Non Cartesian representations (part 2)

Serial axes and high dimensional glyphs

R.W. Oldford

University of Waterloo

Serial Axes

Cartesian coordinate systems are based on orthogonal axes

- every axis is orthogonal to every other axis
- every set of axes is orthogonal to every other distinct set
- which variates are matched to which axes doesn't matter

In contrast, radial axes are an example of a **serial** axis system.

- variates are most easily compared if they appear beside each other.
- the order of the axes matters (in that the display produced is different)

With **serial axes** systems the axes follow an order and the order affects the display.

Pairwise ordering of axes

Repeating axes/variates allows any variate to appear beside (and be compared with) any other.

Consider the complete graph having variates as nodes.

- every path is a selection and ordering of axes
- an Eulerian will have **every** pair of variates appear beside one another
- an ordering of all axes with no repeats is a Hamiltonian

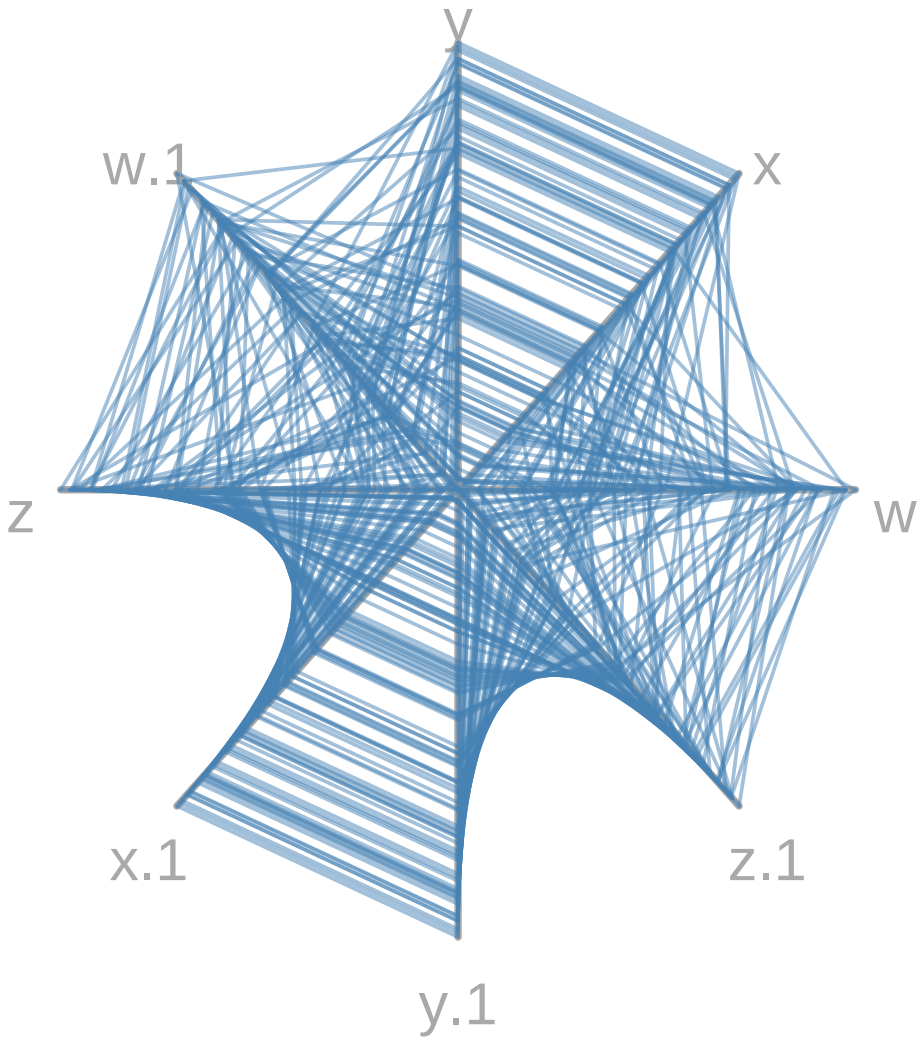
If we have some measure (e.g. correlation) on a pair of variates, we could use these as weights to choose an ordering. For example,

- the early part of a greedy Eulerian will tend to have small/large weights
- a weighted Hamiltonian could be used to produce an ordering of minimal/maximal total weight

The choices could effect very different displays.

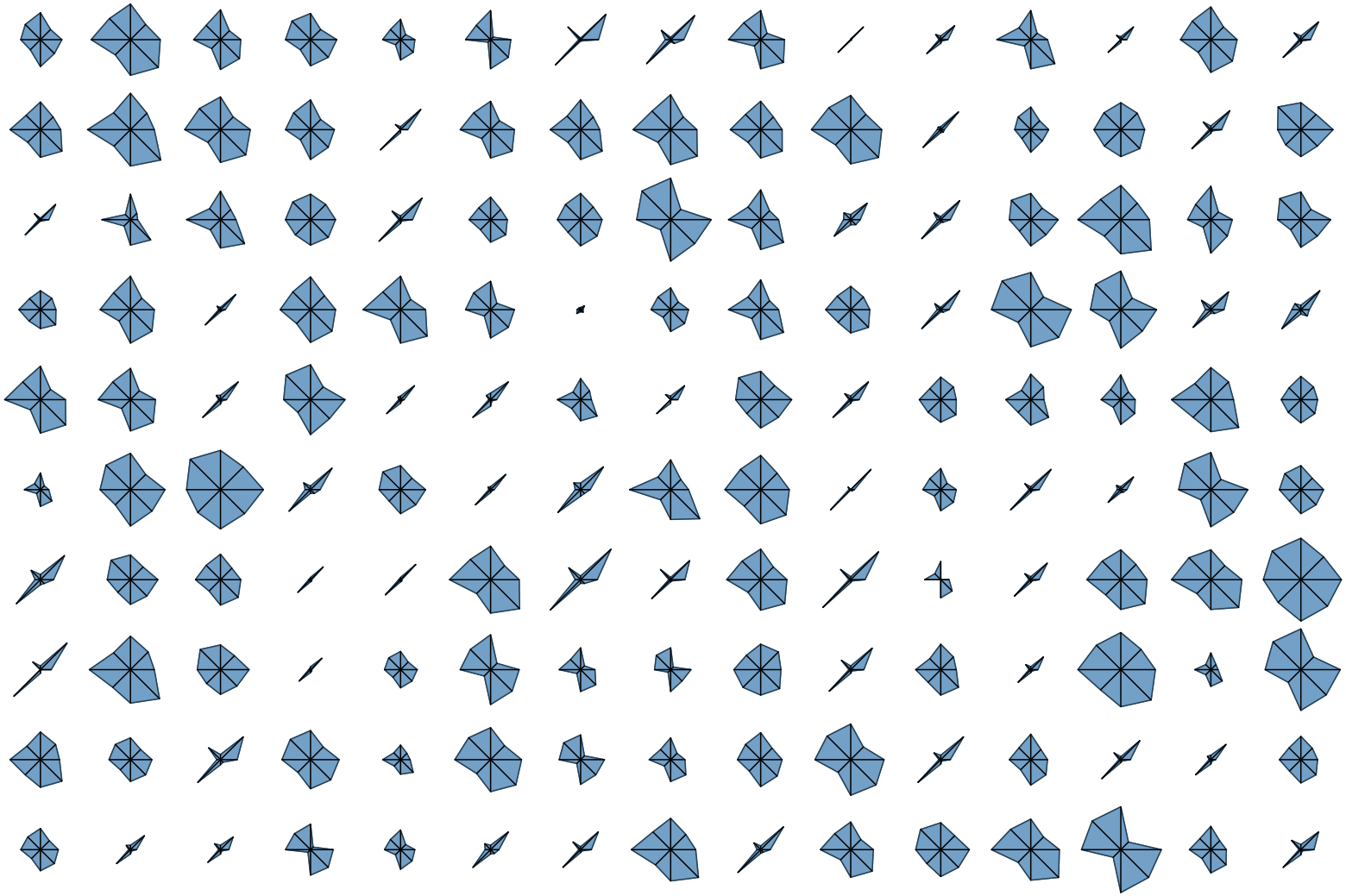
Pairwise ordering of axes - Eulerian

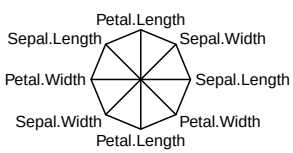
On the fake data



Pairwise ordering of axes - Eulerian

The iris data

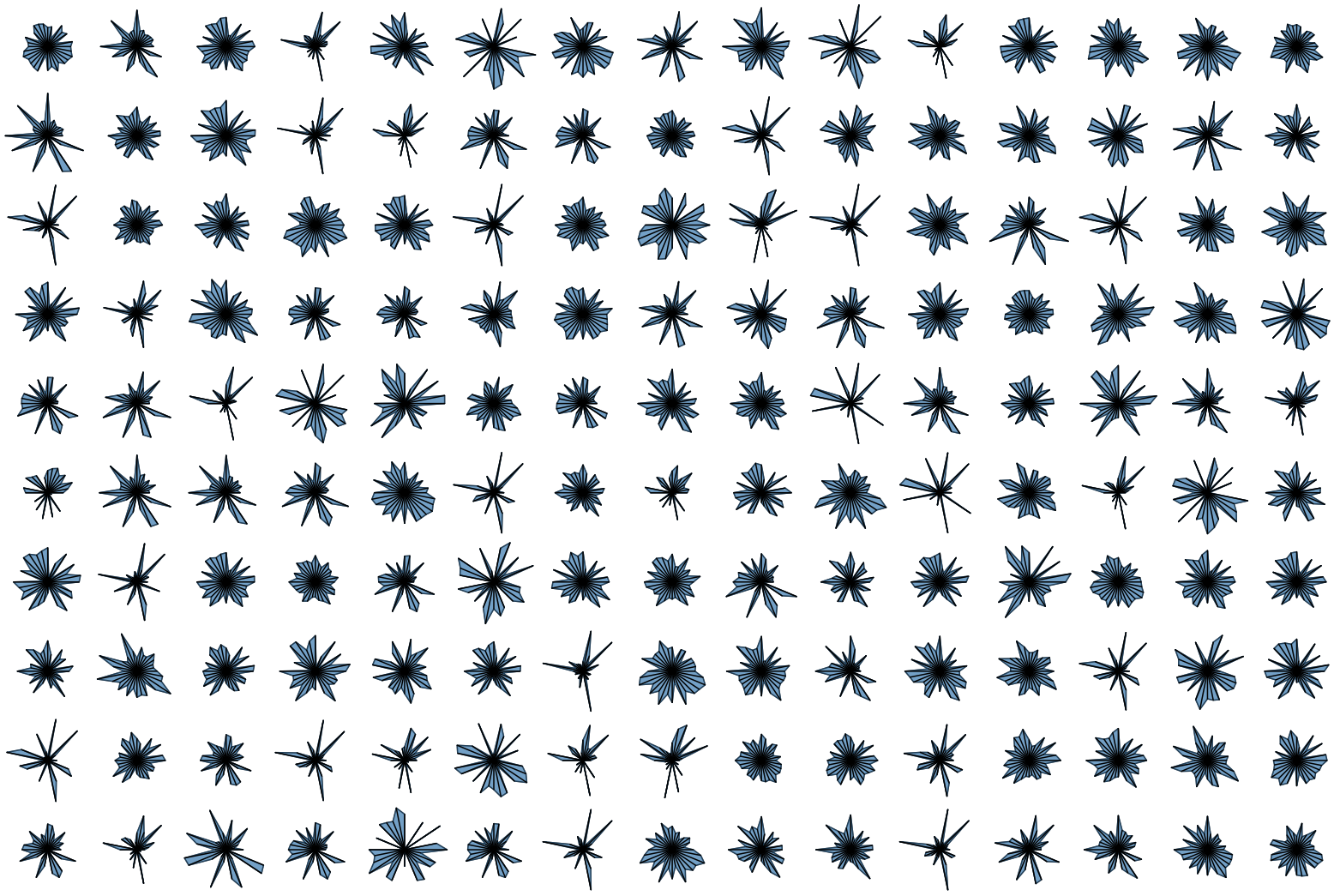




University of Waterloo

Pairwise ordering of axes - Eulerian

The olive data



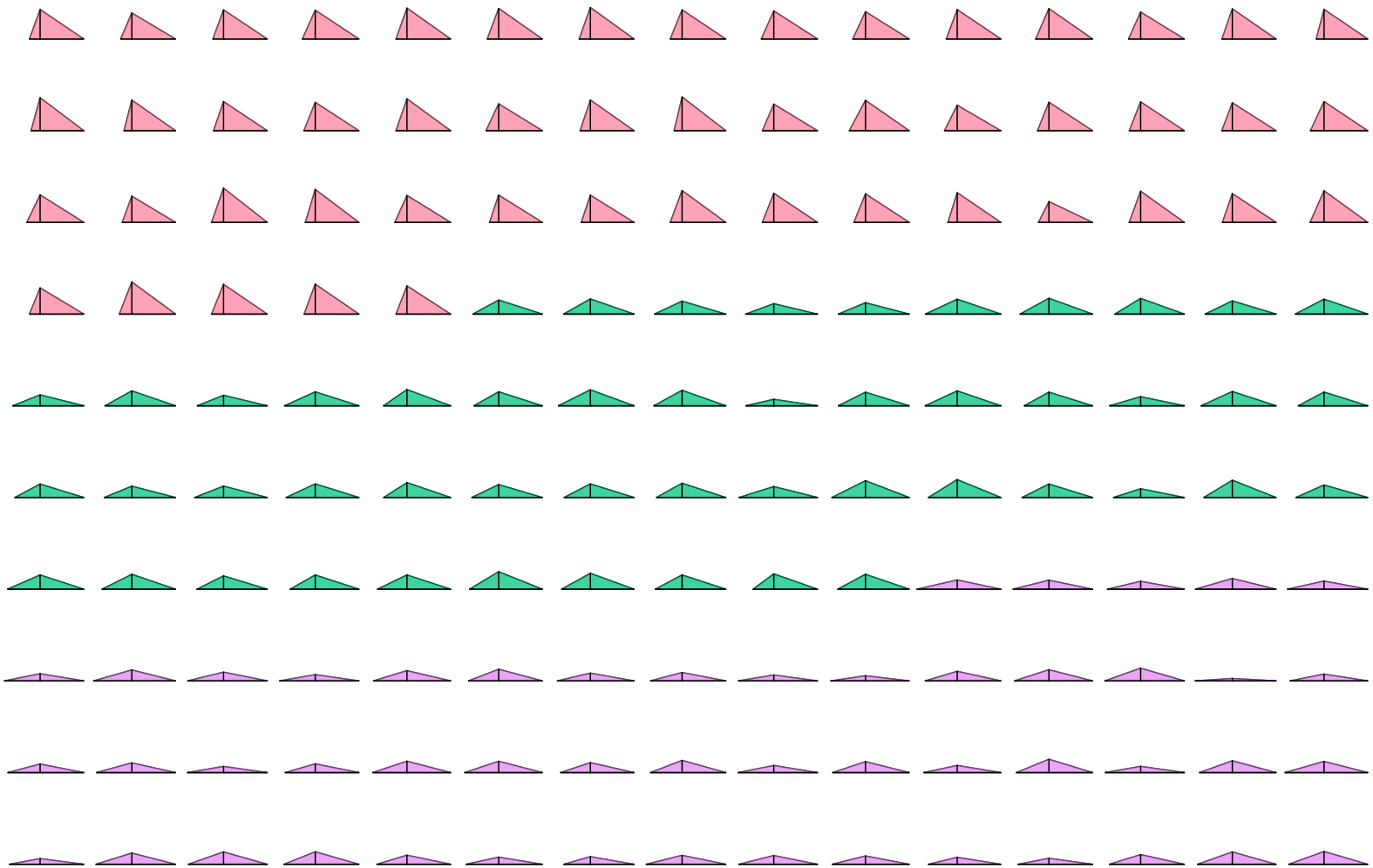
Pairwise ordering of axes - Scaling

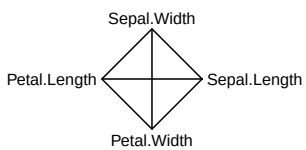
Occasionally it makes sense to scale the data in other ways (not just by variate ranges). For example, for the iris data scaling each flower by its largest measure (sepal length) compares the flower shapes and ignores flower sizes.

```
data <- iris[,-5] # the iris data
title <- "Iris data"
# Scale by row
data <- apply(data, 1, scale01)
# transpose to get the flowers as rows
data <- t(data)
# A function to get a colour for each species (not available to you)
cols <- get_cols(iris["Species"])
# First no repetition of the variates; preserve order
stars(data, labels=NULL,main=title, radius=TRUE,
      lwd=1.25, col.lines = cols, col.stars = cols,
      nrow=10, ncol = 15, key.loc=c(12,-0.5),
      ## data scaling used as is
      scale=FALSE
      )
# Now all pairs appear, again flowers appear in order
stars(data[,eseq(p)],labels=NULL,main=title, radius=TRUE,
      lwd=1.25, col.lines = cols, col.stars = cols,
      nrow=10, ncol = 15, key.loc=c(12,-0.5),
      ## data scaling used as is
      scale=FALSE
      )
```

Pairwise ordering of axes - scaling

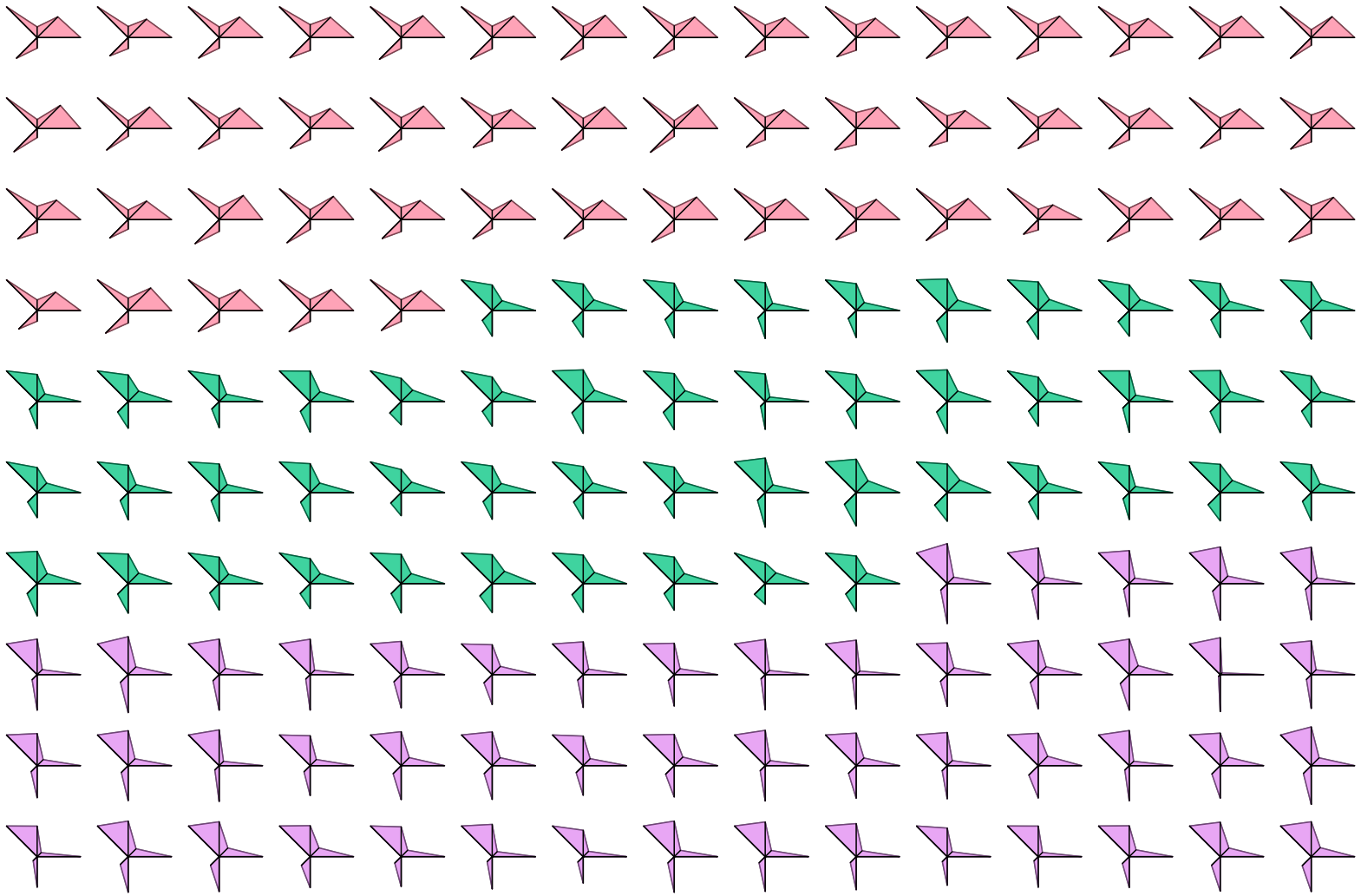
Comparing flower shapes. Flowers in order (50 in each group); no variates repeated.

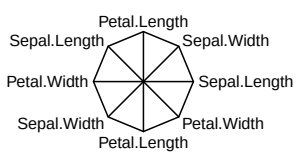




Pairwise ordering of axes - Scaling

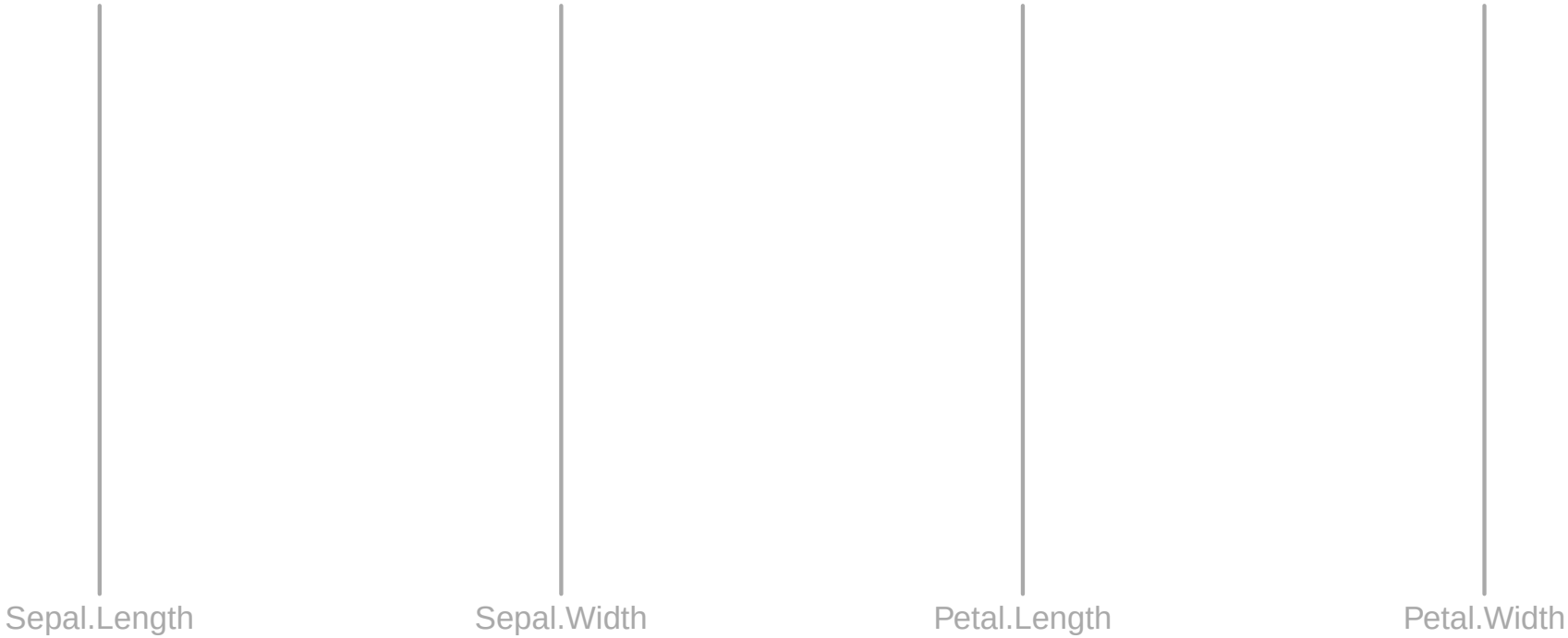
Comparing flower shapes. Flowers in order (50 in each group); All pairs appear (Eulerian).





Parallel axes

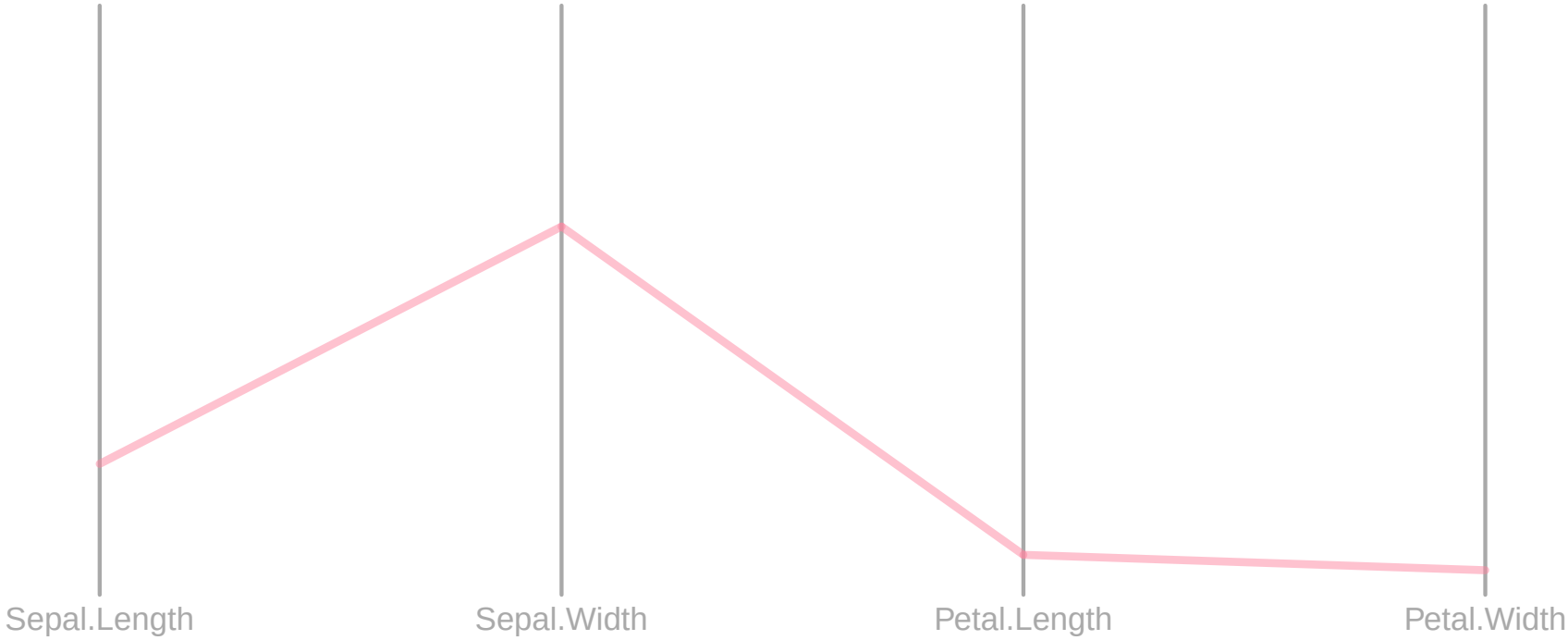
Axes might also be laid out **equi-spaced in parallel** to one another.



These are also called **parallel coordinate** plots (Inselberg).

Parallel axes

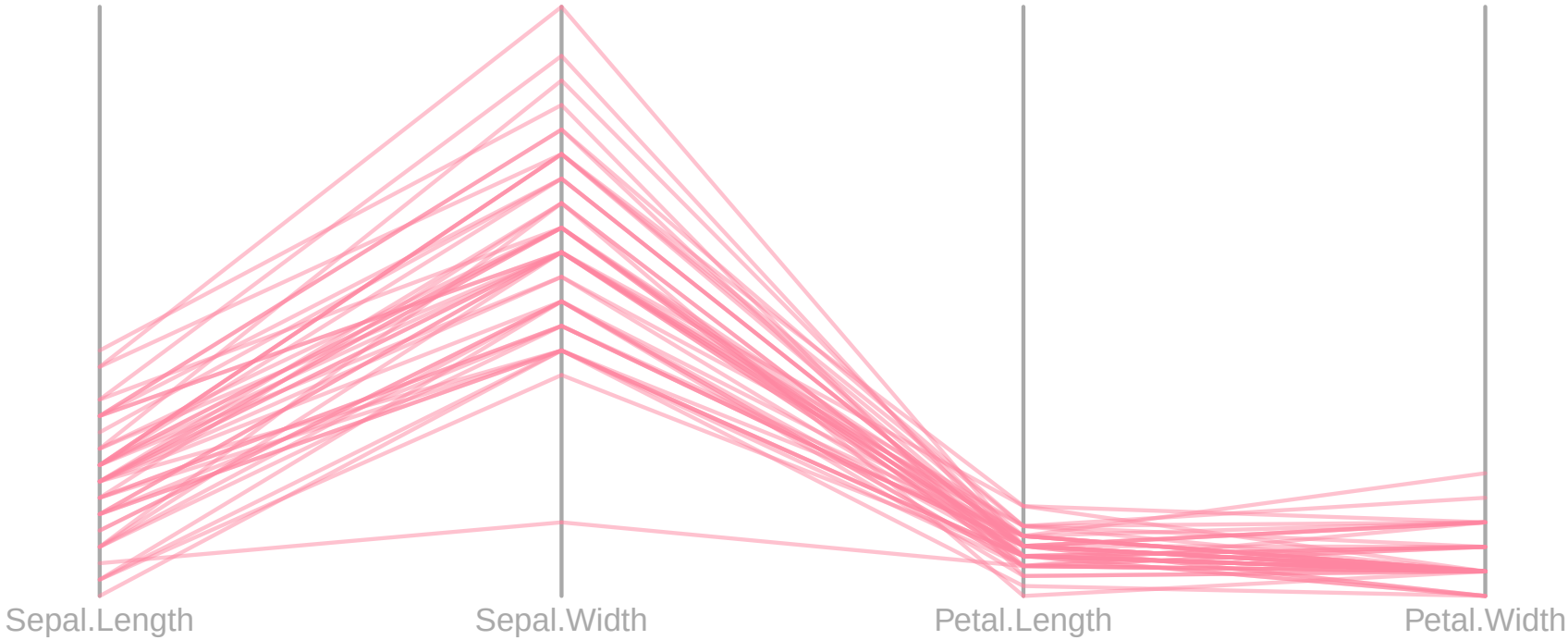
Axes might also be laid out equi-spaced in parallel to one another.



Each point becomes a "curve" formed from the line segments connecting the values on each coordinate axis. Shown above is one *Iris Setosa* flower's measurements.

Parallel axes

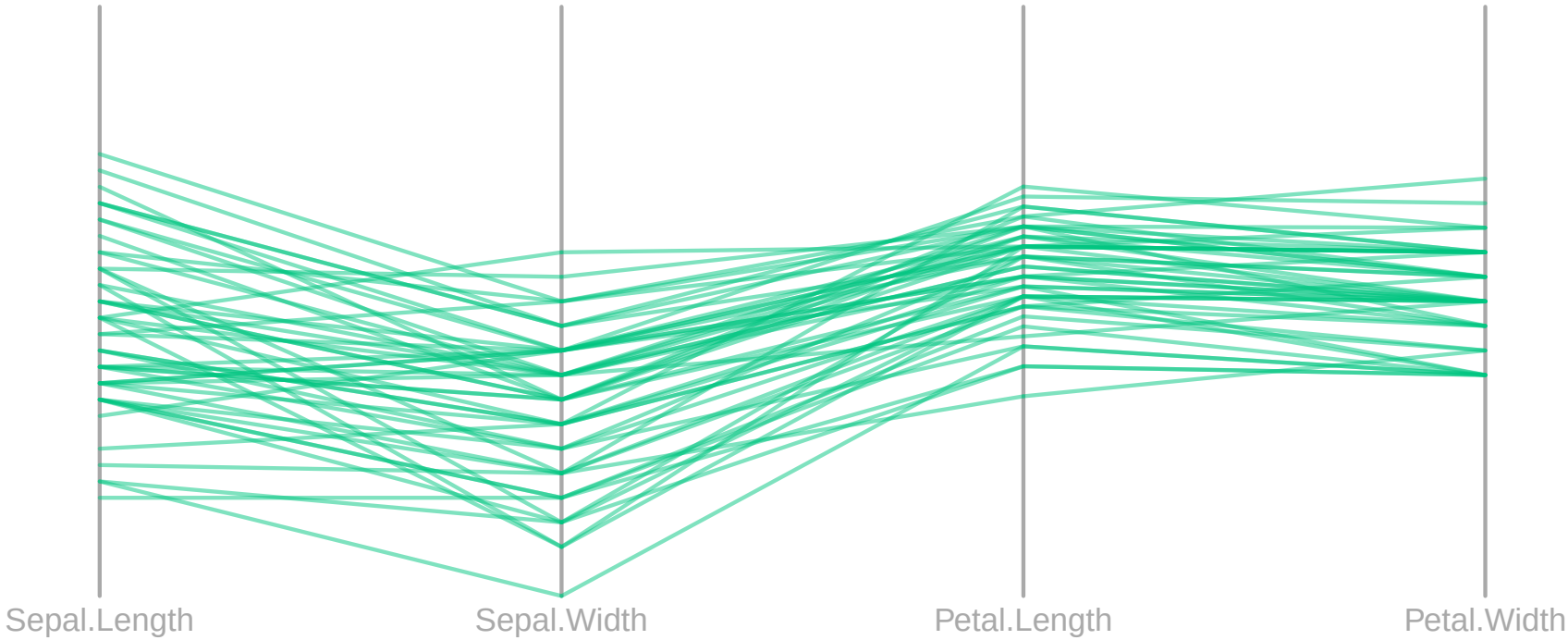
Overlaying all *Iris Setosa* flowers:



The collection of flowers show positive and negative correlations

Parallel axes

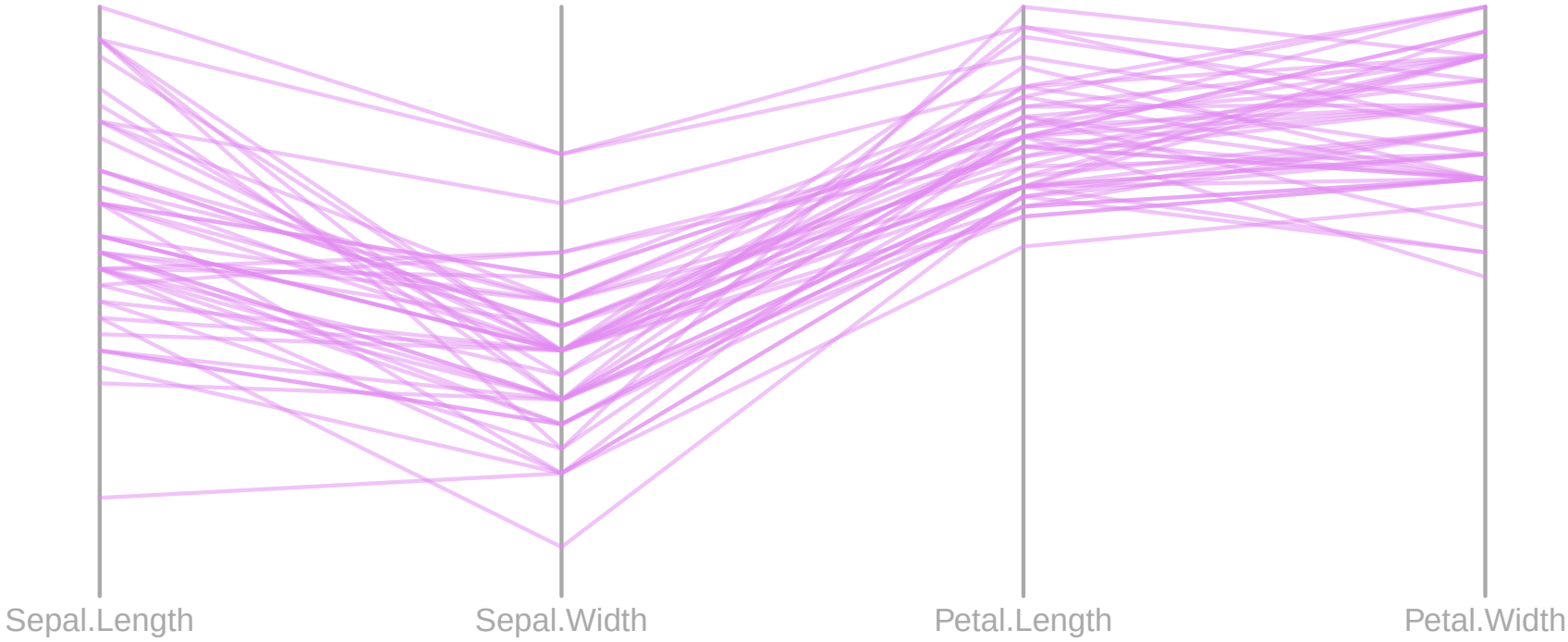
Overlaying all *Iris Versicolor* flowers:



Each collection of flowers show correlations and distinctive shapes.

Parallel axes

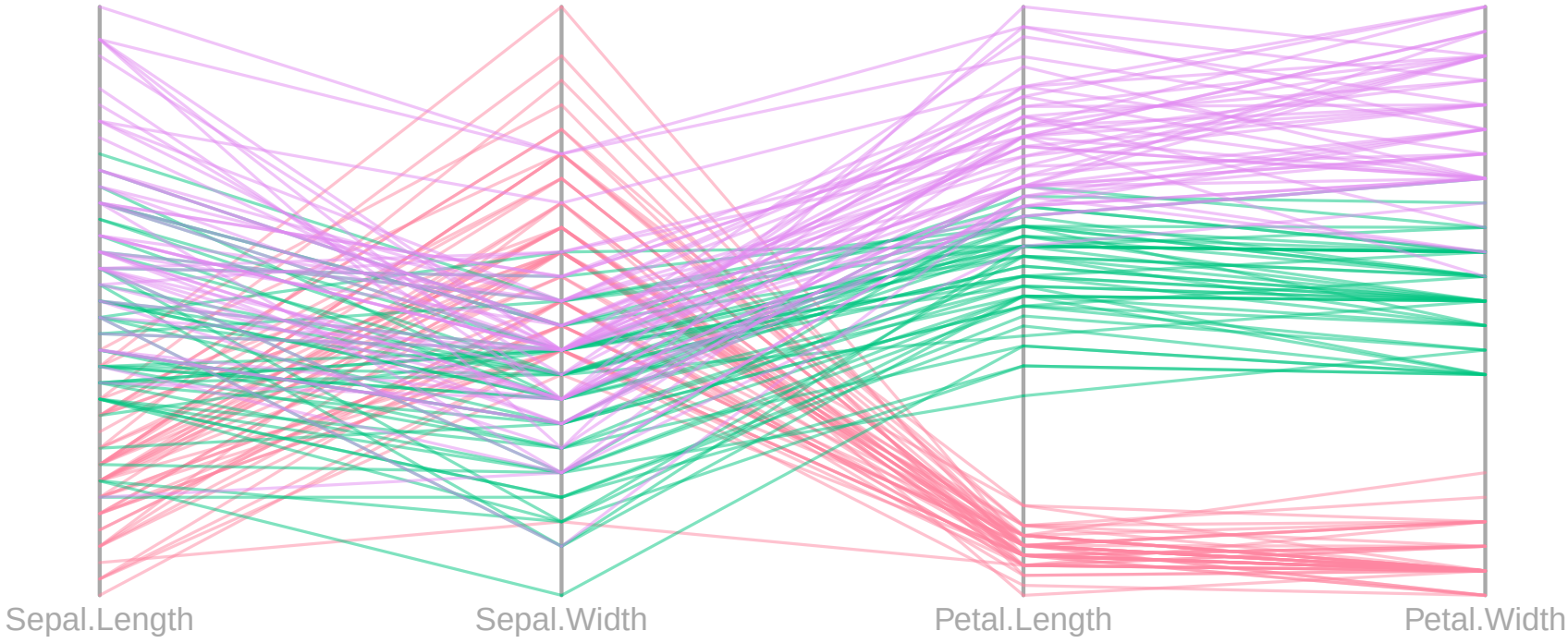
Overlaying all *Iris Virginica* flowers:



Each collection of flowers show correlations and distinctive shapes.

Parallel axes

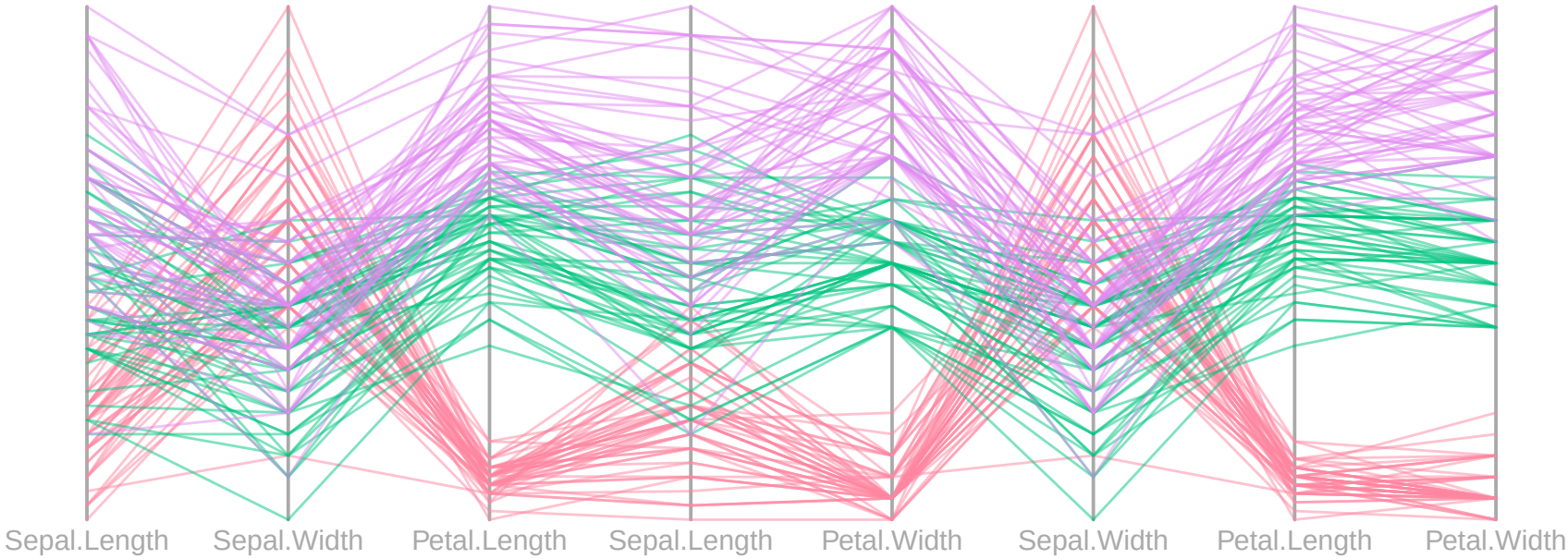
All three species together:



Each collection of flowers show correlations and distinctive shapes.

Parallel axes

All three species together **and** all pairs of variates:



Each collection of flowers show correlations and distinctive shapes.

Line-point duality

Each pair of parallel axes corresponds to a pair of orthogonal axes. Patterns seen in a scatterplot should also appear as patterns in the parallel coordinates.

There is in fact a **line-point duality** that connects the geometry within a orthogonal coordinate pair system to that of the geometry of a parallel coordinate pair system.

- A point in a scatter plot (pair of orthogonal coordinates) is a line between parallel axes (a pair of parallel coordinates)
- A line in a scatter plot (pair of orthogonal coordinates) is a point of intersection of lines between parallel axes (a pair of parallel coordinates)
  - intersection may occur **outside** the parallel axes, possibly at infinity
  - i.e. if you continue the lines beyond the axes they will intersect
- two parallel lines (same slope) in a scatterplot correspond to two different intersection points one above the other in a parallel coordinate plot;
- lines with common intercepts in a scatterplot **do NOT produce** points of intesections in the parallel coordinate plot at the same vertical location; changing slope changes the horizontal **and** vertical position of the point of intersection.
- a curve in a scatterplot is a curve of intersection points in parallel coordinates (think tangent lines, or lines formed by successive points along the curve)

These points can be easily illustrated using `loon`.

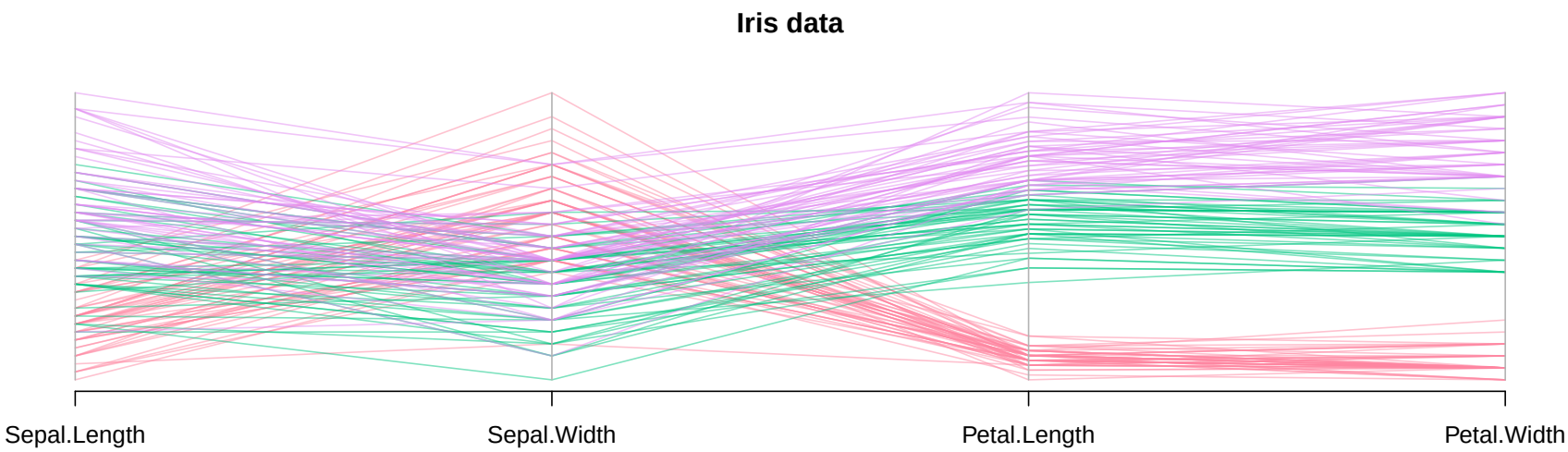
```
library(loon)
oldDir <- getwd()
# Change directory to wherever you saved this code
DataViz <- "/Users/rwoldford/Documents/Admin/courses/Data Visualization
ThisLecture <- "4. Higher dimensional data/A. Non-Cartesian Representat
Rcode <- paste(DataViz,ThisLecture,"R", sep="/")
setwd(Rcode)
source("parallelCoordPointLine.R")
setwd(oldDir)
```

Select each group of interest; invert the others and deactivate them; brush points and lines to observe connection. Select subsets of points and move them around. Jitter points.

Reactivate all points, select and return all to their original position. Repeat with different groups (select by colour).

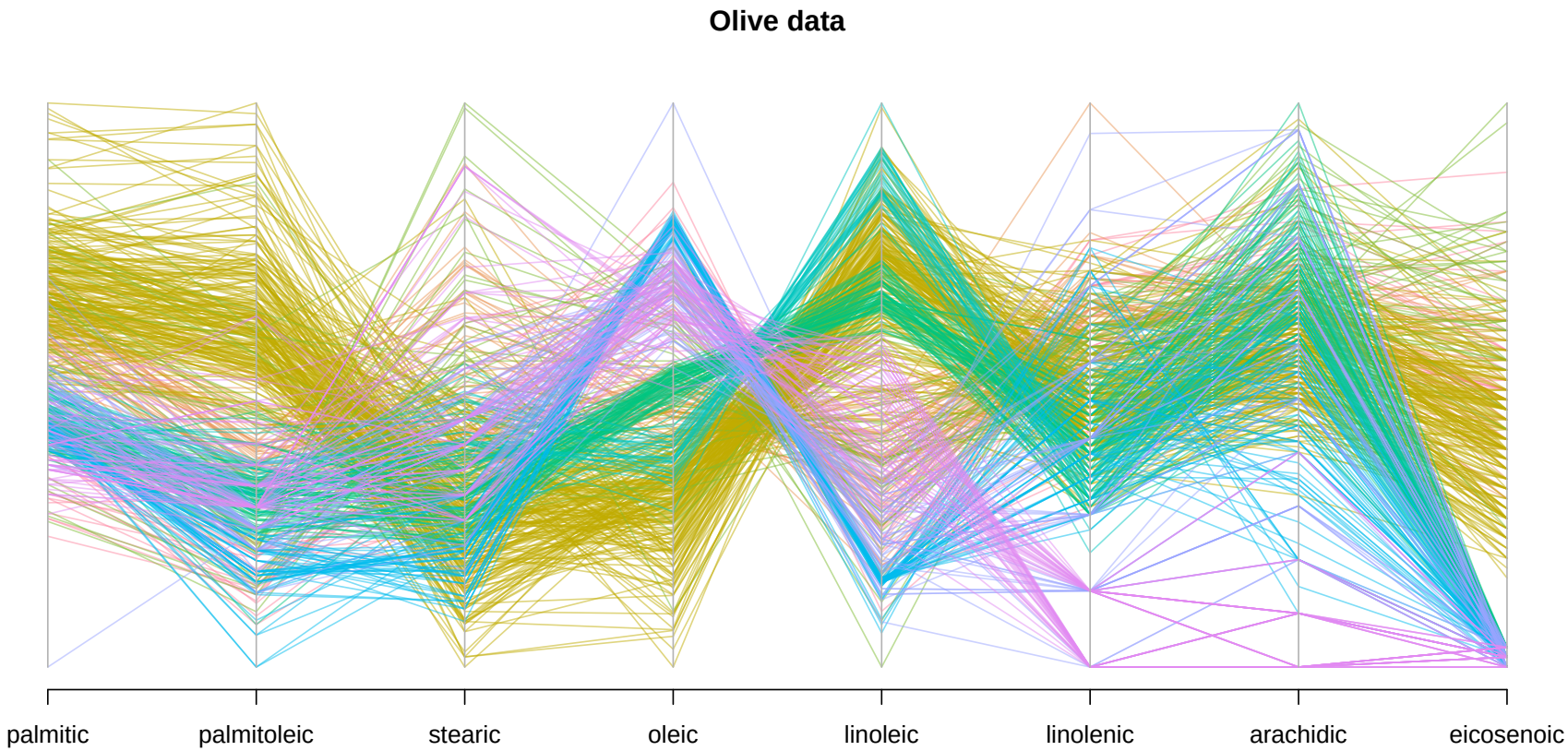
Parallel coordinates

```
data <- iris[,-5]
title <- "Iris data"
library(MASS)
parcoord(data, col=cols, main=title)
```

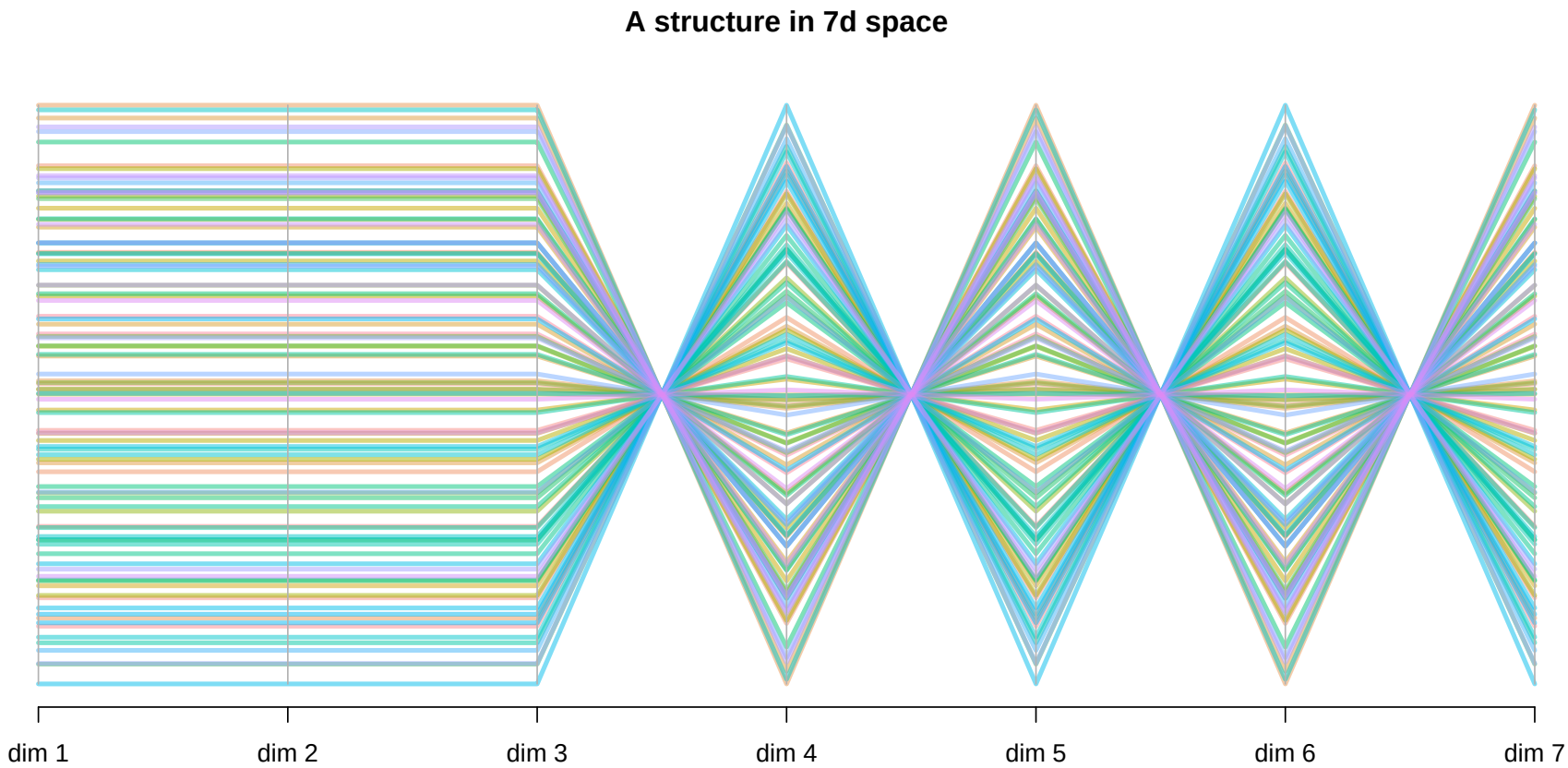


Parallel coordinates

```
data <- olive[,-c(1,2)]
title <- "Olive data"
parcoord(data, col=get_cols(olive[,"Area"]),lwd=1, main=title)
```

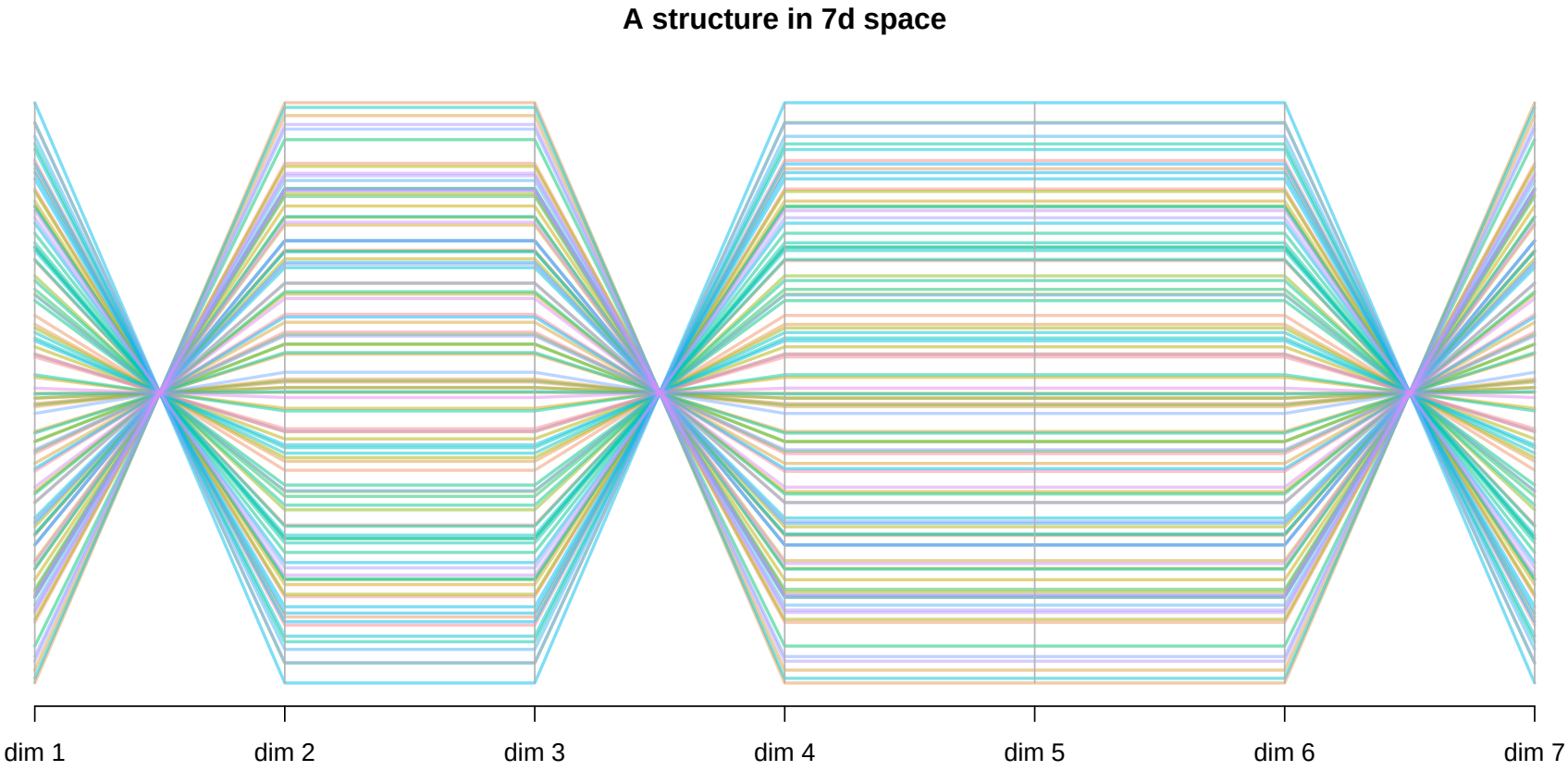


Some structure



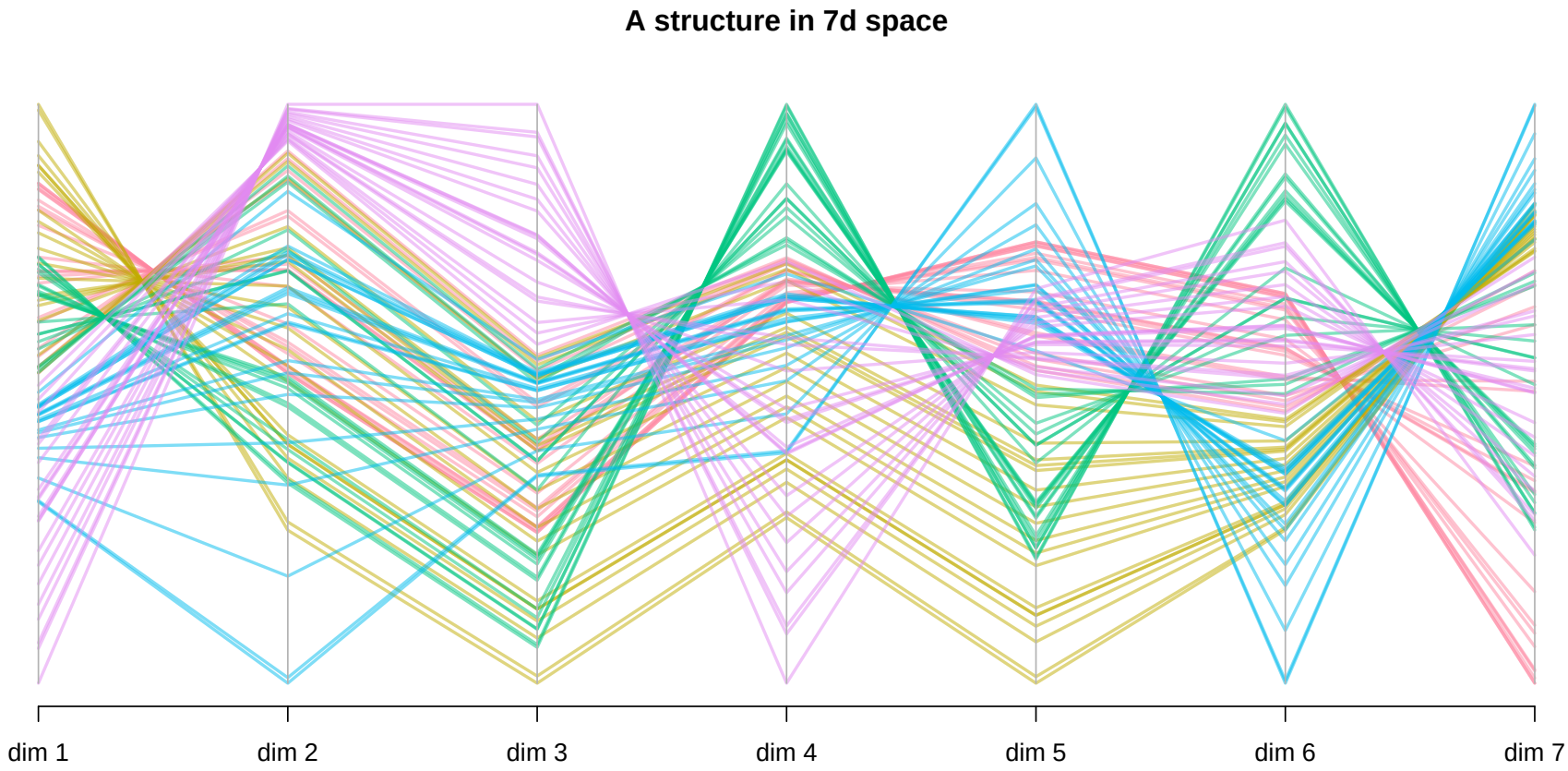
A line in 7 dimensions is always a line (or a point) in every lower dimensional projection.

Some structure



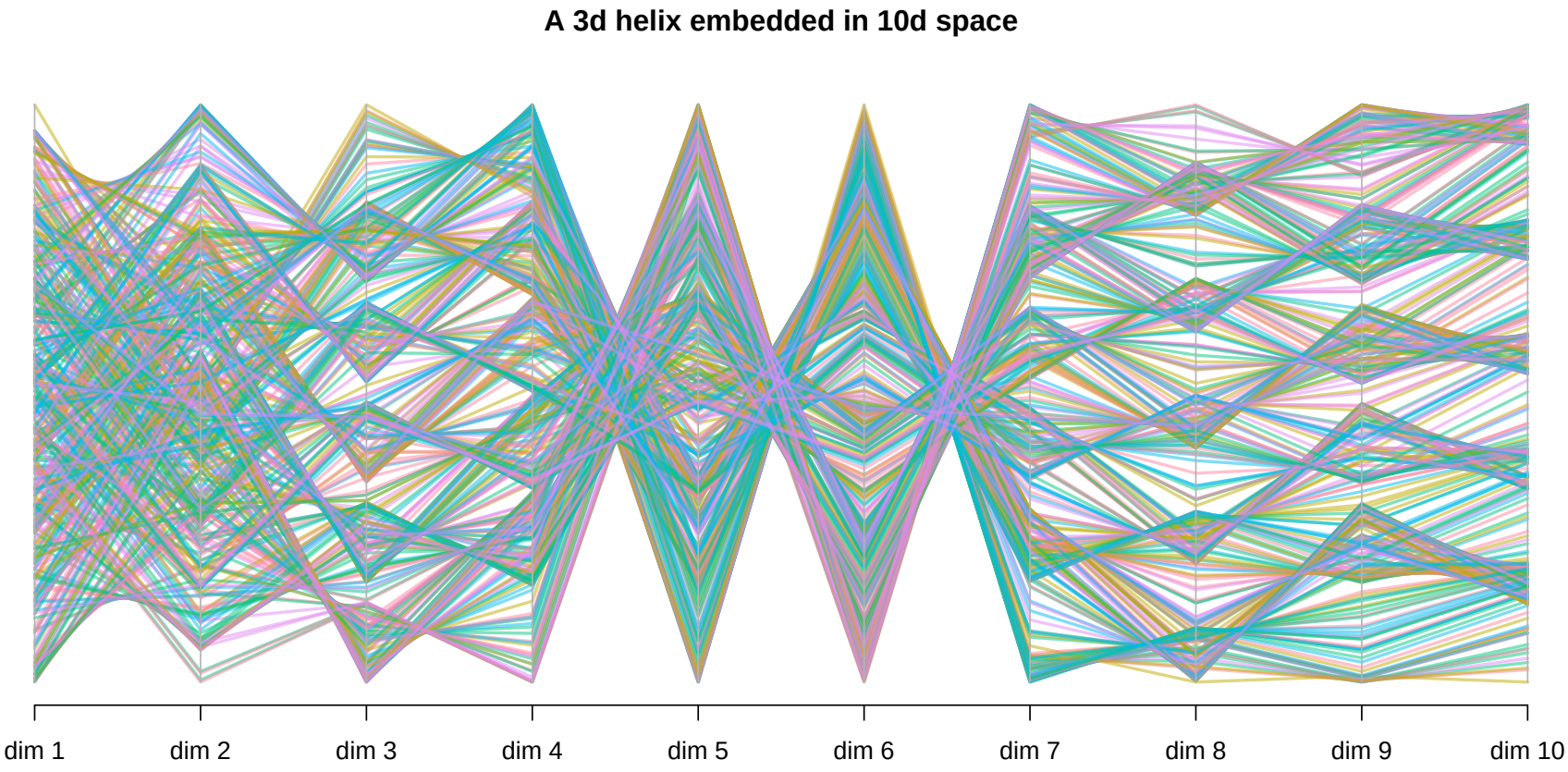
Another line in 7 dimensions. Again it must appear as a line (or a point) in every lower dimensional projection.

Some structure



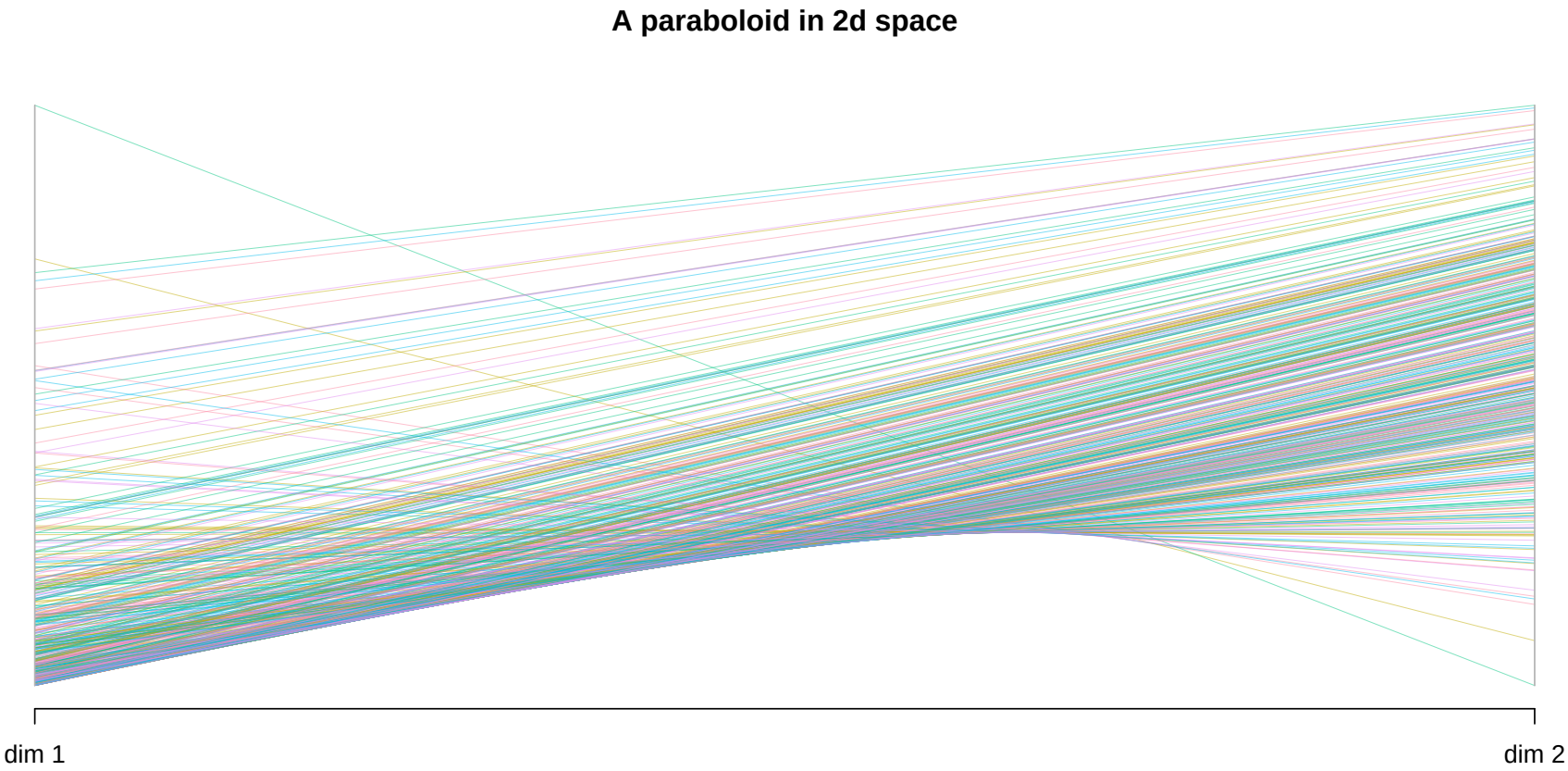
Several (actually 5) different lines in 7 dimensions each given a different colour. More easily explored in `loon`.

Some structure



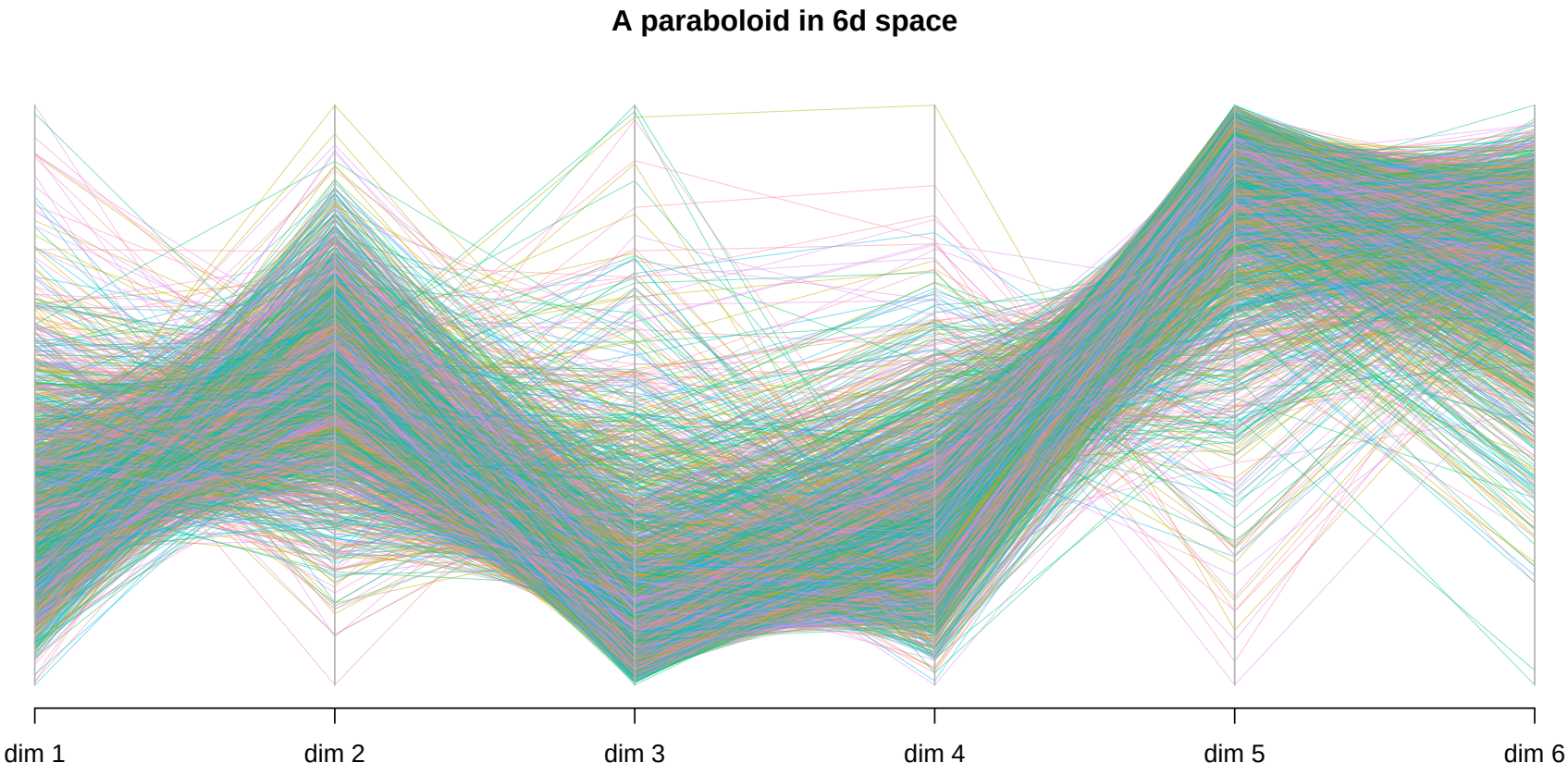
Clearly some structure is visible. Note projections which are nearly a line; and those where several parallel lines appear. In the latter there seem to be alternating slopes.

Some structure



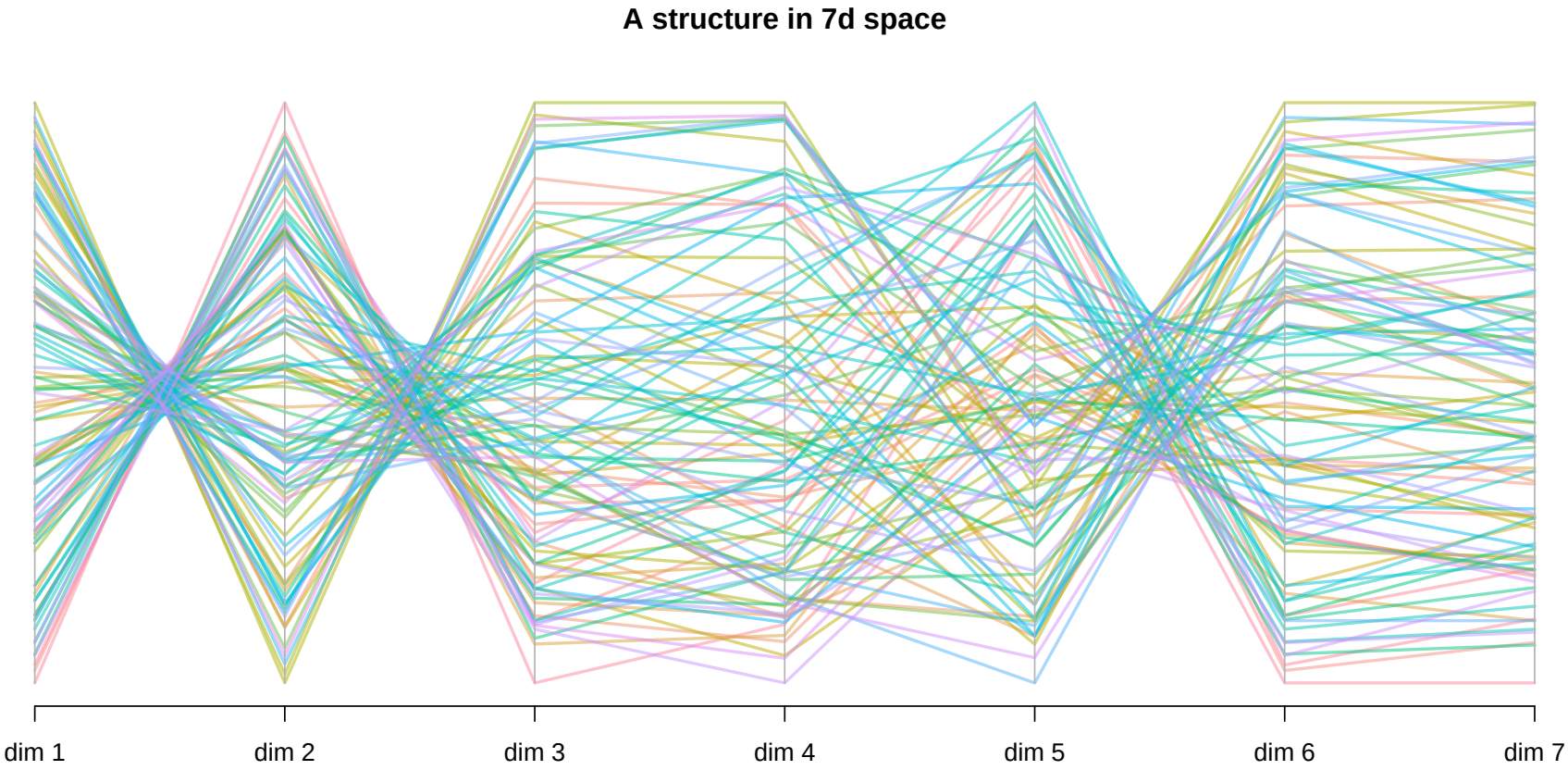
Smooth curvature indicates some smooth curves in those dimensions.

Some structure



Smooth curvature indicates some smooth curves in those dimensions.

Some structure

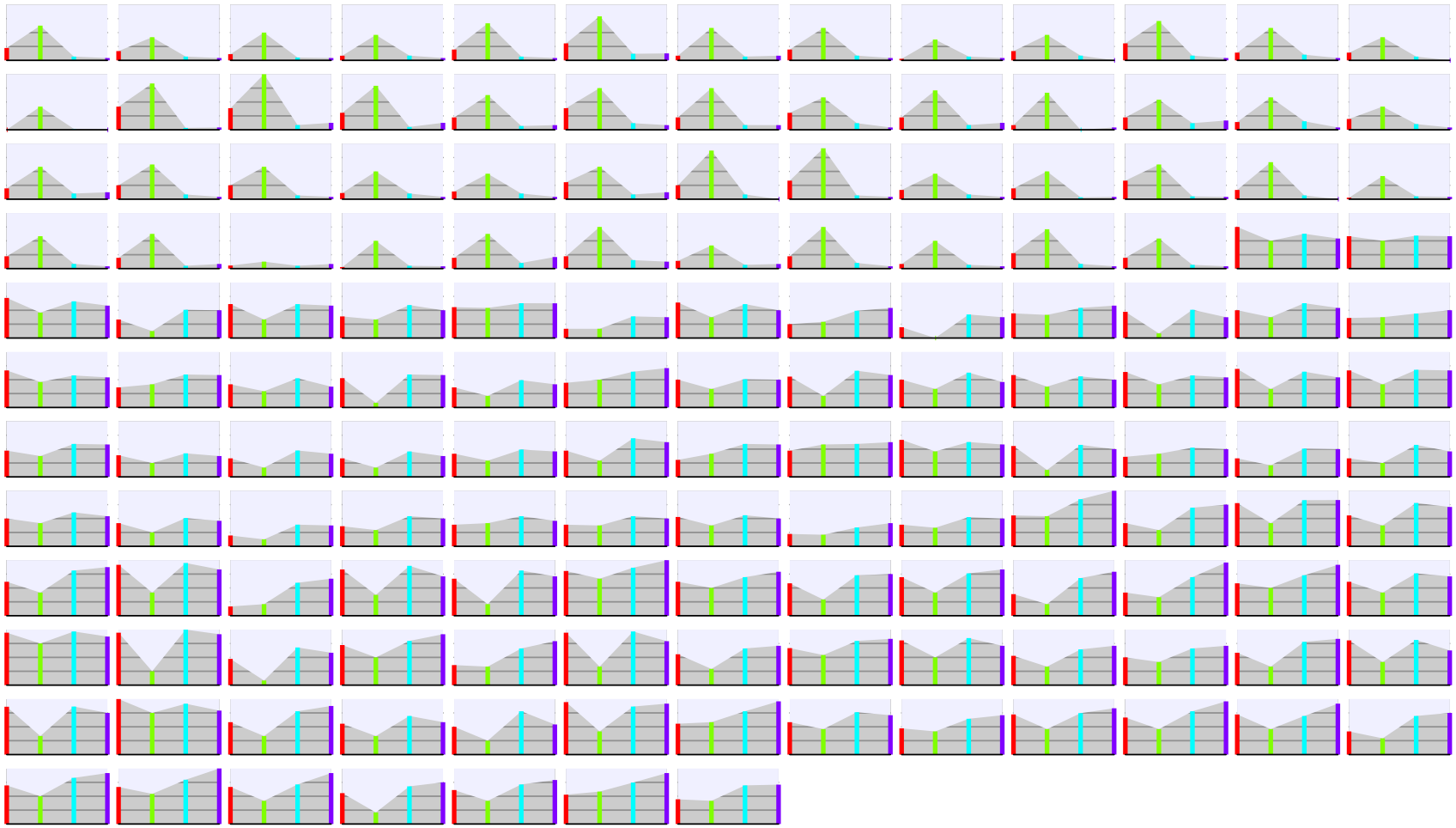


A plane; narrows (or becomes more parallel) more the closer the projection is to edge on.

Parallel axes glyphs

From `mountains.R` (adapted from code by Dan Carr)

```
data <- iris[,1:4]
mountains(data)
```



In `loon`, both radial axes and parallel axes are special cases of **serial axes** plots:

```
library(loon)
labels <- paste(olive[,"Region"], ":\n  ", olive[, "Area"])

l_serialaxes(oliveAcids, linkingGroup = "olive",
             itemLabel = labels, showItemLabels = TRUE,
             showAxesLabels = FALSE,
             axesLayout = "parallel")

l_plot(oliveAcids, linkingGroup = "olive",
       itemLabel = labels, showItemLabels = TRUE)
areas <- as.numeric(olive[,"Area"])
l_hist(areas, linkingGroup = "olive")
```

Example analysis

There is a vignette/article "Analysis: Visible minorities in Canadian cities" available online at https://great-northern-diver.github.io/loon/articles/minorities.html from `loon` which provides an interactive analysis demonstrating use of various serial axes plots.

High dimensional data as glyphs

What if the dimensionality is very large? Can we still have glyphs which show **all** dimensions?

One thought would be to use a square (or perhaps rectangular) "pixel" or "heatmap".

- every point in the high-dimensional space is represented by an array of tiny squares (or pixels)
- each tiny square (or pixel) is an element of the array and its value is that of a specific variate for that point
- each tiny square is coloured so that its colour encodes the value of the variate

In this way high dimensional data (hundreds and even thousands of variates) appear as a single picture glyph for each point.

N.B. Need to decide on the mapping of variates to elements of the array.

High dimensional data as glyphs

Some desirable features of the mapping of variates to array locations:

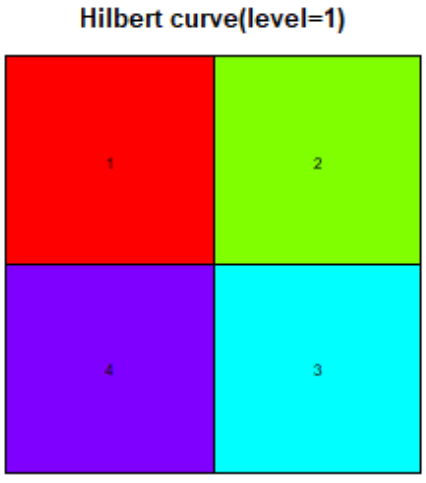
- order of the variates be preserved
- variates that are close to one another in order be close to one another in the array (e.g. especially desirable when each point is a regular time series)
- in the case where each point is an ordering time, say, then if the resolution of time becomes finer (e.g. from days to hours, or hours to minutes), then the positions in the array should not change much as the array becomes larger

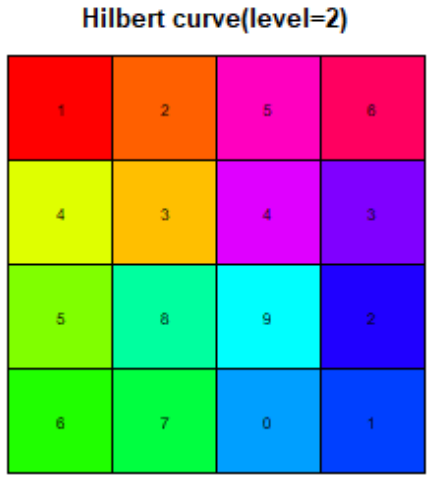
Proposed solutions: regular and recursive "space filling" curves

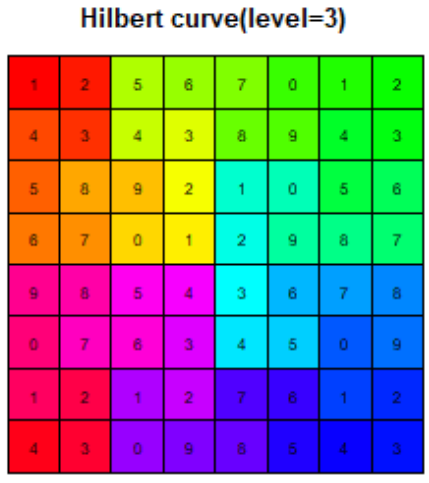
- Hilbert curve
- Morton curve
- Keim's recursive rectangles

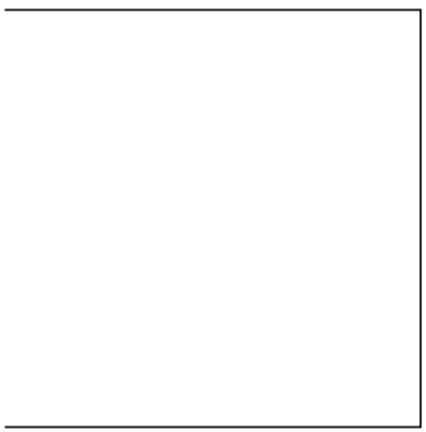
High dimensional data as glyphs

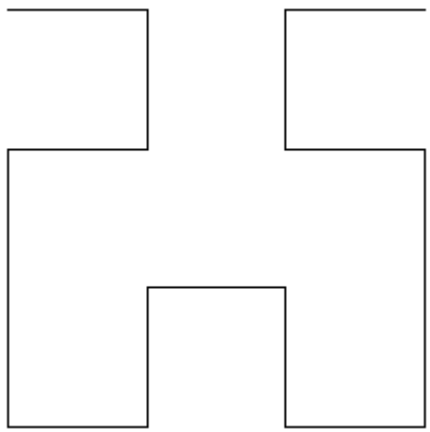
Hilbert:

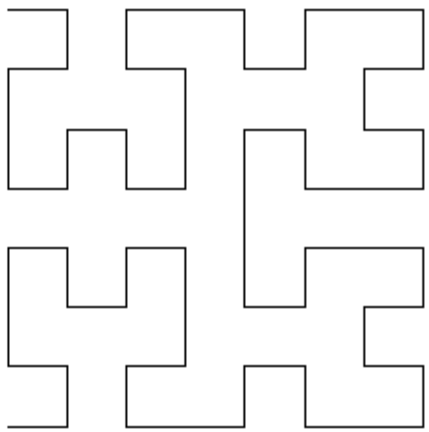








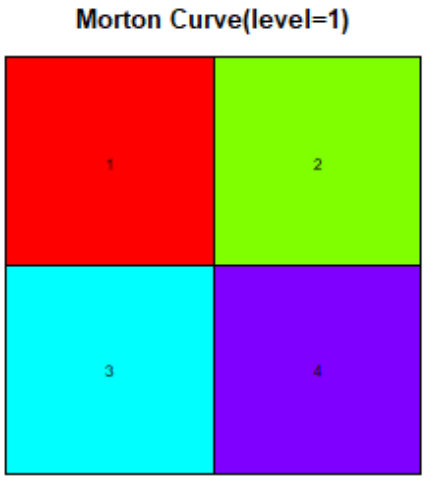


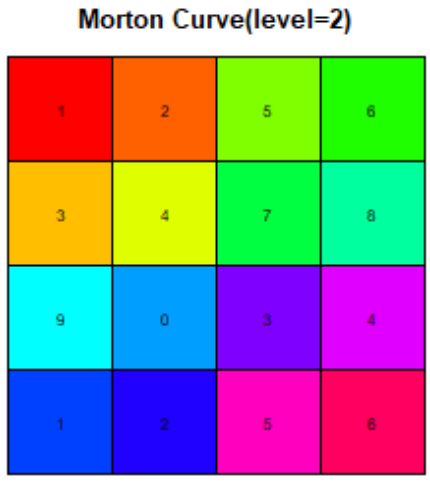


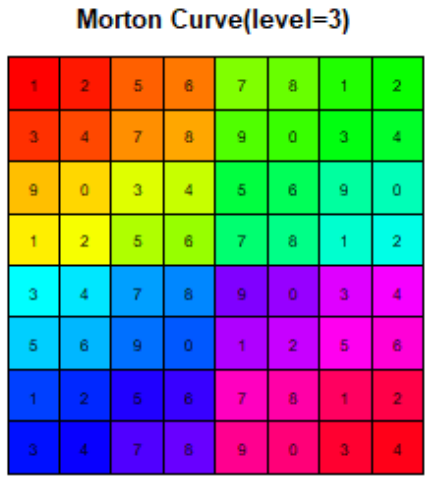
University of Waterloo

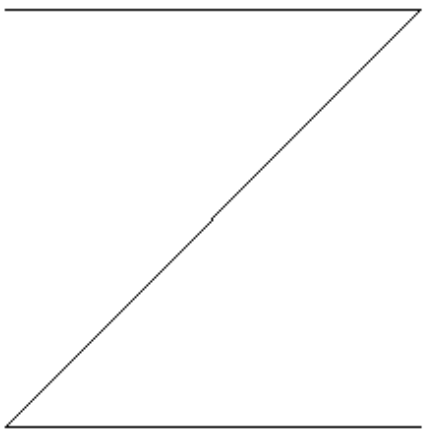
High dimensional data as glyphs

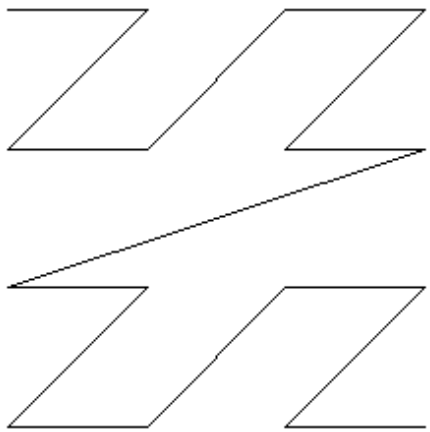
Morton:

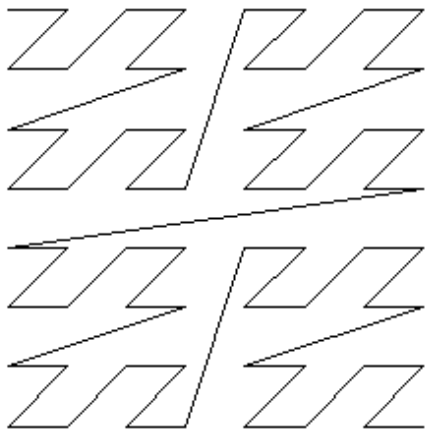








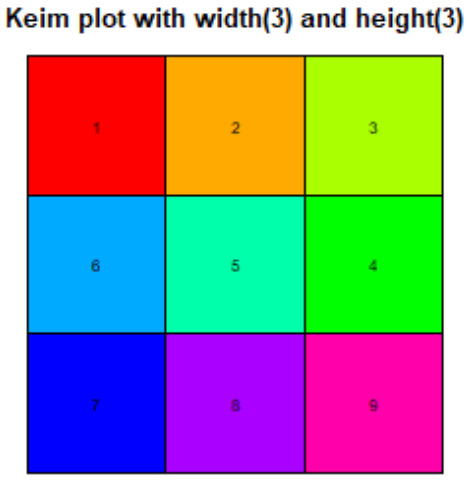


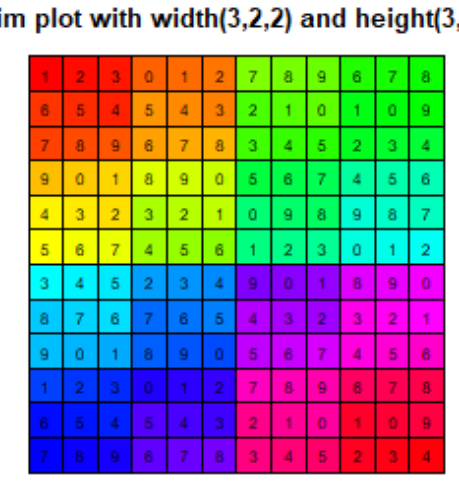


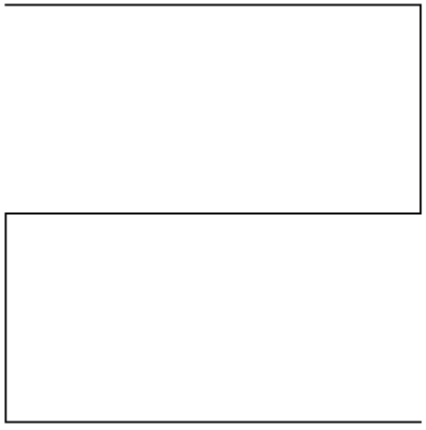
University of Waterloo

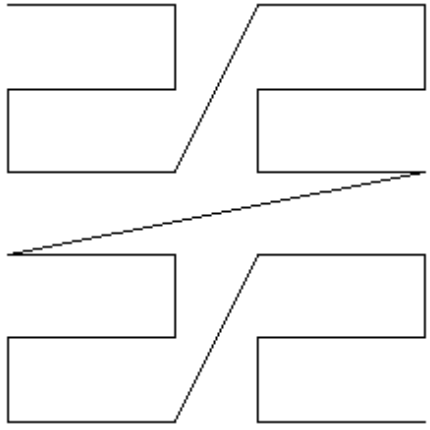
High dimensional data as glyphs

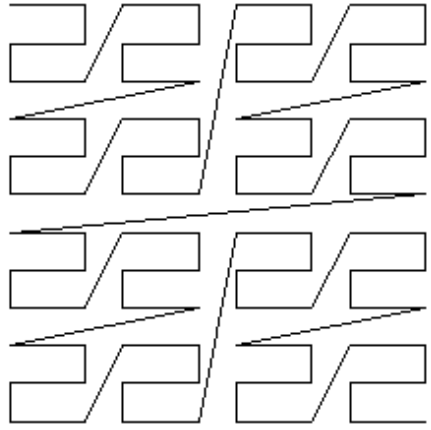
Keim:











University of Waterloo

High dimensional data as glyphs - S&P 500 data

For example, the SP500 data contains S&P 500 constituents stock indices from 1962, when at least one of the constituents is available, to 2015 and if the data is not available, it will return missing data.

The constituents information is in the following website

http://en.wikipedia.org/wiki/List_of_S%26P_500_companies.

The data set comes from `qrmdata` package in CRAN. See `help(SP500_const)` for more information.

```
library(qrmdata)
data("SP500_const") # load the constituents data from qrmdata
```

Here we will select three years of data around the financial crash of 2008.

```
time <- c("2007-01-03", "2009-12-31") # specify time period
data_sp500 <- SP500_const[paste0(time, collapse = "/"),] # grab out data
```

The data has dimensions `dim(data_sp500)` = 756, 505.

Each column is a "point"; it corresponds to one of the stocks that constitute the S&P 500 index. Each row corresponds to a trading day and is a "variate"; the total number of trading days is therefore the dimensionality of each point (stock). The variate values for each stock are the adjusted close price for the various trading days.

The data has 756 dimensions, each corresponding to a single trading day from January 3, 2007 to December 31, 2009. (Note: there are only 5 trading days per week.)

High dimensional data as glyphs - S&P 500 data

For example, some stock indices in the first **seven** days (from January **3**, 2007 to January **11**, 2007 inclusive) are listed below.

MMM	ABT	ABBV	ACN	ACE	ATVI	ADBE
61.93	18.41	NA	30.48	50.27	7.95	39.92
61.68	18.76	NA	31.16	49.61	8.02	40.82
61.26	18.76	NA	30.73	49.25	7.97	40.62
61.40	18.82	NA	31.17	49.28	7.97	40.45
61.47	18.99	NA	31.10	48.77	7.96	39.63
61.60	19.05	NA	31.28	48.76	8.13	39.22
62.24	19.04	NA	31.16	49.36	8.54	39.88

Each row is a dimension/variate/day and each column is headed by the "stock ticker" abbreviation for that stock (e.g. `ABT` is "Abbot Laboratories")

Note also that some data is missing (i.e. recorded as `NA`). We'll remove all stocks with missing data from consideration, and split it into a list of stocks (for the glyph-making functions) as follows:

```
cases <- complete.cases(t(data_sp500)) # identifies cases that have no NAs
x <- na.omit(t(data_sp500)) # omit the missing data
data_omitNA <- split(x,row(x)) # split the data into list
str(data_omitNA[1:3]) # present the first three stocks indices in the data list
```

```
## List of 3
##  $ 1: num [1:756] 61.9 61.7 61.3 61.4 61.5 ...
##  $ 2: num [1:756] 18.4 18.8 18.8 18.8 19 ...
##  $ 3: num [1:756] 30.5 31.2 30.7 31.2 31.1 ...
```

University of Waterloo

High dimensional data as glyphs - S&P 500 data

Moreover, the S&P 500 data also provides the sectors in which stocks belong (see `SP500_const_info`) . Below shows the sectors and subsectors of the first few stocks

```
knitr::kable(head(SP500_const_info, 7))
```

Ticker	Sector	Subsector
MMM	Industrials	Industrial Conglomerates
ABT	Health Care	Health Care Equipment & Services
ABBV	Health Care	Pharmaceuticals
ACN	Information Technology	IT Consulting & Other Services
ACE	Financials	Property & Casualty Insurance
ATVI	Information Technology	Home Entertainment Software
ADBE	Information Technology	Application Software

For example, we might look at stocks from a few different subsectors (n.b. use of cases).

```
bank_loc <- which(SP500_const_info$Subsector[cases] == "Banks")
pharma_loc <- which(SP500_const_info$Subsector[cases] == "Pharmaceuticals")
oil_loc <- which(SP500_const_info$Subsector[cases] == "Oil & Gas Exploration & Production")
dataSubsectors <- data_omitNA[c(bank_loc, pharma_loc, oil_loc)]
```

High dimensional data as glyphs - S&P 500 data

We could also calculate the averages over each sector as

```
nBanks <- length(bank_loc)
bank_ave <- Reduce("+", data_omitNA[bank_loc])/nBanks
nPharmas <- length(pharma_loc)
pharma_ave <- Reduce("+", data_omitNA[pharma_loc])/nPharmas
nOils <- length(oil_loc)
oil_ave <- Reduce("+", data_omitNA[oil_loc])/nOils
```

and add them to the data at the end:

```
nStocks <- length(dataSubsectors)
dataSubsectors[seq(nStocks+1, nStocks +3)] <- list(bank_ave, pharma_ave, oil_ave)
```

High dimensional data as glyphs - S&P 500 data

Now we can begin to construct some glyphs (blue is high, red low value, grey is average).

```
library(colorspace)
cols <- rev(diverge_hcl(21))  # diverge from red to blue
```

and use the `glyphs` package. First we need to make sure that we are using the correct `make_glyphs()` (namely the one from `glyphs` and not `loon`).

```
unloadNamespace("glyphs")
library("glyphs")
```

The glyphs can now be created using the `make_glyphs` function from the `glyphs` package.

The `glyphs` package is not (yet) on CRAN. It is available from the course website or https://github.com/rwoldford/glyphs. (You can install this from the shell/terminal with "R CMD")

High dimensional data as glyphs - S&P 500 data

Create all three types of encodings: Hilbert, Morton, and Keim's recursive rectangles. First Hilbert and Morton:

```
SP500Hilbert <- make_glyphs(dataSubsectors,
                            glyph_type = "Hilbert",
                            origin = "mean", col=cols)

SP500Morton <- make_glyphs(dataSubsectors,
                           glyph_type = "Morton",
                           origin = "mean", cols = cols)
```

with the Keim recursive rectangle, `height` and `width` parameters can be used to effect a layout that is more natural to this problem.

High dimensional data as glyphs - S&P 500 data

With the Keim recursive rectangle, `height` and `width` parameters can be used to effect a layout that is more natural to this problem.

```
width  = c(5, 1, 12, 1) # set the widths
height = c(1, 4,  1, 3)  # and the heights for rectangles
```

organizes weekdays into a week (5 columns of 1 row)

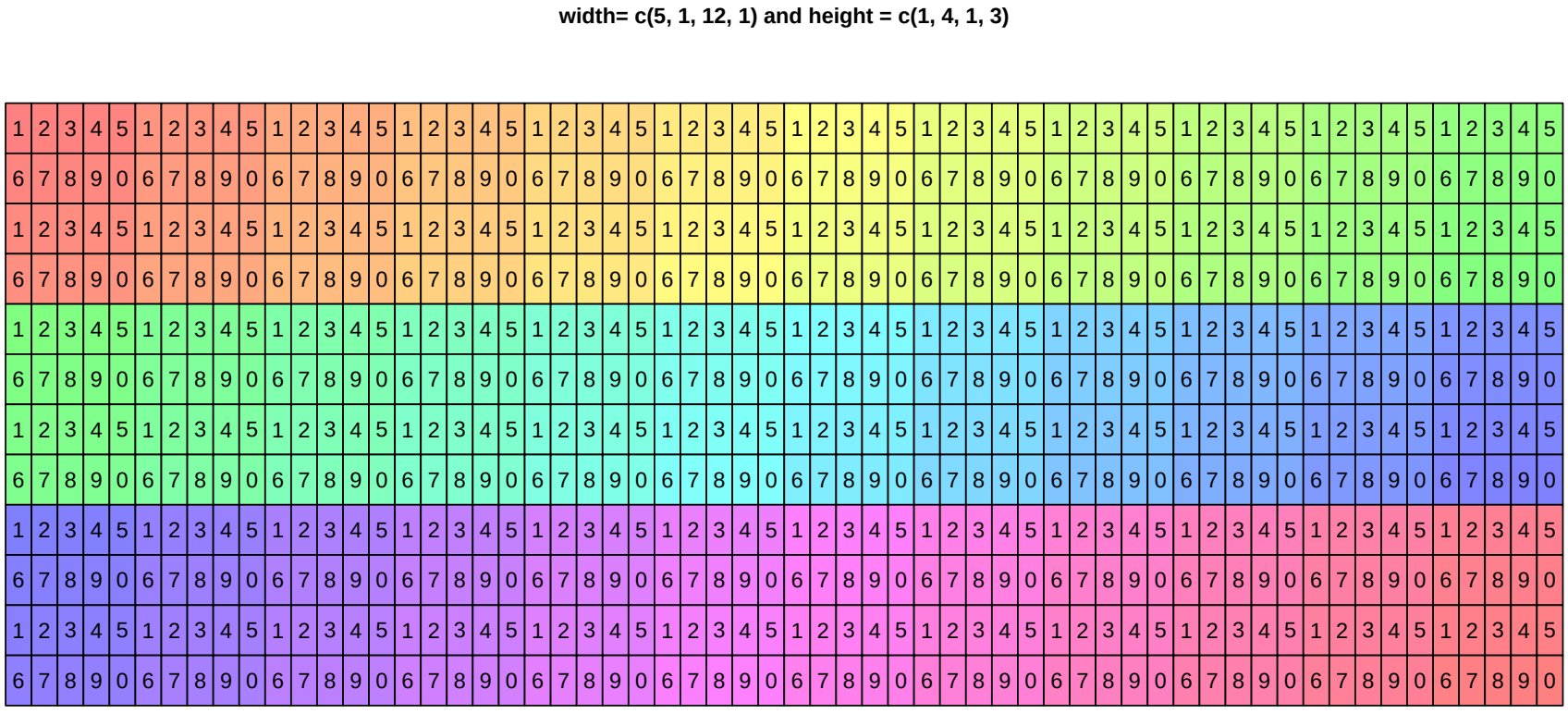
and weeks into a month (1 column of 4 rows)

and months into a year (12 columns of 1 row)

and years into separate rows (1 column of 3 rows)

High dimensional data as glyphs - S&P 500 data

The Keim recursive rectangle with `width = c(5, 1, 12, 1), height = c(1, 4, 1, 3)`



High dimensional data as glyphs - S&P 500 data

So, we'll use Keim recursive rectangle with `width = c(5, 1, 12, 1)`, `height = c(1, 4, 1, 3)` for the S&P 500 data.

```
# With Keim's recursive rectangle we can
# approximate weeks, months, years
SP500Keim <- make_glyphs(dataSubsectors,
                         glyph_type = "rectangle",
                         width = width, height = height,
                         origin = "mean", cols = cols)
```

University of Waterloo

High dimensional data as glyphs - S&P 500 data

Plot these glyphs using this little ad hoc function

```
doit <- function (glyphs, main ="", labels = NULL, labelCol = "grey30") {
  x <- getGridXY(length(glyphs)) # get coordinates for each glyph
  plot_glyphs(x, glyphs = glyphs, axes = FALSE, xlab = "", ylab = "",
              glyphWidth = 0.8, glyphHeight = 0.6,
              main = main, cex.main = 0.8)
  if (!is.null(labels)) text(x, labels = labels, col = labelCol)
}
```

Then to look at the individual stocks via Hilbert encoding, simply execute

```
doit(SP500Hilbert[1:nStocks])
```

High dimensional data as glyphs - S&P 500 data

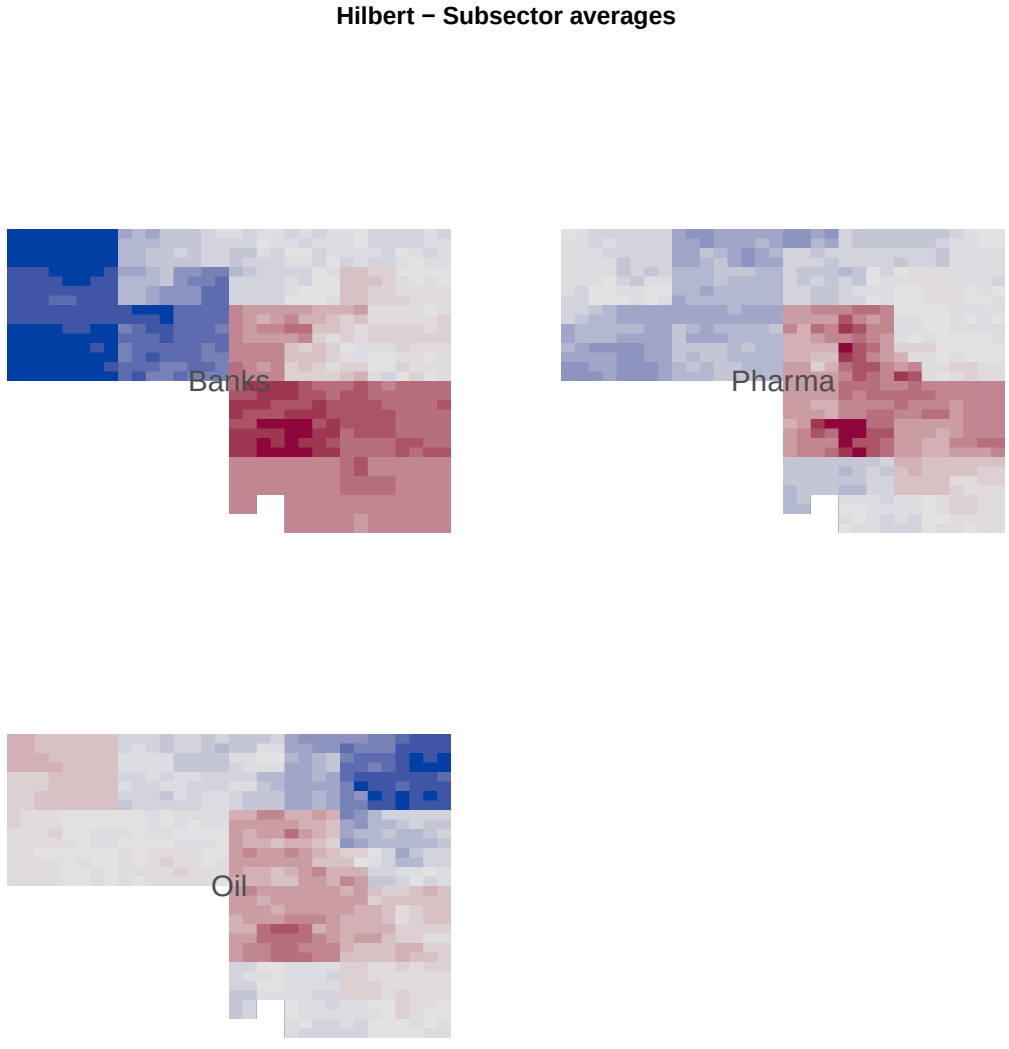
To select the different sectors and averages, a number of indices will be handy

```
ave_indices <- (nStocks+1):(nStocks+3)
bank_indices <- 1:nBanks
pharma_indices <- (nBanks+1):(nBanks+nPharmas)
oil_indices <- (nBanks + nPharmas +1):nStocks
```

We now examine these for each encoding.

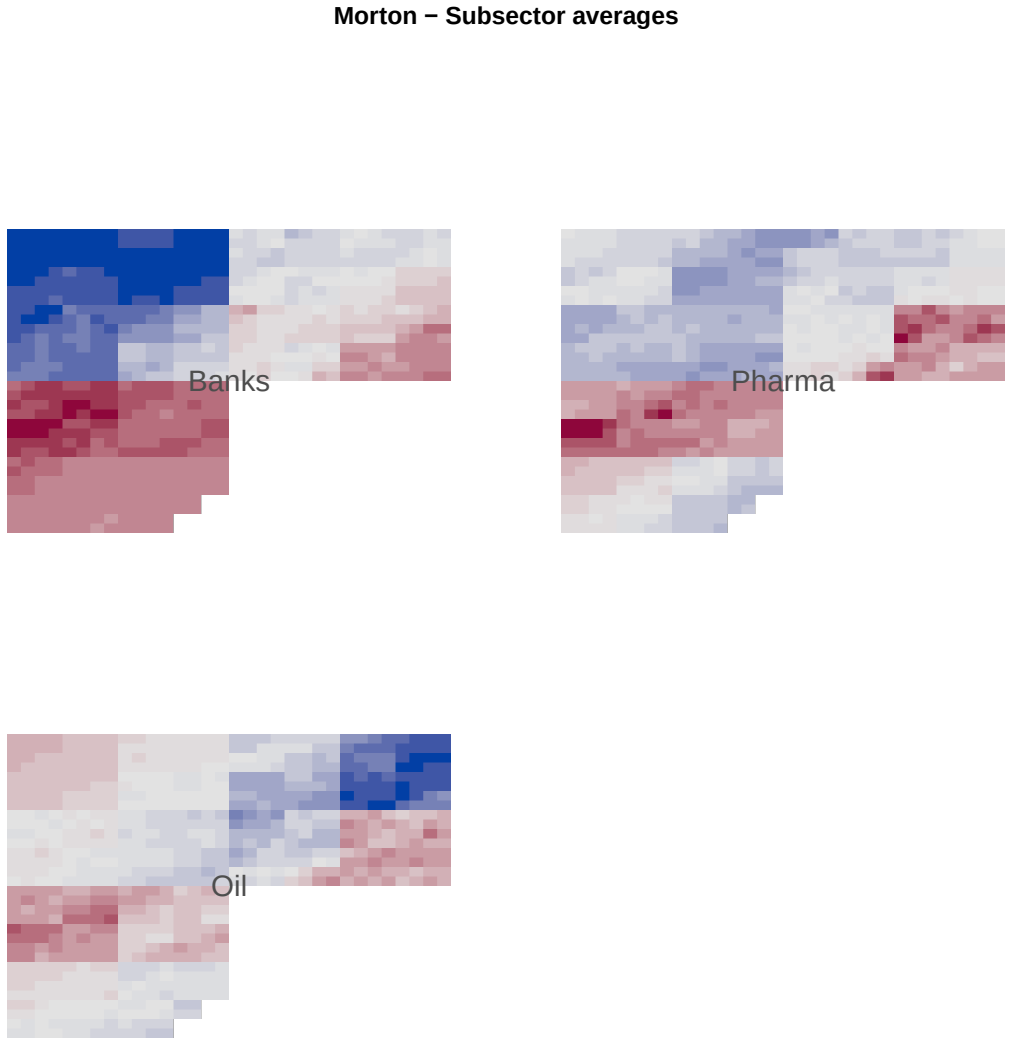
High dimensional data as glyphs - S&P 500 data

```
doit(SP500Hilbert[ave_indices], main = "Hilbert - Subsector averages",
     labels = c("Banks", "Pharma", "Oil"))
```



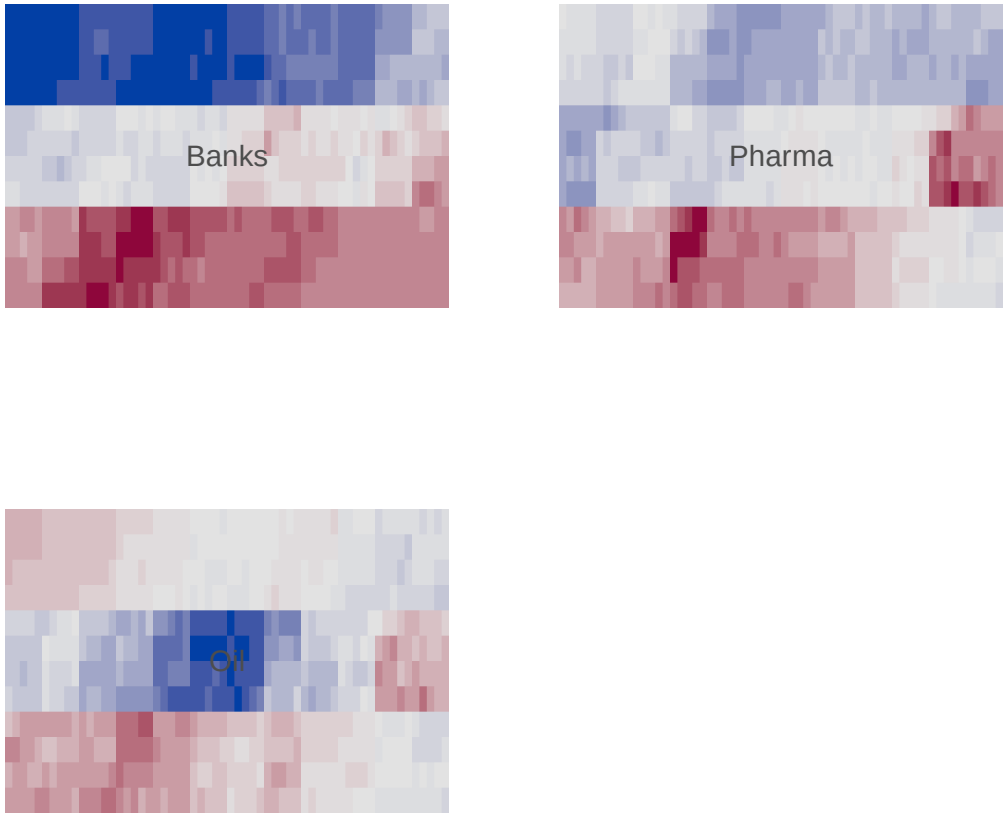
High dimensional data as glyphs - S&P 500 data

```
doit(SP500Morton[ave_indices], main = "Morton - Subsector averages",
     labels = c("Banks", "Pharma", "Oil"))
```



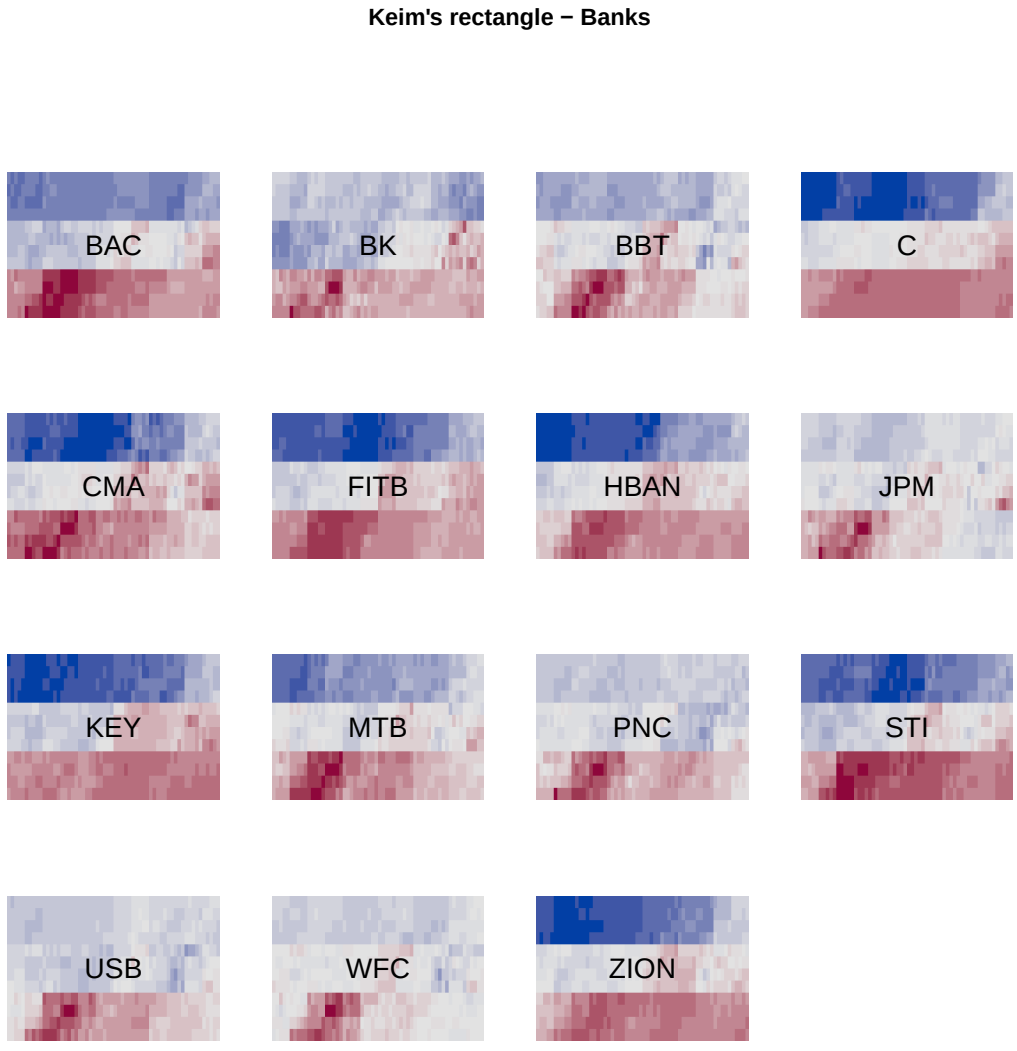
High dimensional data as glyphs - S&P 500 data

```
doit(SP500Keim[ave_indices], main = "Keim's rectangle - Subsector averages",
     labels = c("Banks", "Pharma", "Oil"))
```



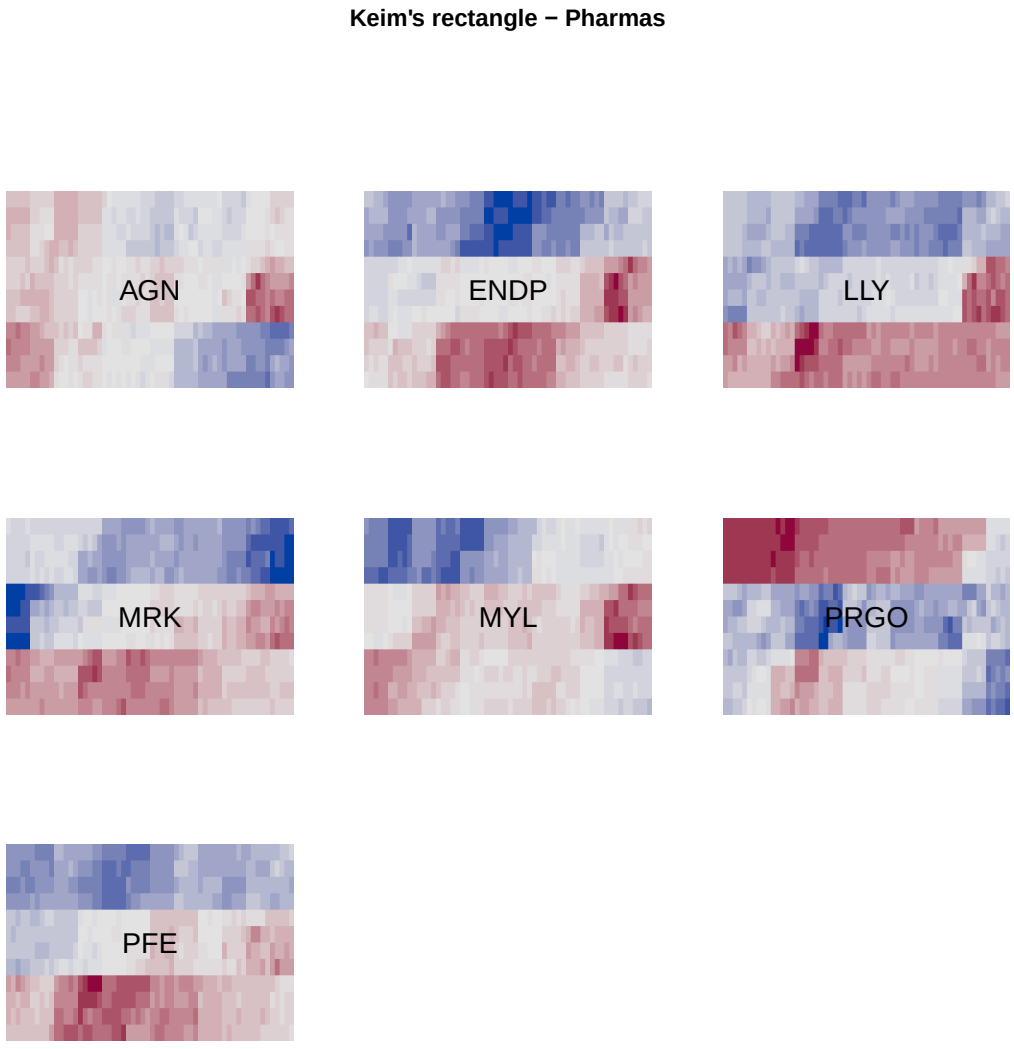
High dimensional data as glyphs - S&P 500 data

```
doit(SP500Keim[bank_indices], main = "Keim's rectangle - Banks",
     labels = as.vector(SP500_const_info[cases,][bank_loc,]$Ticker), labelCol="black")
```



High dimensional data as glyphs - S&P 500 data

```
doit(SP500Keim[pharma_indices], main = "Keim's rectangle - Pharmas",
     labels = as.vector(SP500_const_info[cases,][pharma_loc,]$Ticker), labelCol="black")
```



High dimensional data as glyphs - S&P 500 data

```
doit(SP500Keim[oil_indices], main = "Keim's rectangle - Oil",
     labels = as.vector(SP500_const_info[cases,][oil_loc,]$Ticker), labelCol="yellow")
```