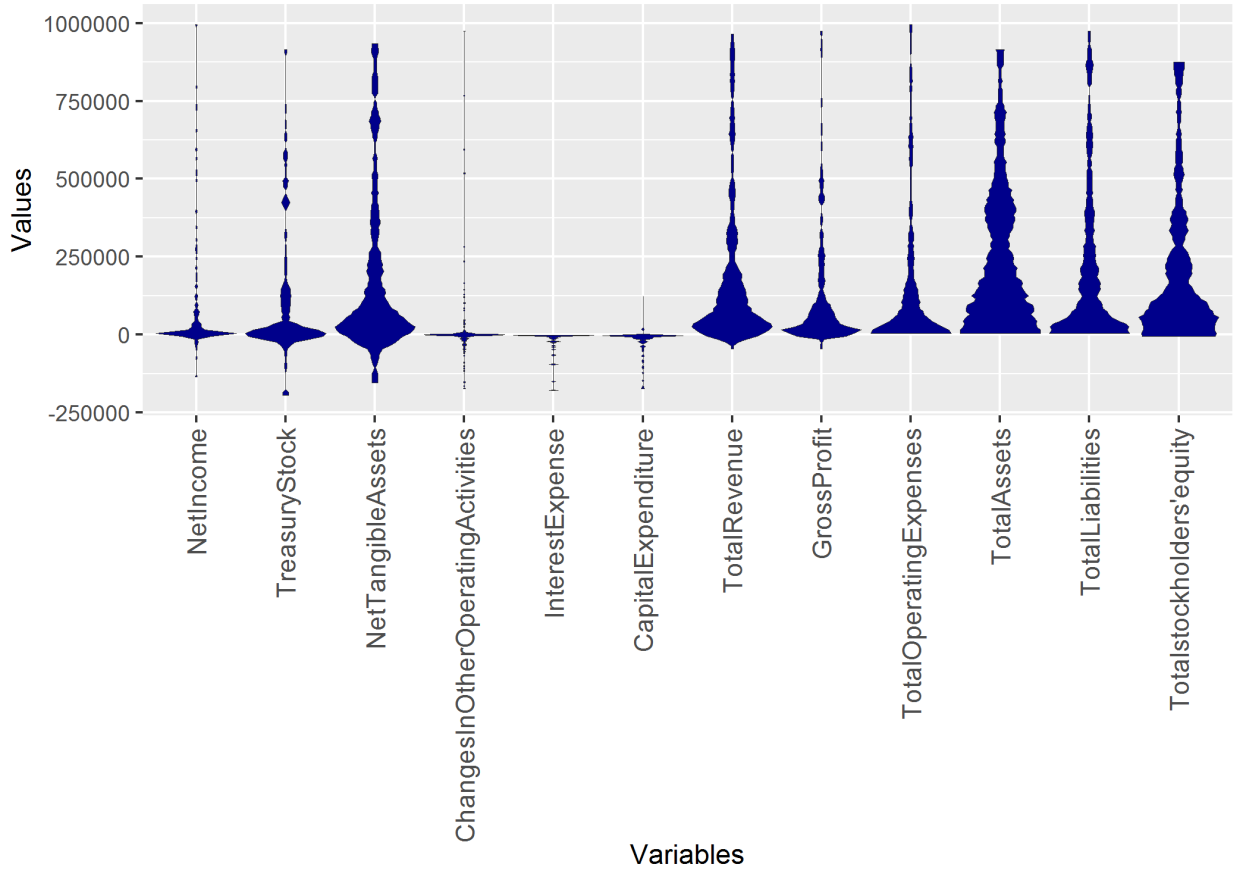
The data consists of Aaccounting information of 261 companies traded in the Frankfurt stock exchange in the German Prime standard. The data set is described in [Thrun et al., 2019]. Here, we use the third quarter instead the first, but the result remains the same because the same features are selected.

```
library(DataVisualizations)
library(ggplot2)
data('AccountingInformation_PrimeStandard_Q3_2019')
str(AI_PS_Q3_2019)
Data=AI_PS_Q3_2019$Data

targets=c('NetIncome','TreasuryStock','NetTangibleAssets',
          'ChangesInOtherOperatingActivities',
```

(continues on next page)

(continued from previous page)

```
          'InterestExpense',
          'CapitalExpenditure','TotalRevenue',
          'GrossProfit','TotalOperatingExpenses',
          'TotalAssets','TotalLiabilities',
          "Totalstockholders'equity")
ind=match(table = colnames(Data),targets)

MDplot(Data[,ind],Ordering = 'Columnwise')+ylim(-200000,1000000)
ggsave(filename='MDplot_stocksdata.png')
```



4.5 Comparison to Violin plot of ggplot2

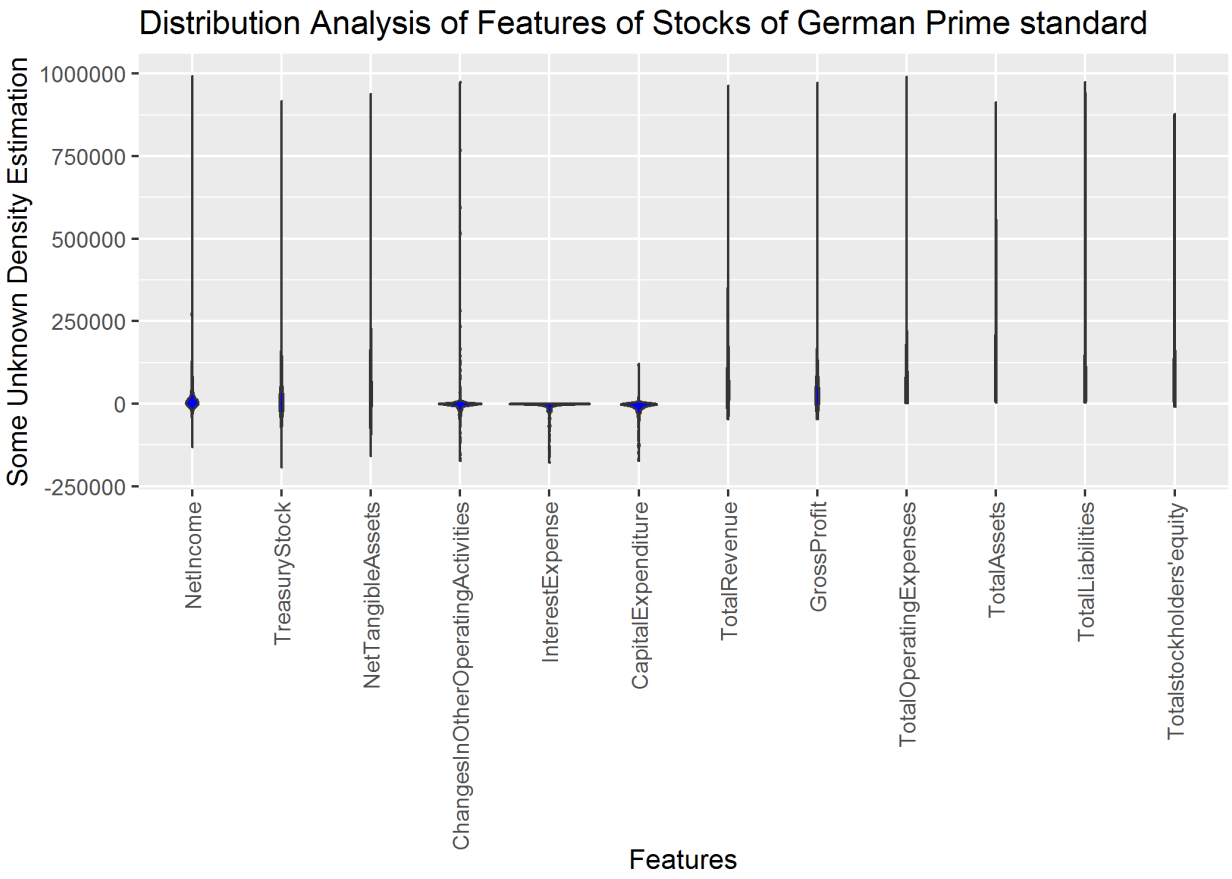
Of course the usual violin plot of ggplot2 does not have all the features of the MD plot. Additionally, we did not compare with the ggplot in our publication ([Thrun et al., 2019]) because we were unable to find out how the density is estimated in ggplot2. The main point here is, that the usual density estimation is sometimes incorrect:

```
library(ggplot2)
dataframe = reshape2::melt(Data[,ind])
colnames(dataframe) <- c('ID', 'Variables', 'Values')
plot =ggplot(data = dataframe,
             aes_string(x = "Variables", group = "Variables", y = "Values"))+ylim(-
↪200000,1000000)
plot=plot + geom_violin(fill="blue")
```

(continues on next page)

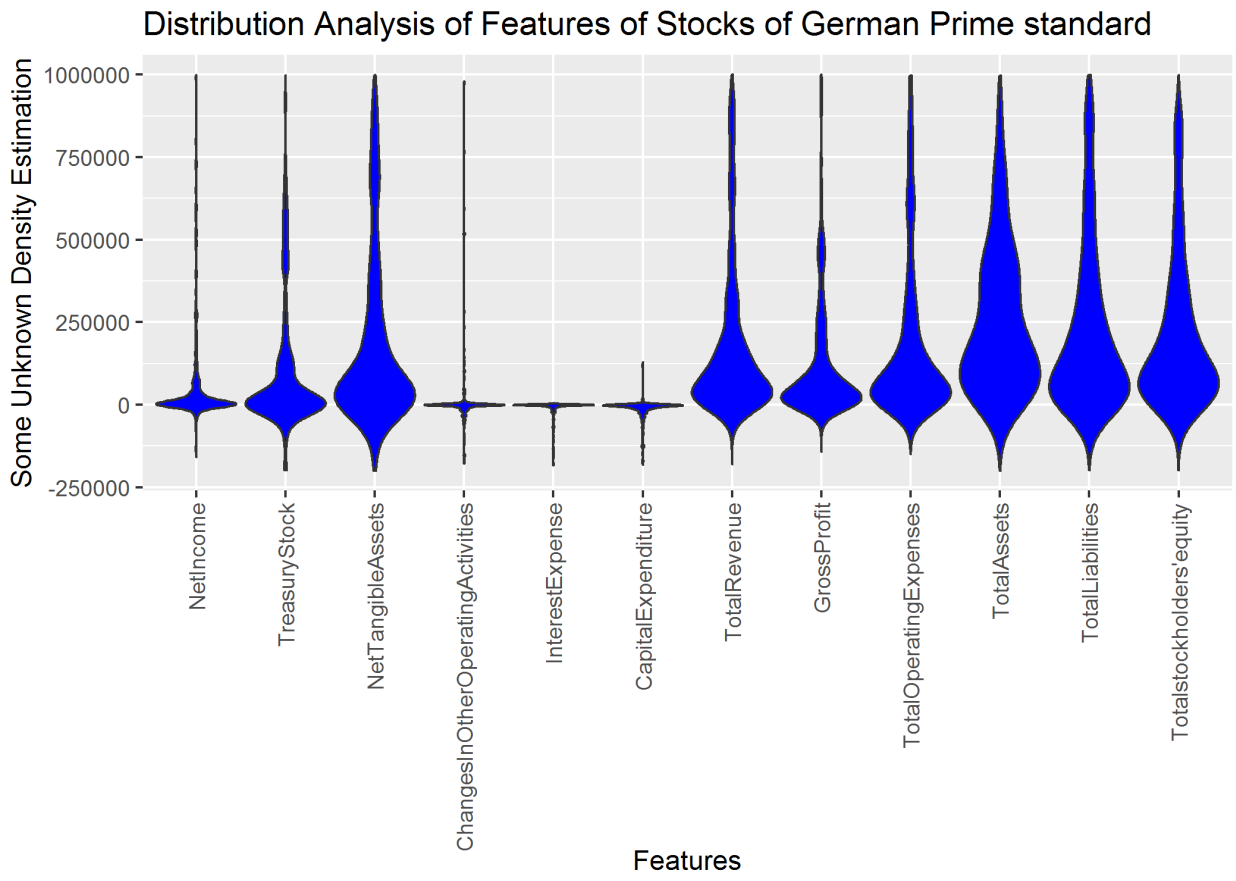
(continued from previous page)

```
plot+ggtitle('Distribution Analysis of Features of Stocks of German Prime standard
↪')+xlab('Feautres')+ylab('Some Unknown Density Estimation')+ggExtra::rotateTextX()
#ggsave(filename='ggplot1_stocksdata.png')
```



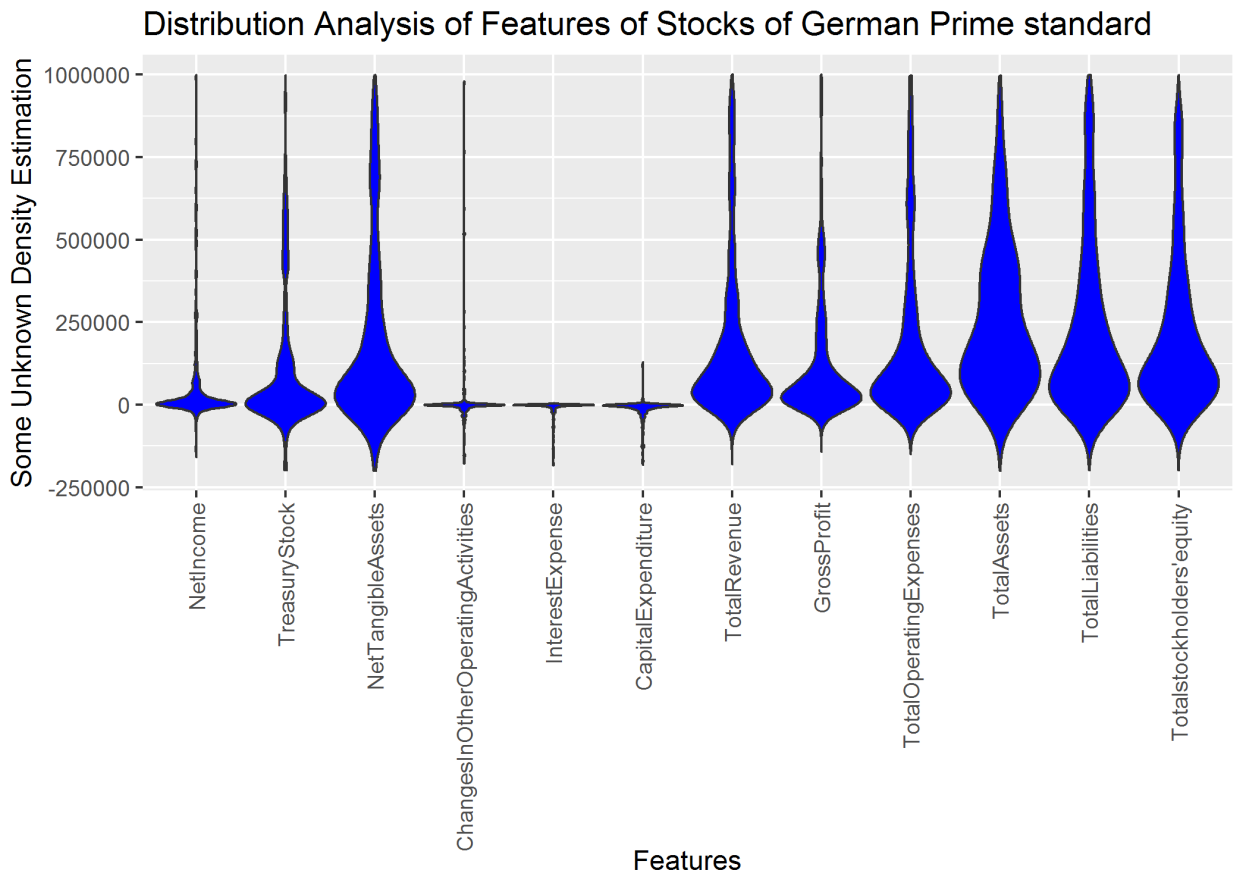
Setting the parameters manually can result in an incorrect estimation of the probability density function (pdf).

```
plot =ggplot(data = dataframe,
             aes_string(x = "Variables", group = "Variables", y = "Values"))+ylim(-
↪200000,1000000)
plot=plot + geom_violin(fill="blue",scale = "width",trim=FALSE)
plot+ggtitle('Distribution Analysis of Features of Stocks of German Prime standard
↪')+xlab('Feautres')+ylab('Some Unknown Density Estimation')+ggExtra::rotateTextX()
#ggsave(filename='ggplot2_stocksdata.png')
```



Correct parameter setting still results in an inferior visualization because multimodality is not visible:

```
plot =ggplot(data = dataframe,
             aes_string(x = "Variables", group = "Variables", y = "Values"))+ylim(-
↪200000,1000000)
plot=plot + geom_violin(fill="blue",scale = "width",trim=TRUE)
plot+ggtitle('Distribution Analysis of Features of Stocks of German Prime standard
↪')+xlab('Feautres')+ylab('Some Unknown Density Estimation')+ggExtra::rotateTextX()
#ggsave(filename='ggplot3_stocksdata.png')
```



In sum, the density estimation of ggplot2 requires the setting of parameters which is in an exploratice approach not feasible. MD plot does not require the setting of paramters for the estimation of the pdf. However, the user can use additional features of MD plot like automatic transformation of variables or distinguishing gaussian from non-gaussian distributions.

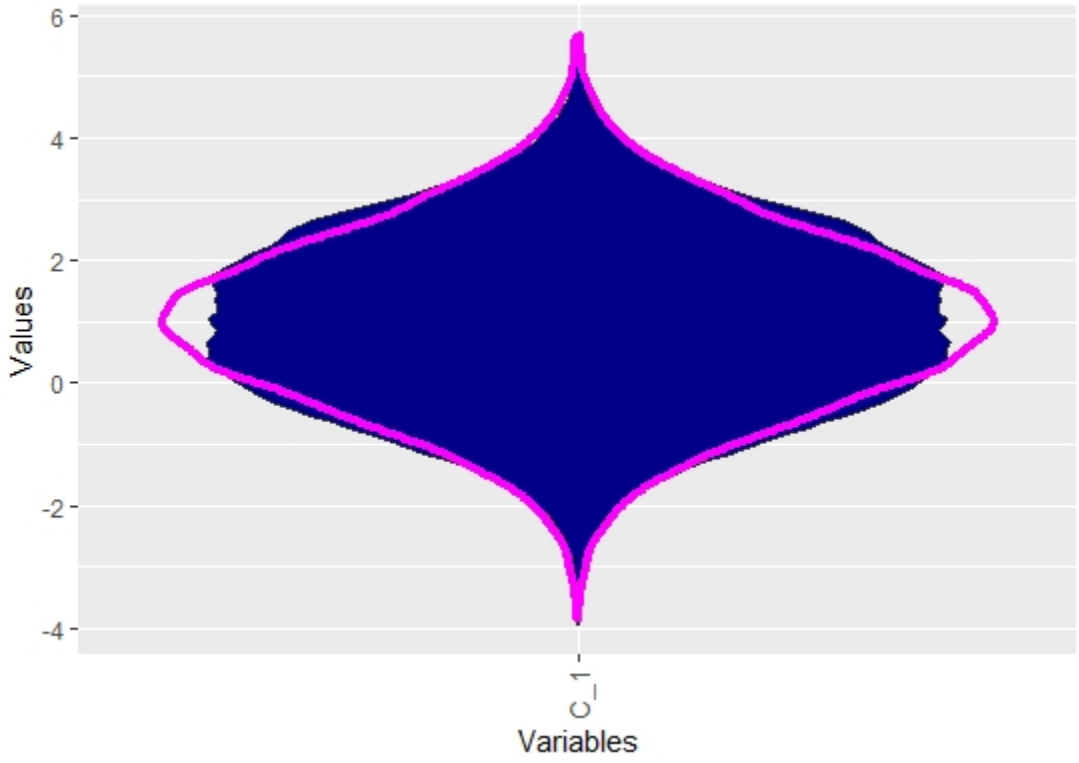
4.6 Advanced Usage

Contrary to geom_violin of ggplot2 we offer various additional functionality.

4.6.1 Overlay with Robustly Estimated Gaussian Distribution

Using the default option RobustGaussian = T we can see that the distribution ist not Gaussian. Robust Gaussian are only estimated and overlayed in the MD plot if statistical testing does not reject the hypothesis that the data is not multimodal. In a sense, this visualization is more sensitive than Hartigan's dip test for multimodality.

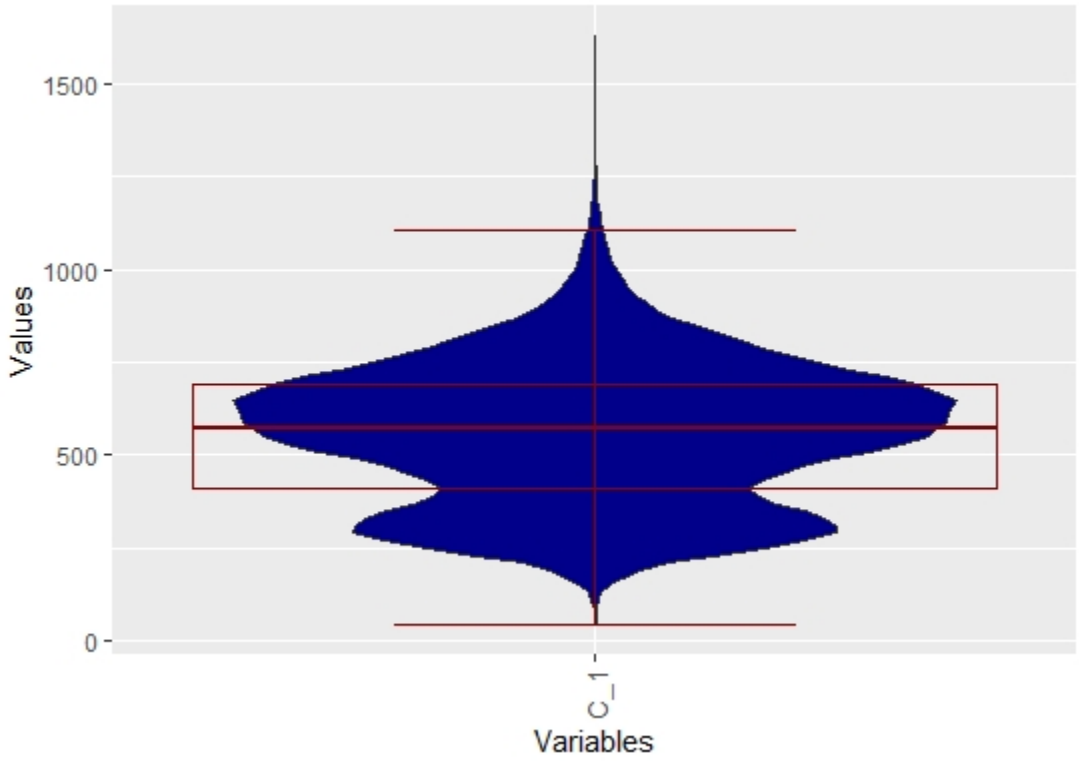
```
library(DataVisualizations)
g1=rnorm(n = 10000,mean = 0,sd = 1)
g2=rnorm(n = 10000,mean = 2,sd = 1)

Data=c(g1,g2)
diptest::dip.test(Data)

MDplot(Data,RobustGaussian = T)
```



4.6.2 Overlay with Box Plot

Box plots are unable to visualize multimodality. However, the Dplot can be overlayed by them.

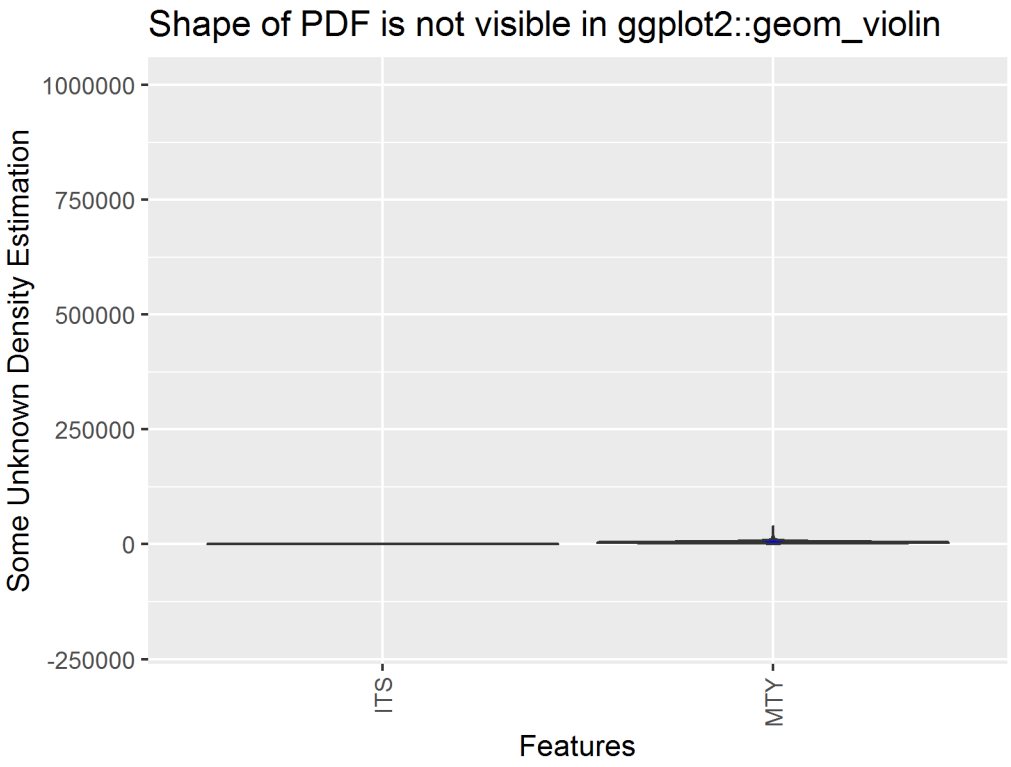
```
library(DataVisualizations)
data(ITS)
MDplot(ITS,BoxPlot = T)
```

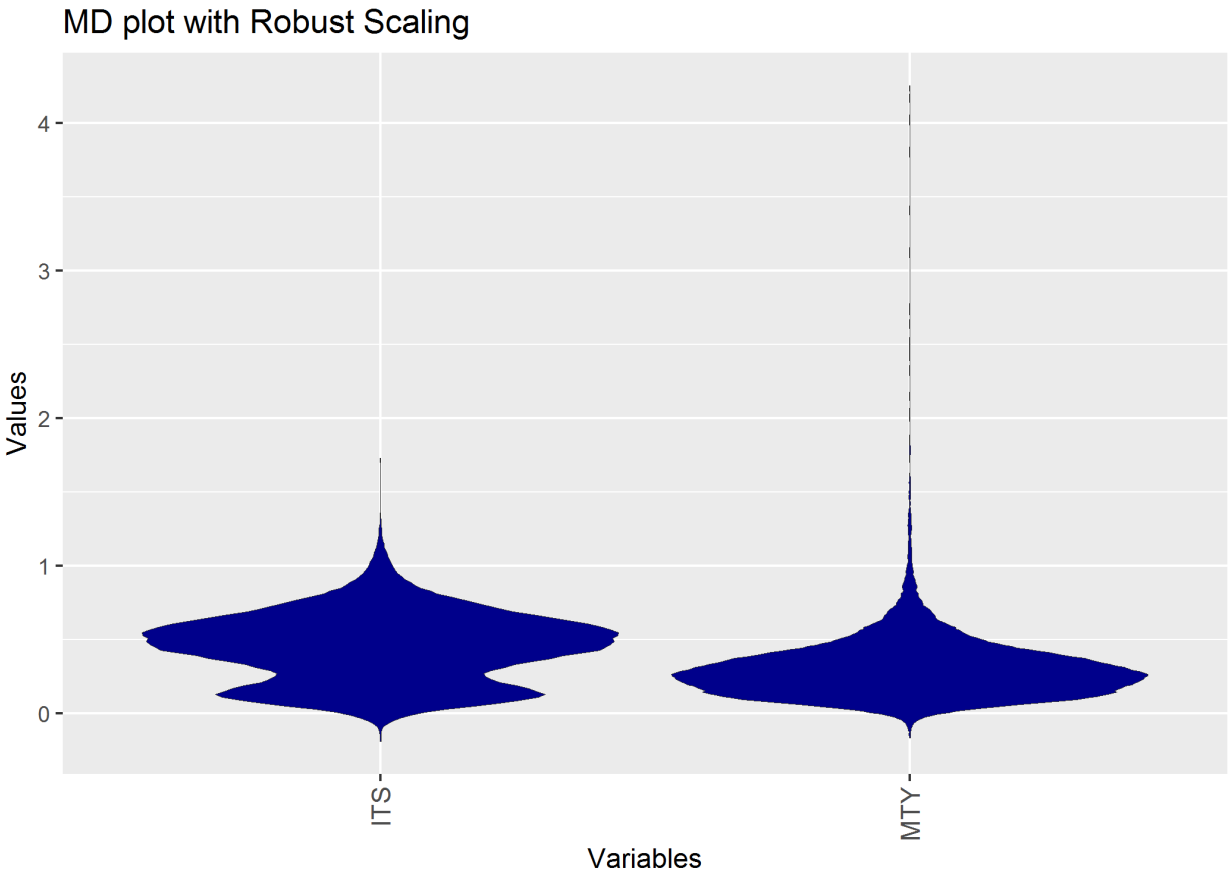


4.6.3 Scaling

In bean plots or violin plots the visualizations of more than one features fails often in praxis because the scaling is not the same. In the MDplot there exists an option for transformining the data without chaning the shape of the pdf

```
library(DataVisualizations)
library(ggplot2)
data(ITS)
data(MTY)
Data=cbind(ITS,MTY)

dataframe = reshape2::melt(Data)
colnames(dataframe) <- c('ID', 'Variables', 'Values')
plot =ggplot(data = dataframe,
             aes_string(x = "Variables", group = "Variables", y = "Values"))+ylim(-
↪200000,1000000)
plot=plot + geom_violin(fill="blue",scale = "width",trim=TRUE)
plot+ggtitle('Shape of PDF is not visible in ggplot2::geom_violin')+xlab('Features
↪')+ylab('Some Unknown Density Estimation')+ggExtra::rotateTextX()

ggsave(filename='ggplot_fails.png')

MDplot(Data,Scaling = 'Robust')+ggtitle('MD plot with Robust Scaling')
ggsave(filename='RobustMDplot.png')
```

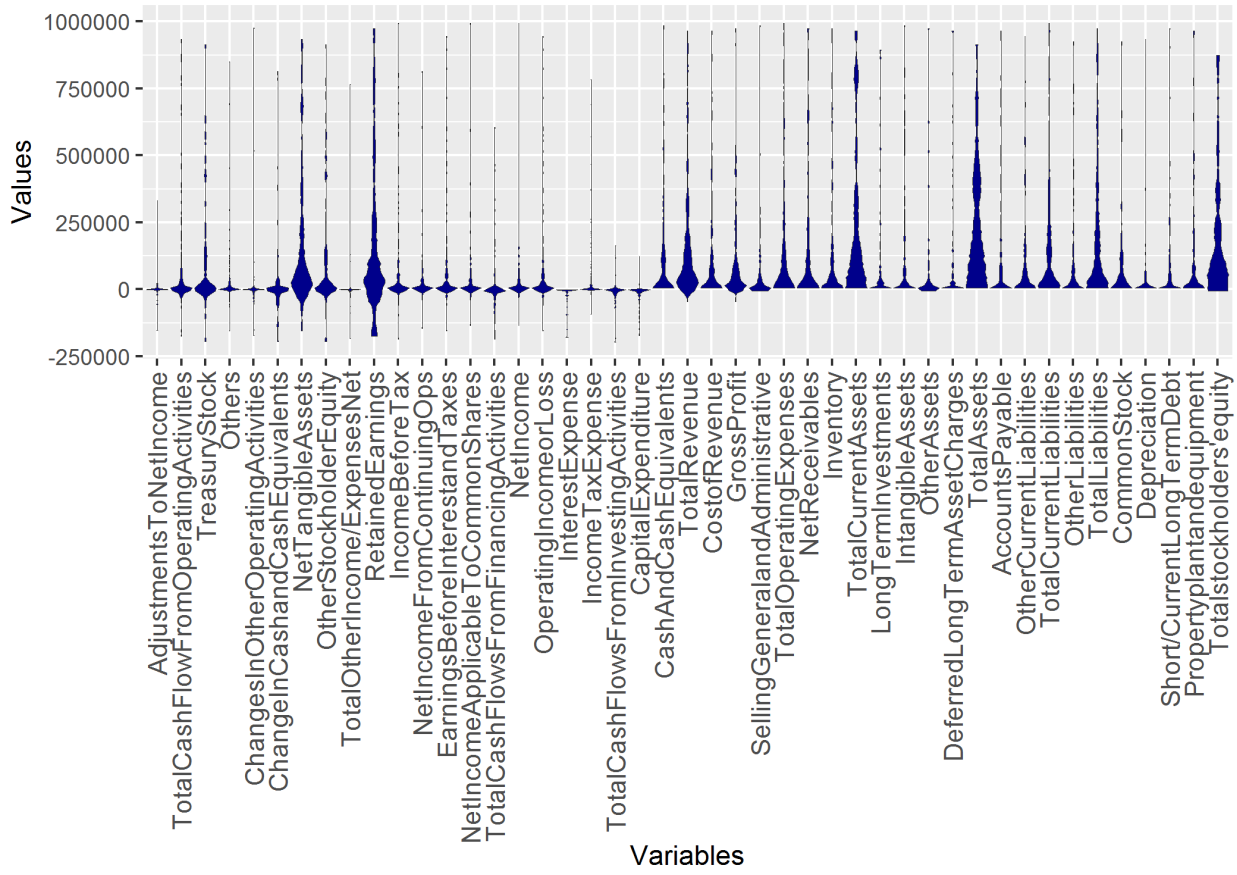

Shape of PDF is not visible in ggplot2::geom_violin
Some Unknown Density Estimation
1000000
750000
500000
250000
0
-250000
ITS
MTY
Features


MD plot with Robust Scaling
Values
4
3
2
1
0
ITS
MTY
Variables

4.6.4 Ordering

Data can ordered in the plots in varous ways. The resulting ordering or event transformations can be used in subsequent analysis:

```
library(DataVisualizations)
library(ggplot2)
data('AccountingInformation_PrimeStandard_Q3_2019')
Data=AI_PS_Q3_2019$Data

List=MDplot(Data,Ordering = 'Default',OnlyPlotOutput = FALSE)
DataOrdered=List$DataOrdered
str(DataOrdered)

List$ggplotObj+ylim(-200000,1000000)
ggsave(filename='MDplot_stocksdata_ordered.png')
```



[Thrun/Ultsch, 2019] Thrun, M. C., & Ultsch, A.: Analyzing the Fine Structure of Distributions, Technical Report of the University of Marburg, 2019.

[Thrun et al., 2019] Thrun, M. C., Gehlert, T., & Ultsch, A.: Analyzing the Fine Structure of Distributions, PloS one, under review, preprint available at arXiv:1908.06081, 2019.

CHAPTER 5

Example Applications

- Distribution Analysis of Flow Cytometry Big Data for the Detection of Lymphoma
- Investigation of High-Dimensional Accounting Information with the Goal to Explain Clusters

5.1 Content of Flow Cytometry Data

Multiparameter flow cytometry is an immunologically based standard diagnostic procedure in oncology. It requires special medical knowledge in analysis and interpretation and a very extensive data aquisition. The knowledge of performing a manual analysis must be acquired in addition to the clinical background over several years of training. The method is of great clinical value for the care of oncological patients due to its simple application and high measuring speed compared to molecular genetic methods. We will show here, that there are infact obvious differences between ill and healthy patients if the MD plot is used.

5.2 Data of Flow Cytometry

The procedures of reading data can be found in https://rdrr.io/github/aultsch/DataIO/. Unfortunatly, the dataset and prior classification cannot be provided as long as this is an going research project. Additonally, we use a random sample. In the next lines we load the data, check that the key of the cases is the same between prior classification and data matrix and show the dimensionality of the data set.

```
library(dbt.DataIO)
Path=ReDi("MirroredDensityPlot2019/09Originale",'F')
setwd(Path)
V=ReadLRN('01HealthyandCLLRandomSamplesC20n328000')
Data=V$Data
Key=V$Key
V=ReadCLS('02HealthyandCLLRandomSamplesN328000')
ClsKey=V$ClsKey
Cls=V$Cls
#Is Key the same?
```

(continues on next page)

(continued from previous page)

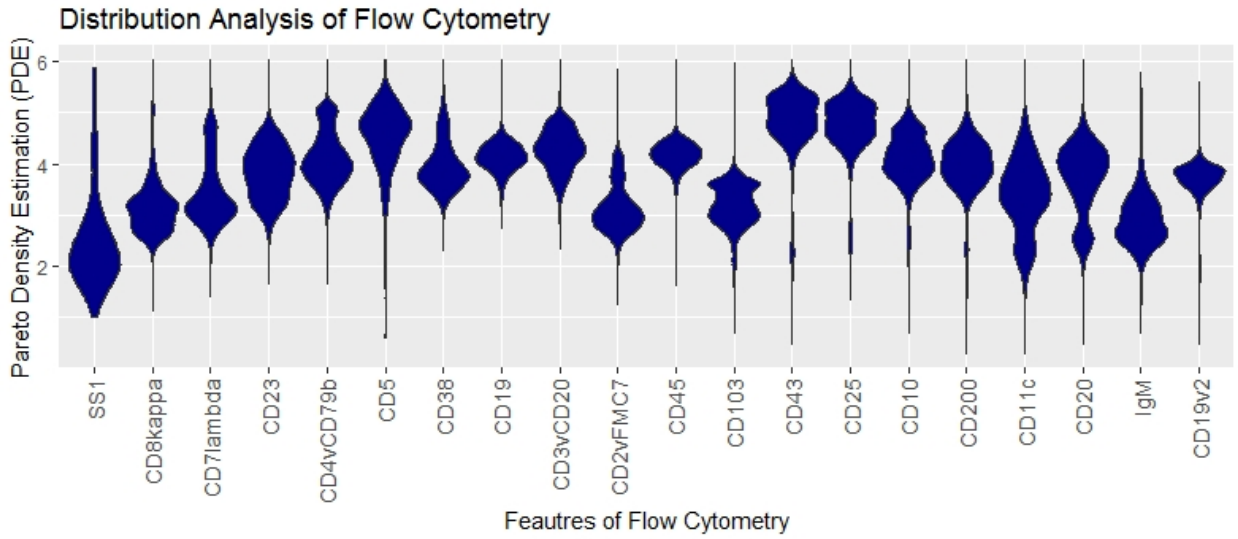
```
dbt.ClassAnalysis::TheSameKey(ClsKey,Key)
#Big Data
dim(Data)
```

5.3 The Mirrored-Density plot (MD plot) Estimates Every Probability Density Function

The Mirrored-Density plot (MD plot) can be used with an prior classification if one is interested in the differences between classes. The differences are here interesting, because they can be exploited for detection purposes.

The MDplot uses the syntax of ggplot2.

```
library(DataVisualizations)
MDplot(Data,Ordering = 'Columnwise')+ggtitle('Distribution Analysis of Flow Cytometry
↪')+xlab('Feautres of Flow Cytometry')+ylab('Pareto Density Estimation (PDE)')
```



No feature is overlayed with with a roubustly estimated unimodal Gaussian distribution in magenta, meaning that statistical testing showed a significant difference.

Multimodality (more than one hill) is visible in the features: CD20,CD7lambda,CD4vCD79b, CD2vFMC7, CD103, CD43, CD25, CD10, CD11c

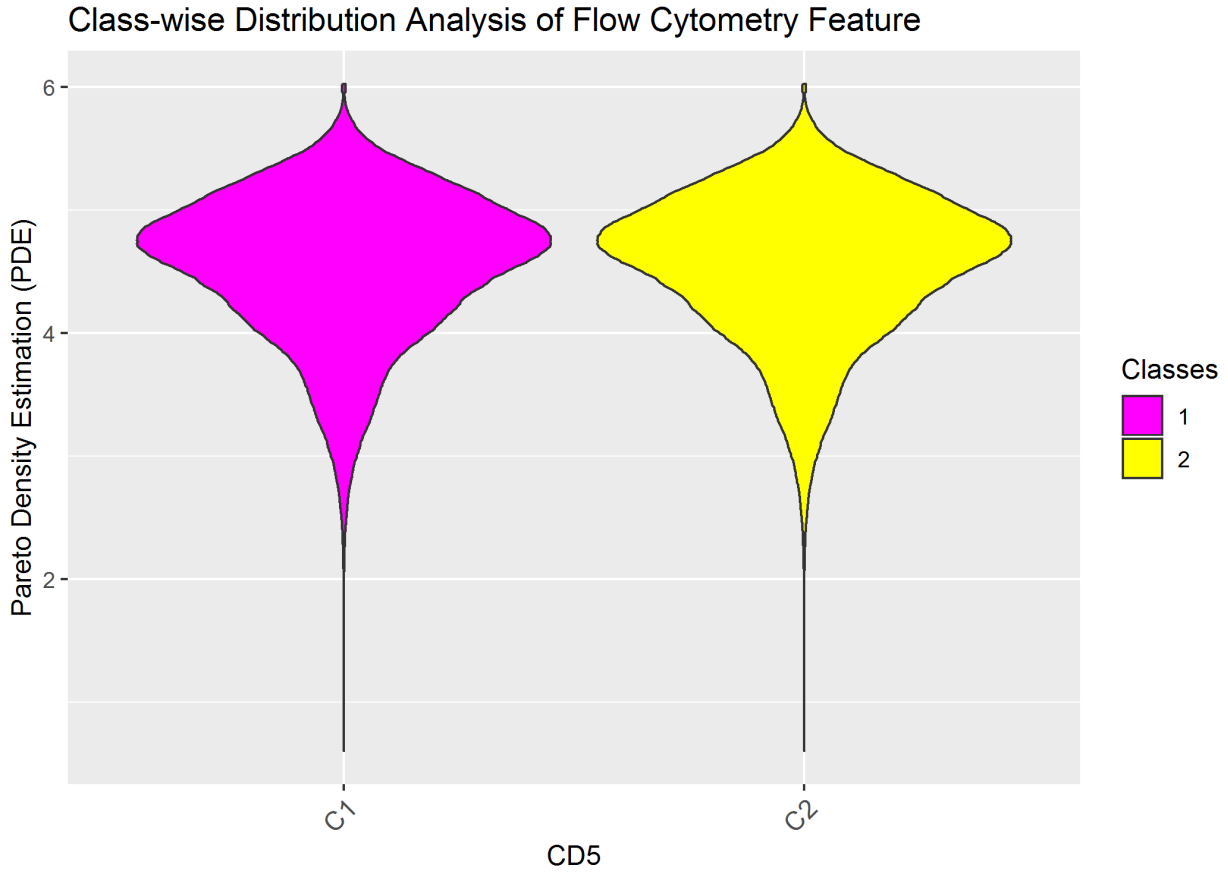
Skewness is visible in the features: SS1,CD8kappa, CD23, CD7lambda, CD5, CD38, CD11c, igM

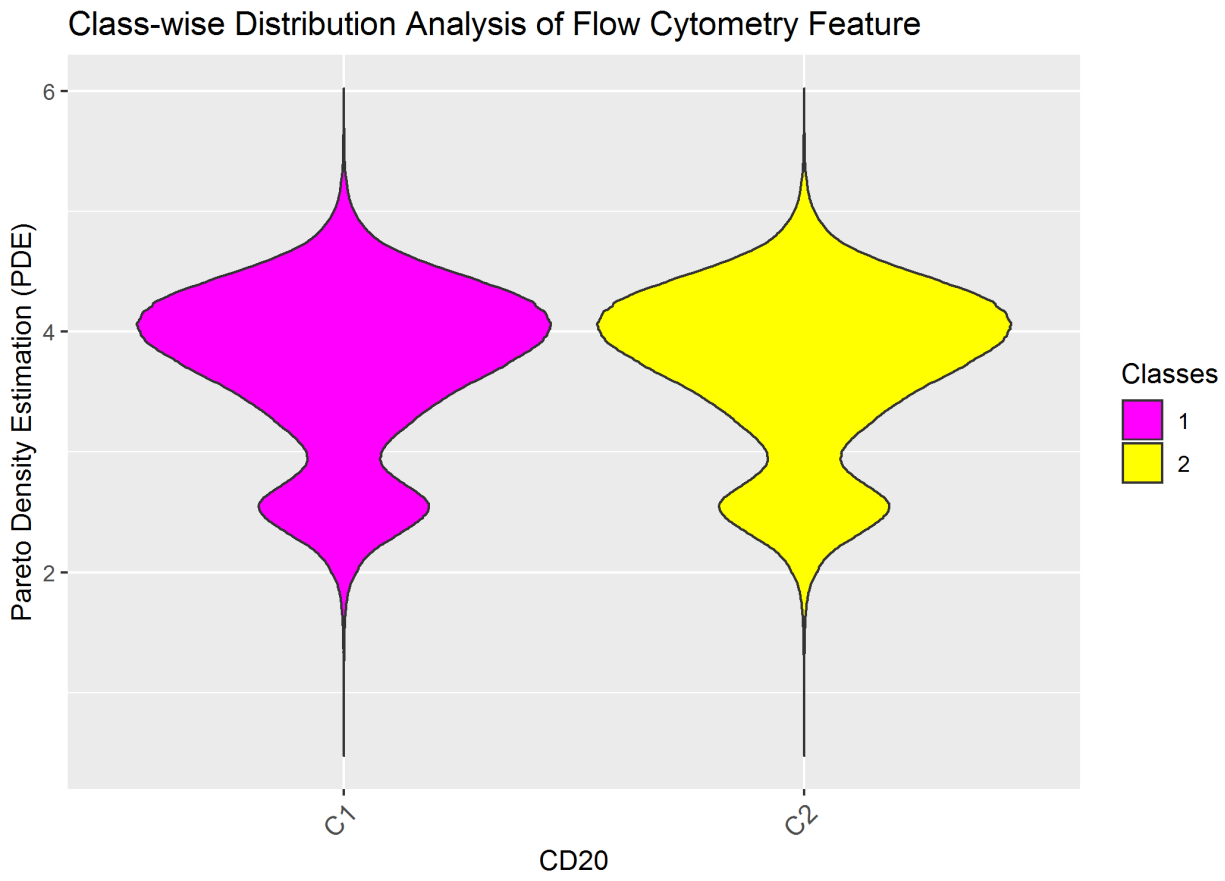
No Data clipping is visible.

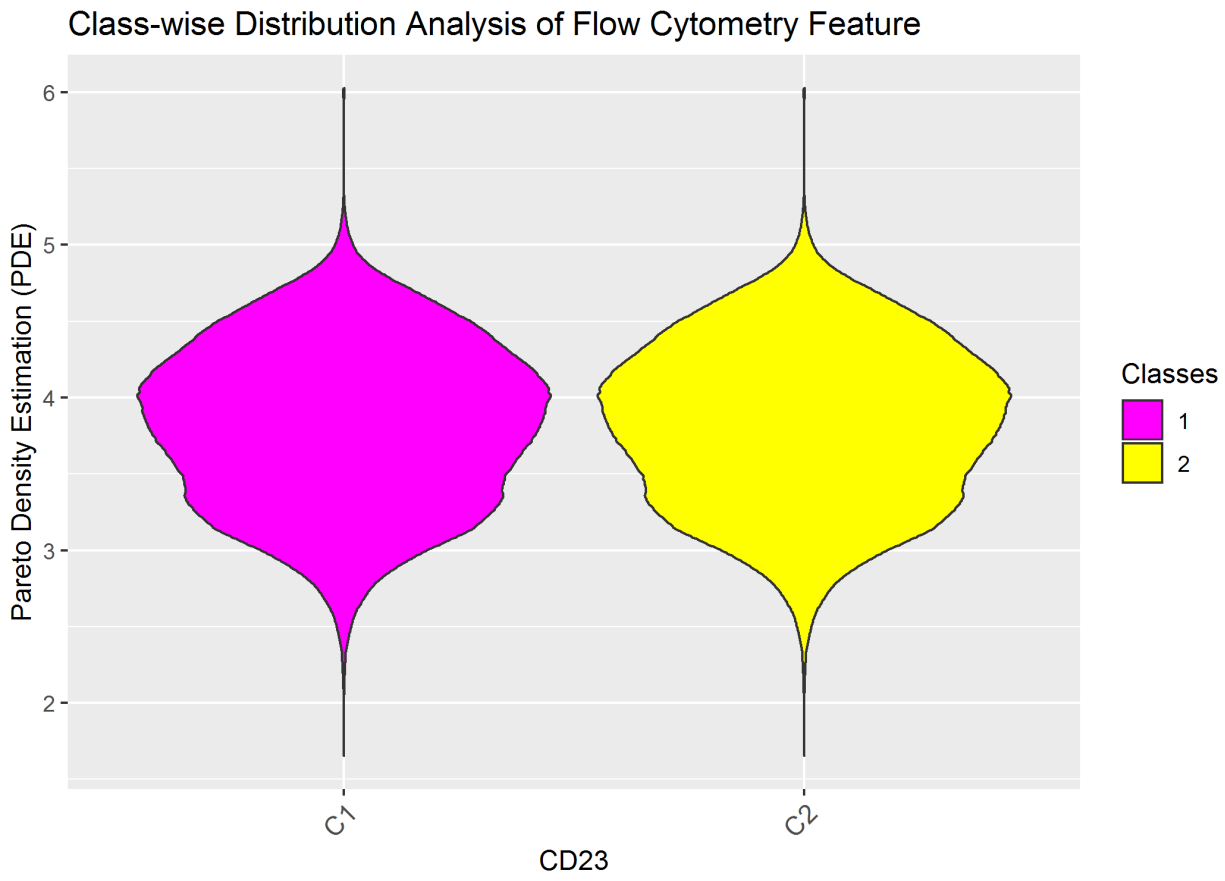
5.4 Class-wise MD plot for Flow Cytometry Application

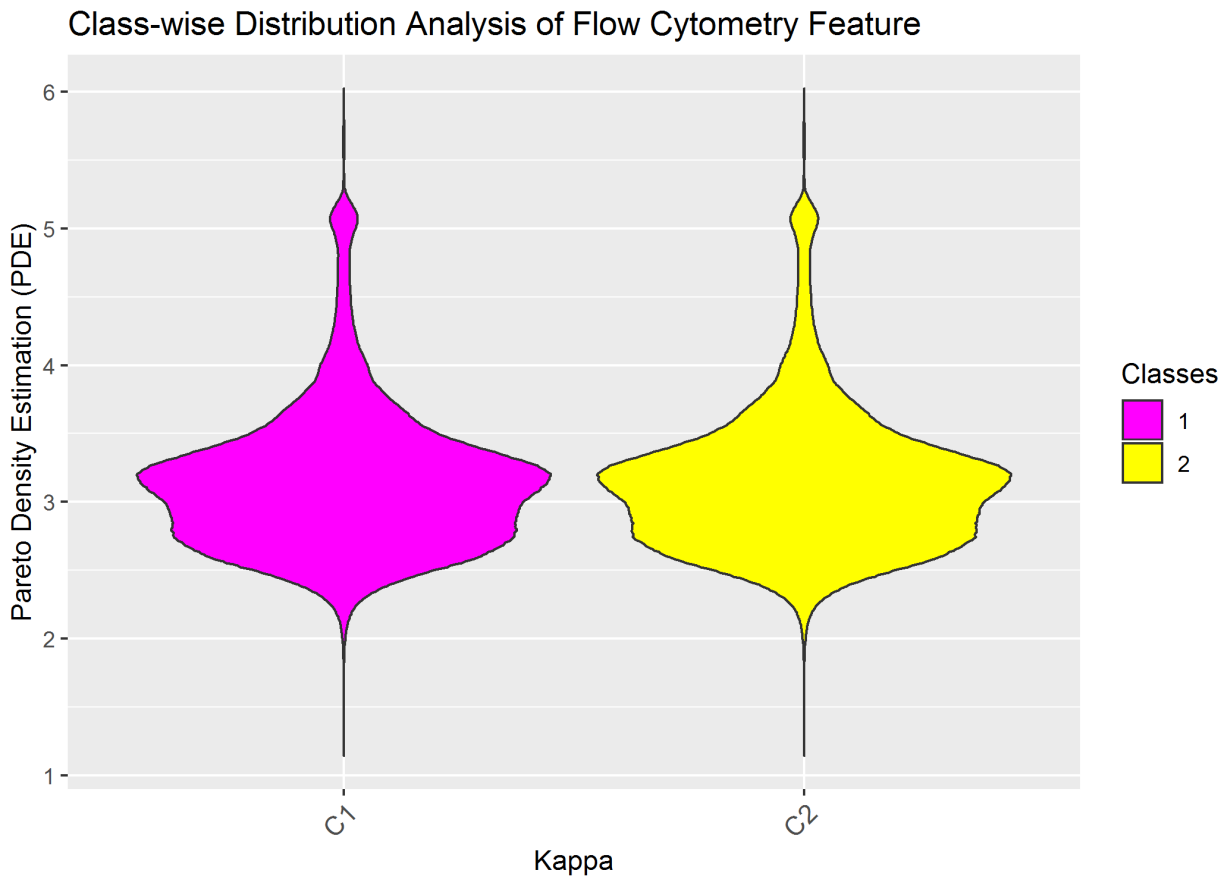
An follow up question could be if we are able to distinguish ill from healthy patients by features, if a prior classification is given. We select features which are used in current panels.

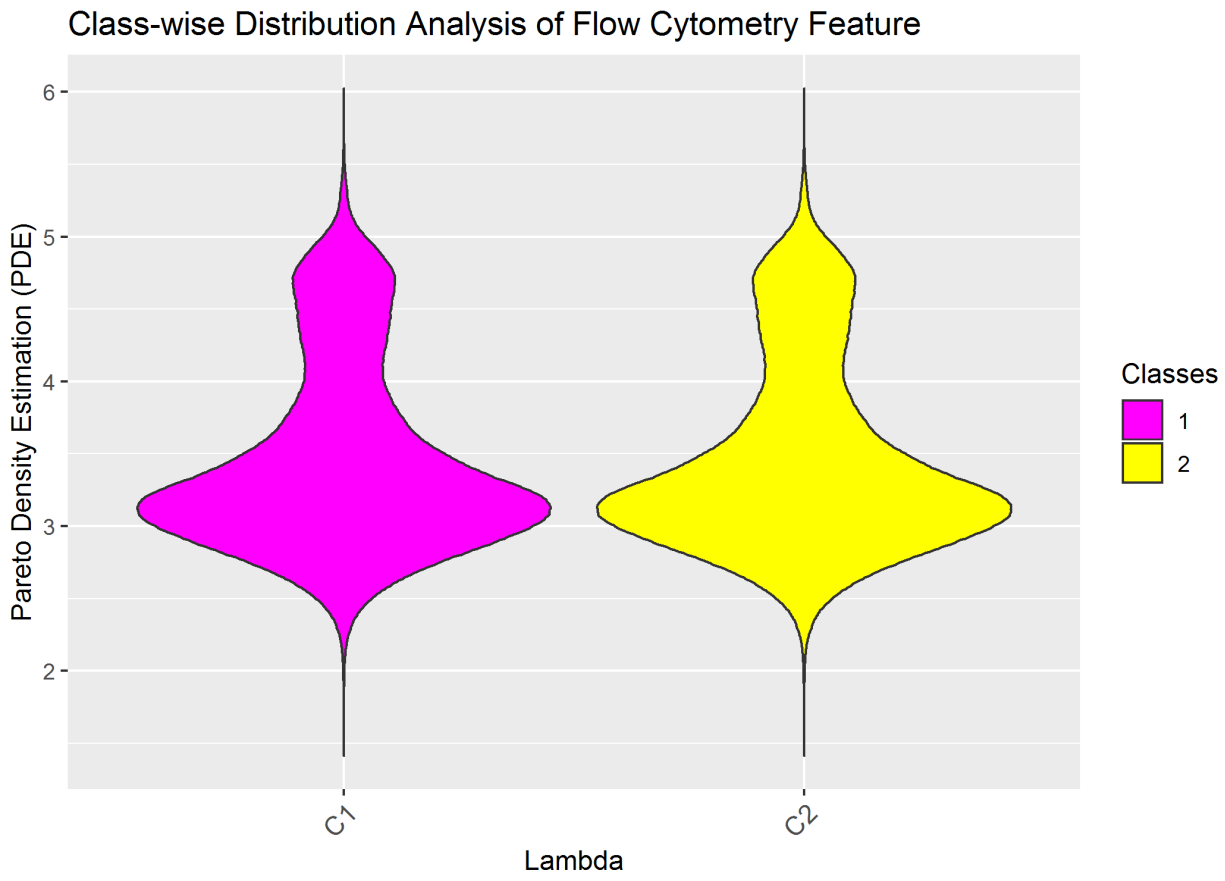
```
library(DataVisualizations)

ClassMDplot(Data[,'CD5'],Cls)$ggobject+ggtitle('Class-wise Distribution Analysis of␣
↪Flow Cytometry Feature')+xlab('CD5')+ylab('Pareto Density Estimation (PDE)')

ClassMDplot(Data[,'CD20'],Cls)$ggobject+ggtitle('Class-wise Distribution Analysis of␣
↪Flow Cytometry Feature')+xlab('CD20')+ylab('Pareto Density Estimation (PDE)')

ClassMDplot(Data[,'CD23'],Cls)$ggobject+ggtitle('Class-wise Distribution Analysis of␣
↪Flow Cytometry Feature')+xlab('CD23')+ylab('Pareto Density Estimation (PDE)')

ClassMDplot(Data[,'CD8kappa'],Cls)$ggobject+ggtitle('Class-wise Distribution Analysis␣
↪of Flow Cytometry Feature')+xlab('Kappa')+ylab('Pareto Density Estimation (PDE)')

ClassMDplot(Data[,'CD7lambda'],Cls)$ggobject+ggtitle('Class-wise Distribution␣
↪Analysis of Flow Cytometry Feature')+xlab('Lambda')+ylab('Pareto Density Estimation␣
↪(PDE)')
```




Class-wise Distribution Analysis of Flow Cytometry Feature
Pareto Density Estimation (PDE)
6
4
2
C1
C2
CD20
Classes
1
2


Class-wise Distribution Analysis of Flow Cytometry Feature
Pareto Density Estimation (PDE)
6
5
4
3
2
C1
C2
CD23
Classes
1
2


Class-wise Distribution Analysis of Flow Cytometry Feature
Pareto Density Estimation (PDE)
6
5
4
3
2
1
C1
C2
Kappa
Classes
1
2



Of course we do not see any difference because the data was not gated priorly and because one feauture alone is insufficent.

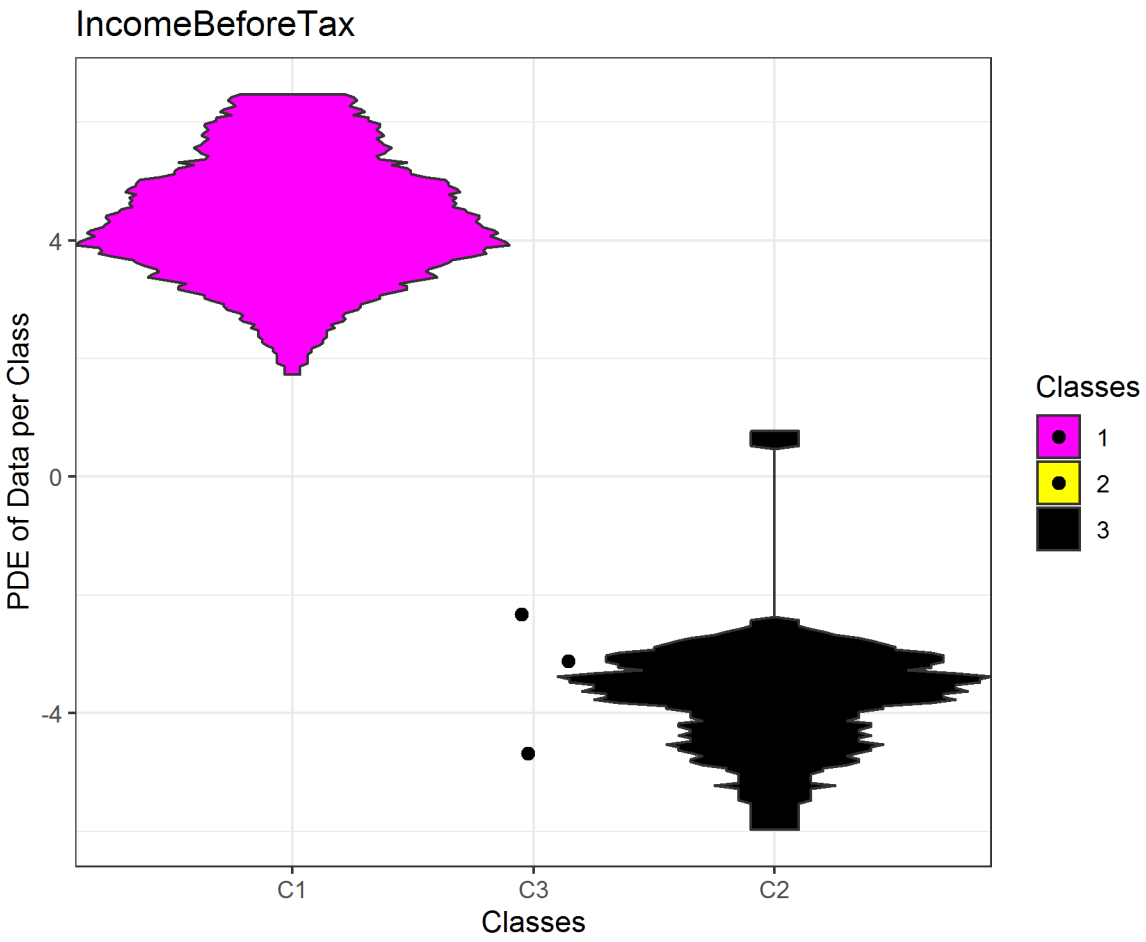
5.5 Content of High-Dimensional Accounting Information

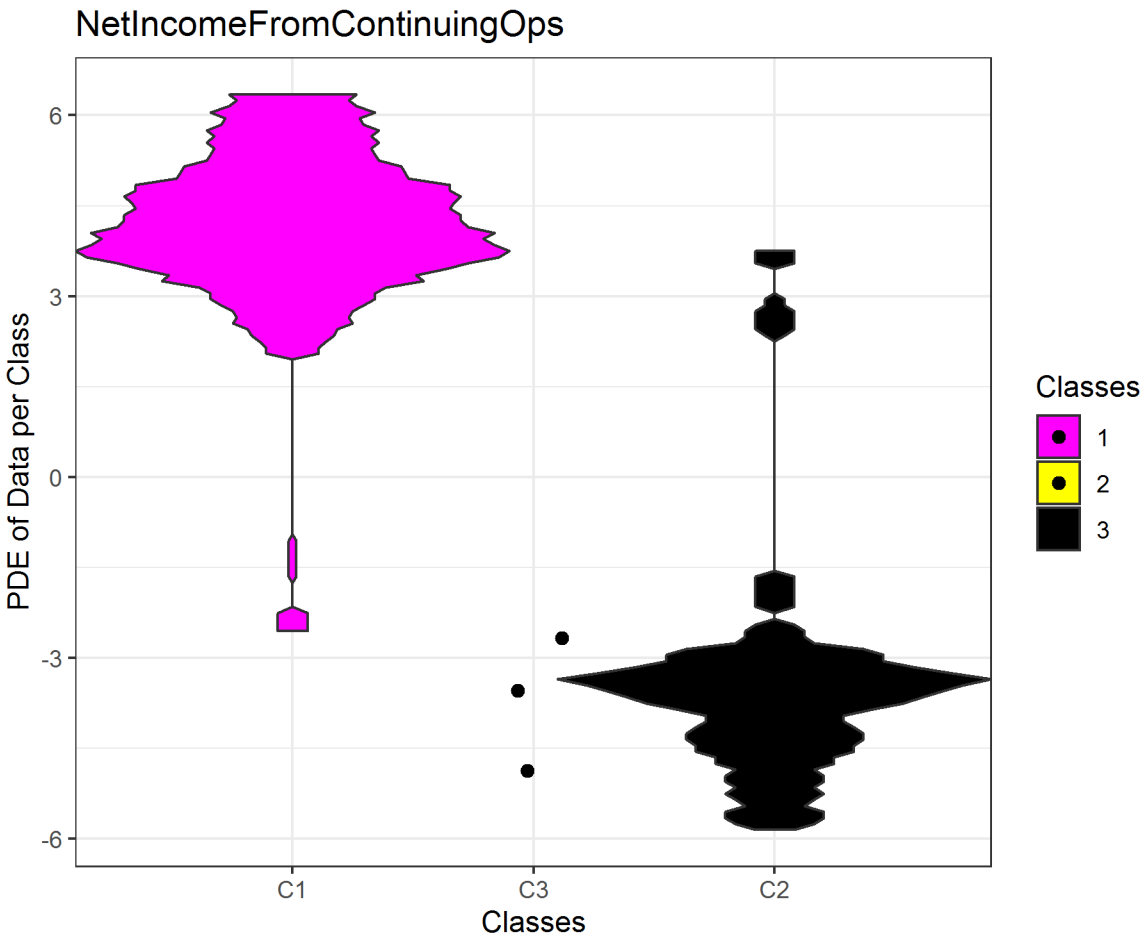
We use a clustering of manuscript currently under review [Thrun/Ultsch, 2019] in order to show that the the high-dimensional clustering can be distinguished by one single feature. The data consists of Aaccounting information of 261 companies traded in the Frankfurt stock exchange in the German Prime standard. We select four features which are understandable (details, see [Thrun/Ultsch, 2019]) when published.

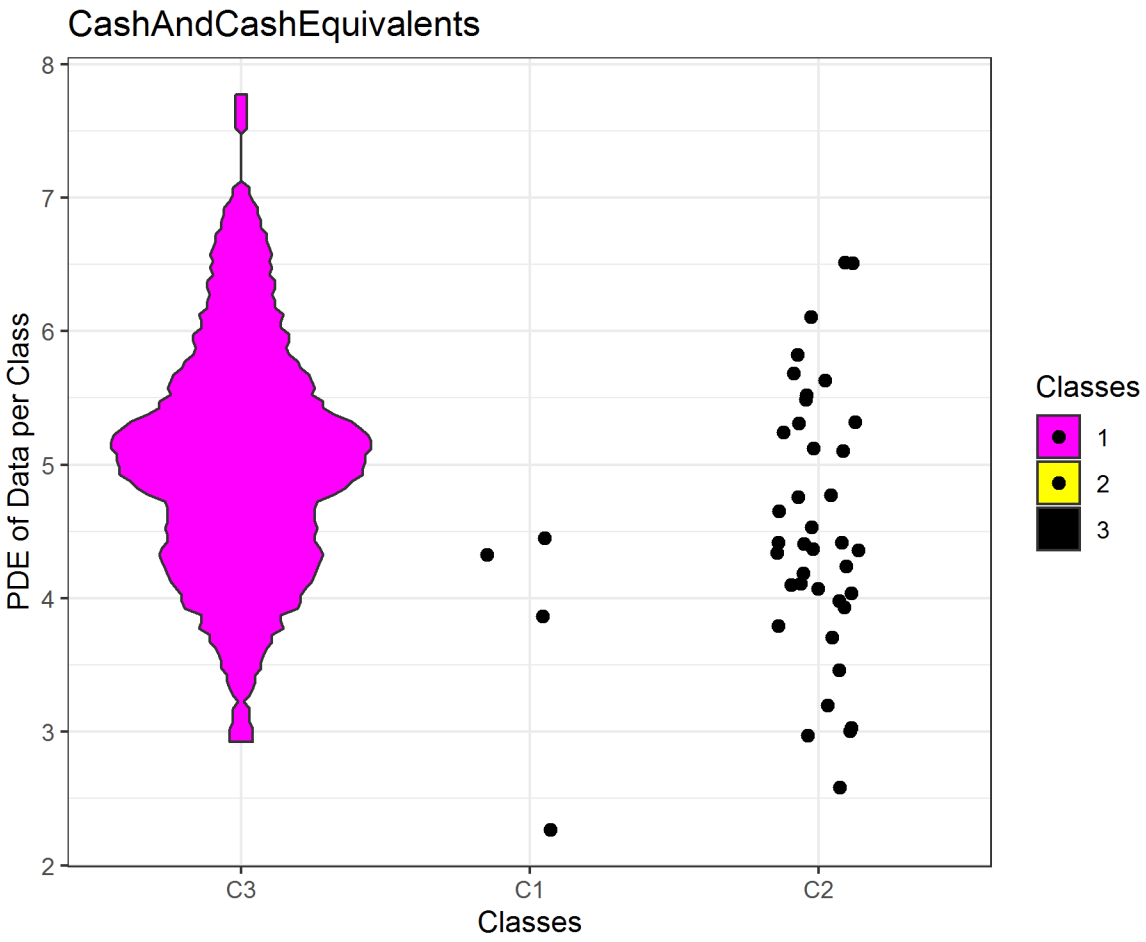
5.6 Class-wise MD plot for Accounting Information Explains Clustering

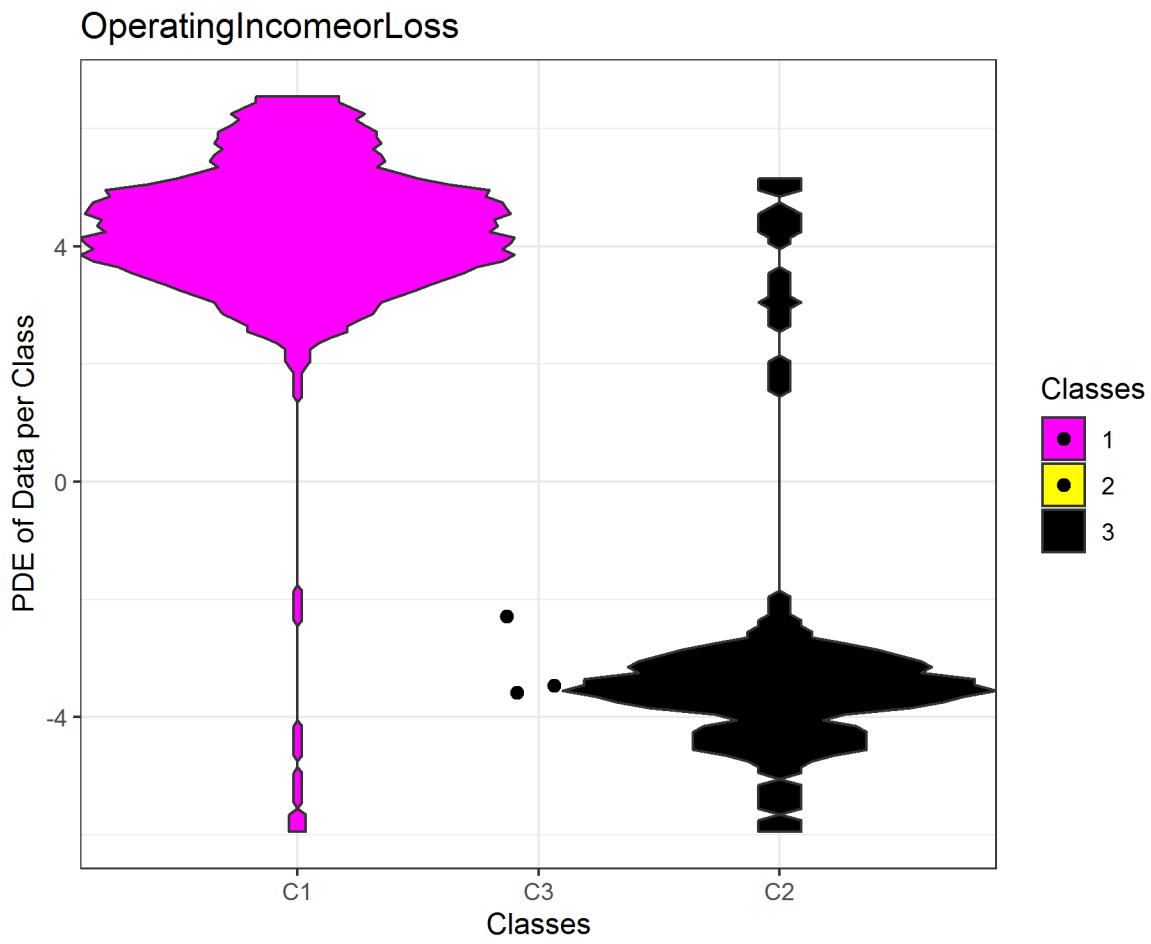
The features have a large positive and negativ range. Therefore, we first compute the absolute log multiplied by the sign of the data observation. Thus, the data is then visualized on the logarithmic scale. The first feature is clearly seperated by the clustering, the other three features are not completly seperated. If not enough data is available for the estimation of the probability density function (pdf), scatter plots are drawn. In sum, the clustering of the high-dimensional data set of 45 features can be explained by one single feature afterwards if outliers are disregarded (cluster 3)!

```
data('AccountingInformation_PrimeStandard_Q3_2019.rda')
str(AI_PS_Q3_2019)
Data=AI_PS_Q3_2019$Data
Cls=AI_PS_Q3_2019$Cls

NamesTarget=c('IncomeBeforeTax','NetIncomeFromContinuingOps',"CashAndCashEquivalents",
↪"OperatingIncomeorLoss")
TargetFeatures=DataVisualizations::SignedLog(Data[,match(table = colnames(Data),
↪NamesTarget)])

i=1
ClassMDplot(TargetFeatures[,i],Cls,main = colnames(TargetFeatures)[i])$ggobject+theme_
↪bw()
#ggsave(filename=paste0(colnames(TargetFeatures)[i],'.png'))

i=2
ClassMDplot(TargetFeatures[,i],Cls,main = colnames(TargetFeatures)[i])$ggobject+theme_
↪bw()
#ggsave(filename=paste0(colnames(TargetFeatures)[i],'.png'))

i=3
ClassMDplot(TargetFeatures[,i],Cls,main = colnames(TargetFeatures)[i])$ggobject+theme_
↪bw()
#ggsave(filename=paste0(colnames(TargetFeatures)[i],'.png'))

i=4
ClassMDplot(TargetFeatures[,i],Cls,main = colnames(TargetFeatures)[i])$ggobject+theme_
↪bw()
#ggsave(filename=paste0(colnames(TargetFeatures)[i],'.png'))
```


IncomeBeforeTax
PDE of Data per Class
4
0
-4
C1
C3
C2
Classes
Classes
1
2
3


NetIncomeFromContinuingOps
PDE of Data per Class
6
3
0
-3
-6
C1
C3
C2
Classes
Classes
1
2
3


CashAndCashEquivalents
PDE of Data per Class
8
7
6
5
4
3
2
C3
C1
C2
Classes
Classes
1
2
3



[Thrun/Ultsch, 2019] Thrun, M. C., & Ultsch, A.: Stock Selection via Knowledge Discovery using Swarm Intelligence with Emergence, IEEE Intelligent Systems, Vol. under review, pp., 2019.