E01

漫談巨量資料於R軟體中的統計圖與視覺化

吳漢銘
國立政治大學 統計學系

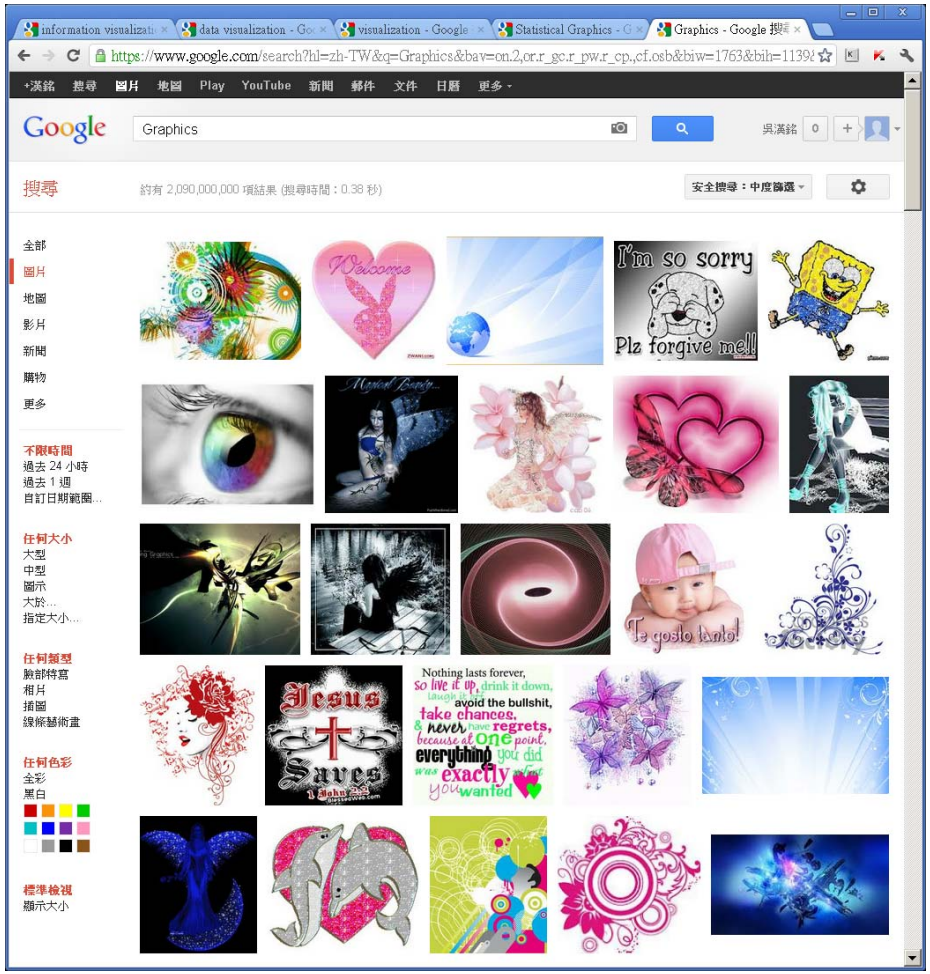
https://hmwu.idv.tw

大綱

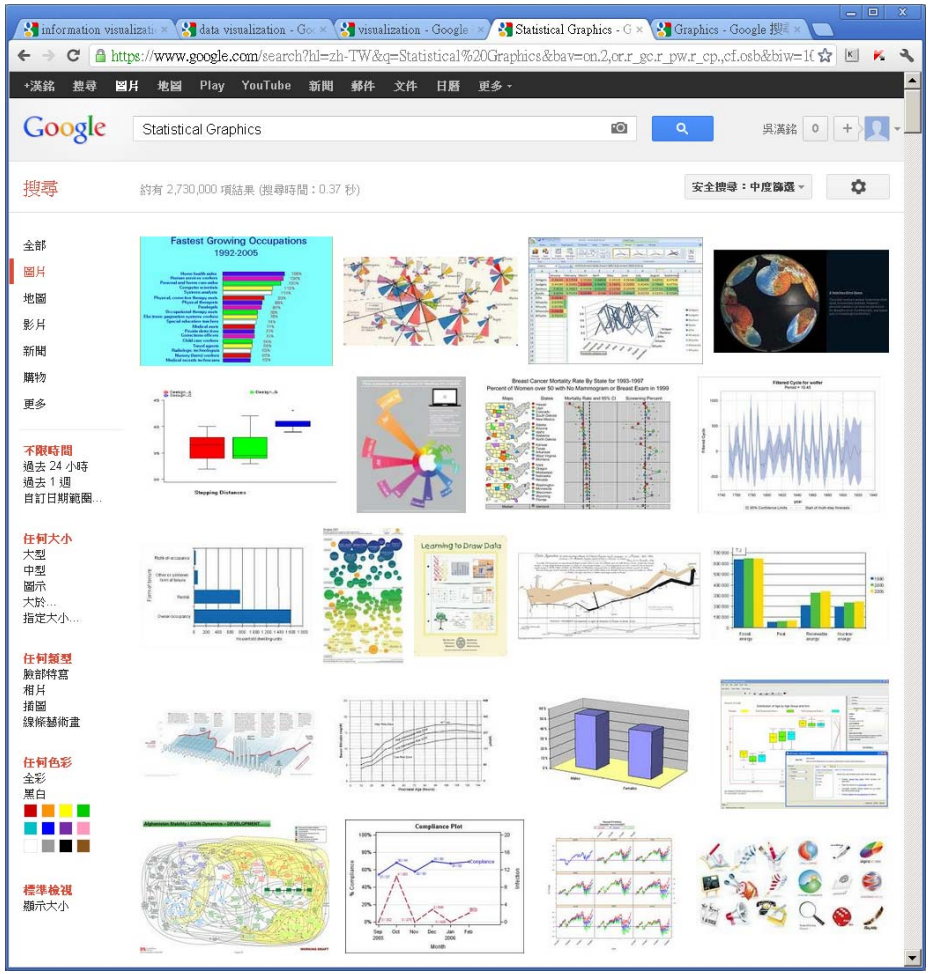
- What is Visualization? Why Data Visualization?
- Graphics Systems: **graphics**, **lattice**, **ggplot2**.
- Big Data Visualization, The Challenge of Visualizing Big Data
- How to Visualize Billion+ Records
  - Bin => Aggregate => Smooth => Plot
- R Functions: **smoothScatter {graphics}, heatscatter {LSD}, hist2d {gplots}, kde2d {MASS}**.
- R Packages: **hexbin, bigvis, tabplot, ggplot2.SparkR, graphics.SDA**

Graphics

Graphics

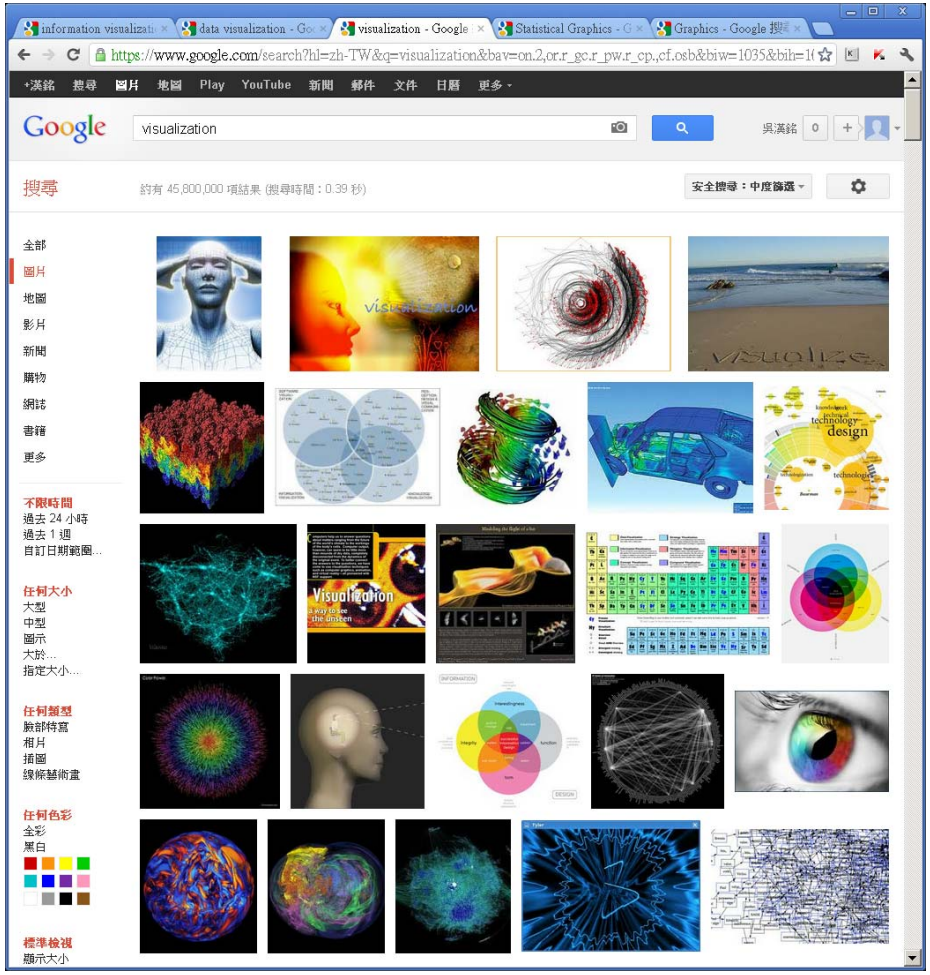


Statistical Graphics

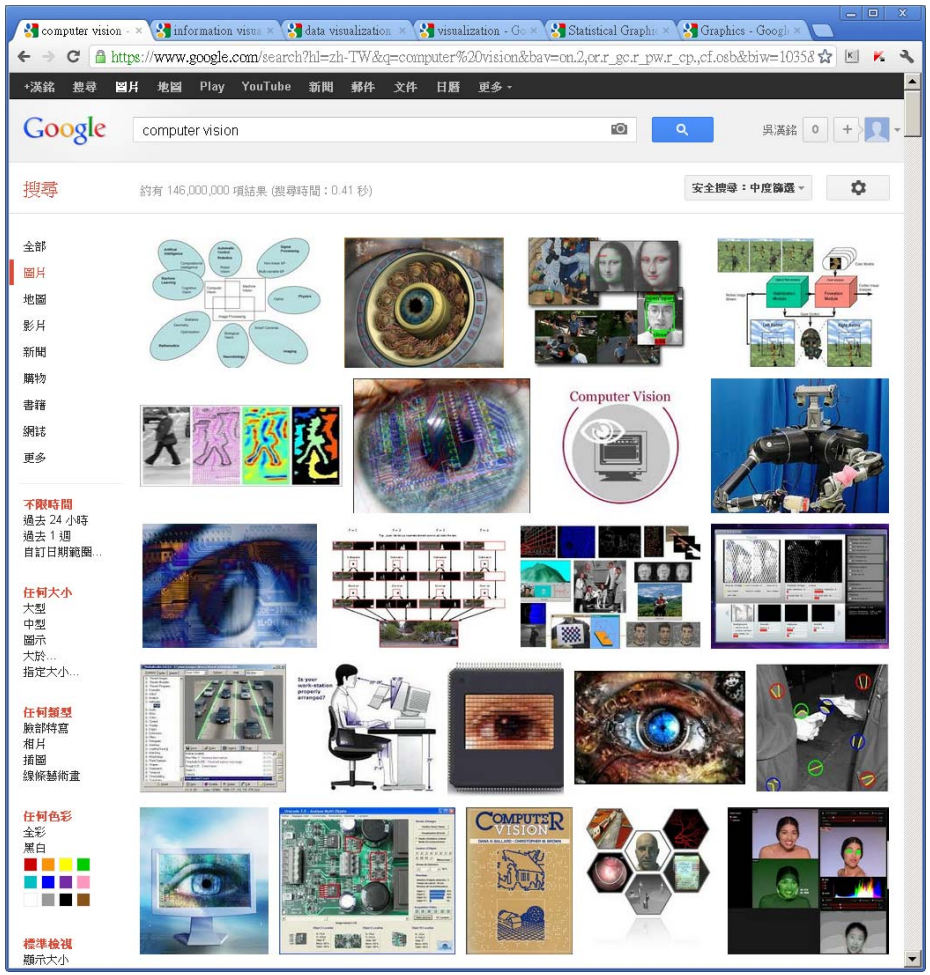


Visualization

Visualization

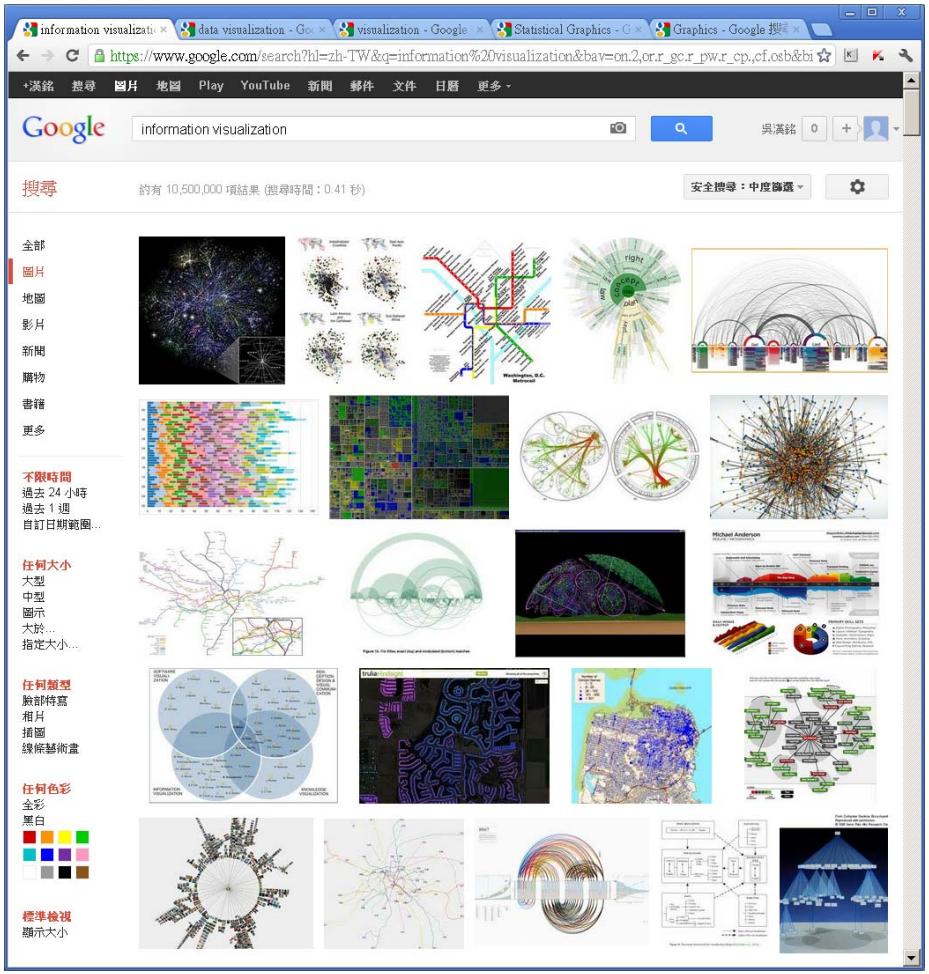


Computer Vision

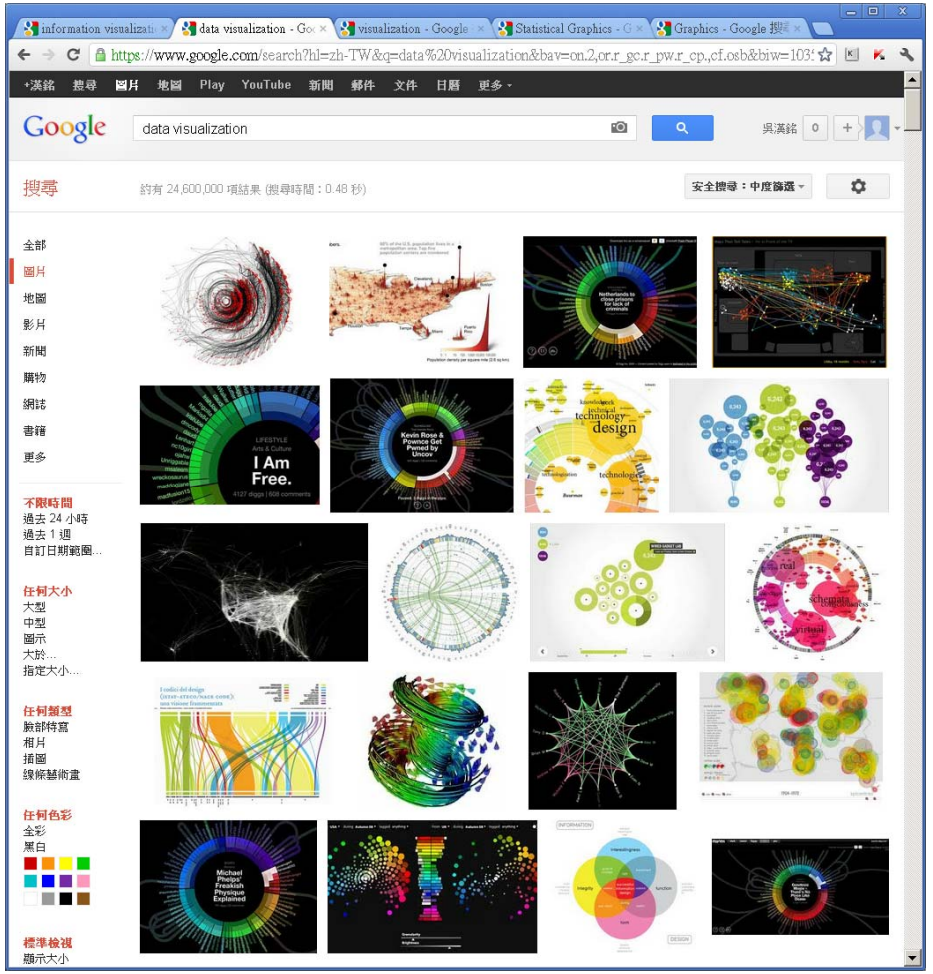


Data Visualization

Information Visualization



Data Visualization



What is Visualization?

People said

- Seeing is believing.
  (眼見為憑)
- Seeing is better than hearing a hundred times.
  (百聞不如一見)
- A picture is worth a thousand words.
  (一幅圖像勝過千言萬語)

The longest name of a city in New Zealand.

What is visualization?

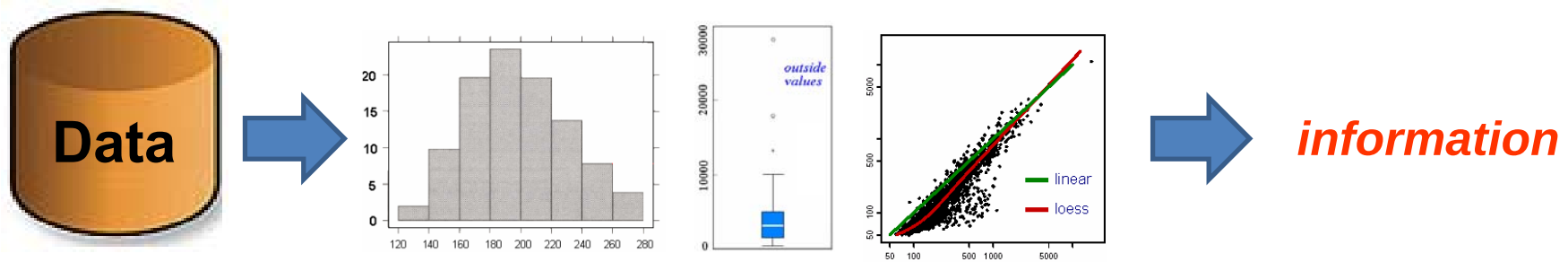
- Making things/processes/abstractions visible (to transform into pictures) that are not directly accessible by the human eye.
- Computer aided extraction and display of information from data.

The shortest city name in the world is in Norway with one letter (A).

Picture Source:
http://pictzz.blogspot.com/2009/03/pictures-is-worth-thousand-words.html

Graphical Methods

- **The purpose of statistical graphics is** to provide visual representations of quantitative information.



Visualization = Graphing for Data + Fitting + Graphing for Model

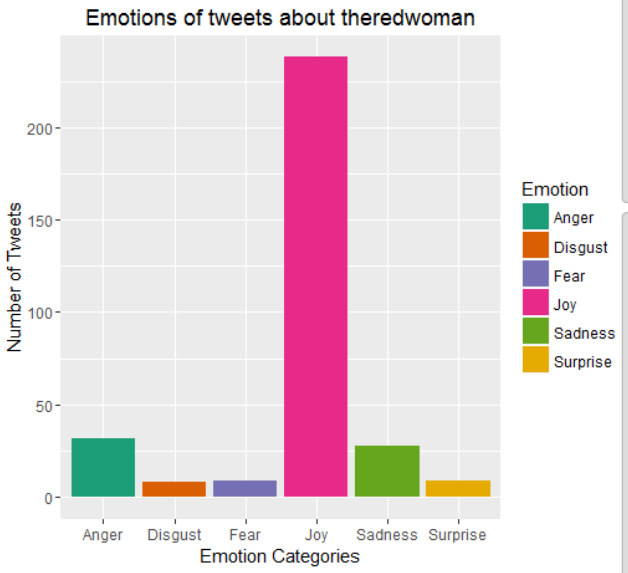
- **Statistical graphics comprise** a set of strategies and techniques that provide the research with important insights about the data under examination and help guide the subsequent steps of the research process.
- ***Exploratory Data Analysis (EDA) Tool***
  - Summaries for large, complicated data sets.
  - Revealing structure, patterns, features, trends, outliers, anomalies, and relationships in data .
  - Extract important variables.
  - Checking assumptions in statistical models.
  - Interaction between the researcher and the data.
  - Identifying the areas of interest.

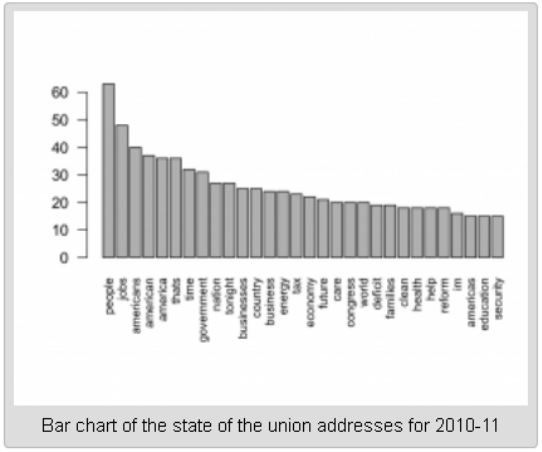
Infovis and Statistical Graphics: Different Goals, Different Looks

Journal of Computational and Graphical Statistics, Volume 22, 2013 - Issue 1

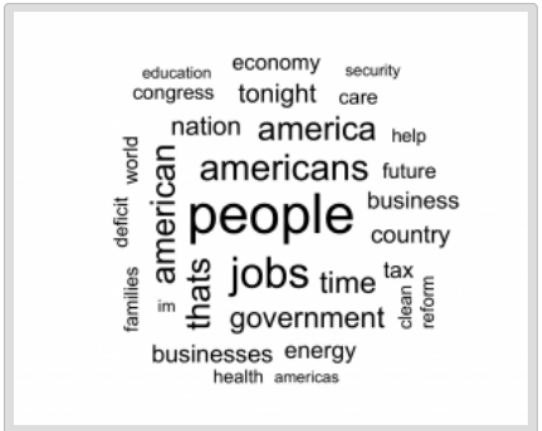
- **Infovis and Statistical Graphics: Different Goals, Different Looks**
  Andrew Gelman & Antony Unwin, Pages: 2-28
- **InfoVis Is So Much More**, Robert Kosara, Pages: 29-32
- **InfoVis and Statistical Graphics: Comment**
  Paul Murrell, Pages: 33-37
- **Graphical Criticism: Some Historical Notes**
  Hadley Wickham , Pages: 38-44
- **Tradeoffs in Information Graphics**
  Andrew Gelman & Antony Unwin , Pages: 45-49

http://emarketingwall.com/how-twitter-responded-to-the-latest-episode-of-game-of-thrones





Bar chart of the state of the union addresses for 2010-11



word cloud of the state of the union addresses for 2010-11

https://www.r-bloggers.com/words-in-politics-some-extensions-of-the-word-cloud/

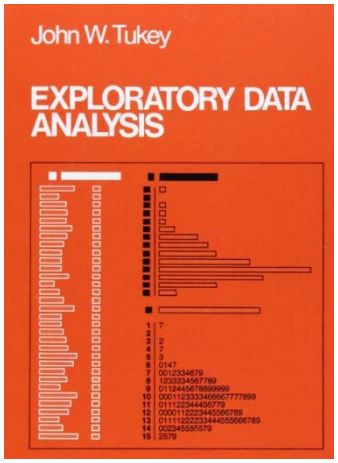
探索式資料分析
Exploratory Data Analysis, EDA

John Tukey (1915~2000) (統計學界的畢卡索)

「對正確的問題有個近似的答案，

勝過對錯的問題有精確的答案。」

"An approximate answer to the right question is worth a great deal more than a precise answer to the wrong question."



- Summaries for large, complicated data sets.
- Revealing structure, patterns, features, trends, outliers, anomalies, and relationships in data .
- Extract important variables.
- Checking assumptions in statistical models.
- Interaction between the researcher and the data.
- Identifying the areas of interest.

Visualization = Graphing for Data + Fitting + Graphing for Model

Why Data Visualization?

- It is not about "**infographics**", the beautiful, heavily customized products of expert graphic designers.
- Data visualization can provide clear understanding of patterns in data, detect hidden structures in data, condense information.
- **Anscombe's quartet** comprises four datasets. They were constructed in 1973 by the statistician Francis Anscombe to demonstrate both the importance of graphing data before analyzing it and the effect of outliers on statistical properties.
- Four datasets have nearly identical simple statistical properties, yet appear very different when graphed.

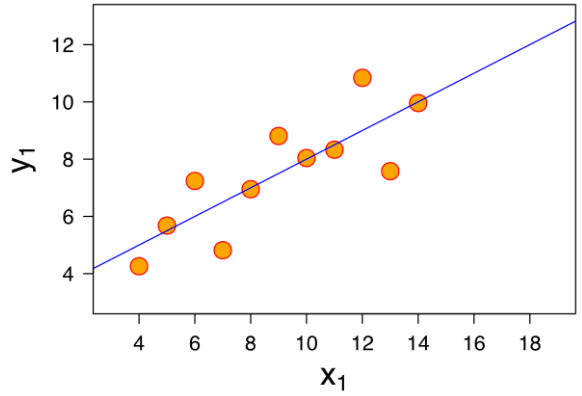
	I		II		III		IV	
	x	*y*	*x*	*y*	*x*	*y*	*x*	*y*
1	10	8.04	10	9.14	10	7.46	8	6.58
2	8	6.95	8	8.14	8	6.77	8	5.76
3	13	7.58	13	8.74	13	12.74	8	7.71
4	9	8.81	9	8.77	9	7.11	8	8.84
5	11	8.33	11	9.26	11	7.81	8	8.47
6	14	9.96	14	8.1	14	8.84	8	7.04
7	6	7.24	6	6.13	6	6.08	8	5.25
8	4	4.26	4	3.1	4	5.39	19	12.5
9	12	10.84	12	9.13	12	8.15	8	5.56
10	7	4.82	7	7.26	7	6.42	8	7.91
11	5	5.68	5	4.74	5	5.73	8	6.89

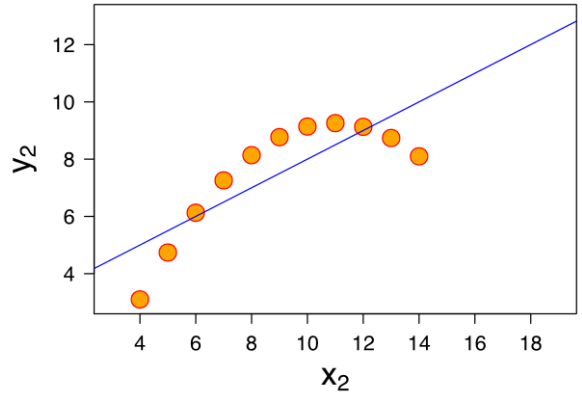
https://en.wikipedia.org/wiki/Anscombe's_quartet

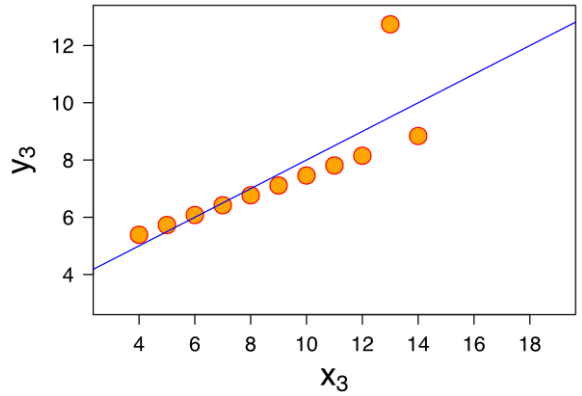
http://ryanwomack.com/IASSIST/DataViz/

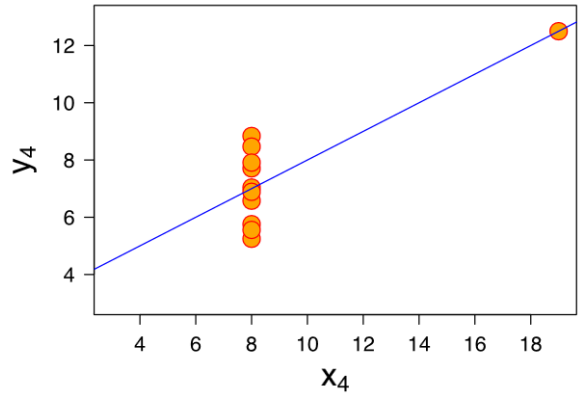
Anscombe's Quartet

- **Mean of x** in each case: 9 (exact)
- **Sample variance of x** in each case: 11 (exact)
- **Mean of y** in each case: 7.50 (to 2 decimal places)
- **Sample variance of y** in each case: 4.122 or 4.127 (to 3 decimal places)
- **Correlation** between x and y in each case: 0.816 (to 3 decimal places)
- **Linear regression line** in each case: y = 3.00 + 0.500x (to 2 and 3 decimal places, respectively)









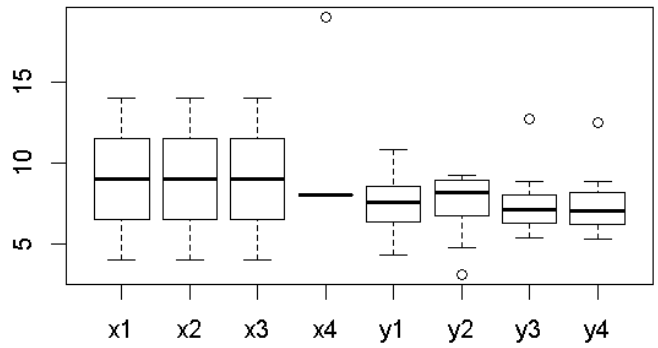
anscombe {datasets}

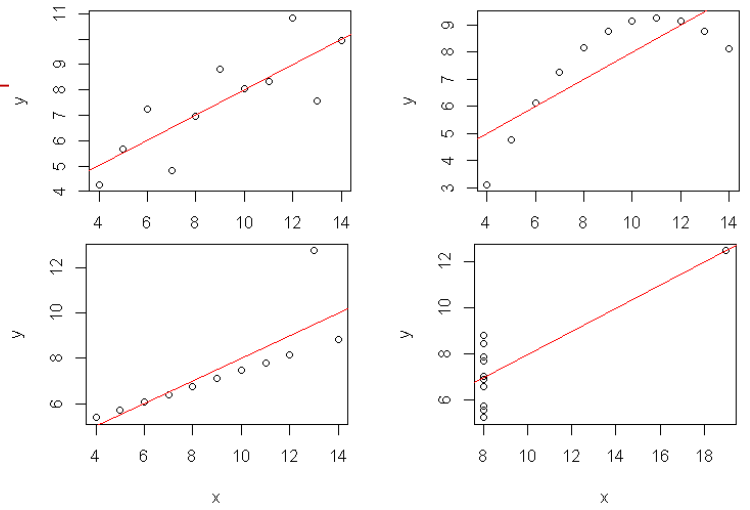
Anscombe's Quartet of 'Identical' Simple Linear Regressions

```
> head(anscombe, 3)
   x1 x2 x3 x4    y1   y2    y3    y4
1  10 10 10  8  8.04 9.14  7.46  6.58
2   8  8  8  8  6.95 8.14  6.77  5.76
3  13 13 13  8  7.58 8.74 12.74  7.71
> apply(anscombe, 2, mean)
      x1       x2       x3       x4       y1       y2       y3       y4
9.000000 9.000000 9.000000 9.000000 7.500909 7.500909 7.500000 7.500909
> apply(anscombe, 2, sd)
      x1       x2       x3       x4       y1       y2       y3       y4
3.316625 3.316625 3.316625 3.316625 2.031568 2.031657 2.030424 2.030579
> mapply(cor, anscombe[,1:4], anscombe[,5:8])
       x1        x2        x3        x4
0.8164205 0.8162365 0.8162867 0.8165214
> mapply(function(x, y) lm(y~x)$coefficients, anscombe[, 1:4], anscombe[, 5:8])
                 x1       x2        x3        x4
(Intercept) 3.0000909 3.000909 3.0024545 3.0017273
x           0.5000909 0.500000 0.4997273 0.4999091
```

```
par(mfrow=c(2, 2))
regplot <- function(x, y){
  plot(y~x)
  abline(lm(y~x), col="red")
}
mapply(regplot, anscombe[, 1:4], anscombe[, 5:8])
```

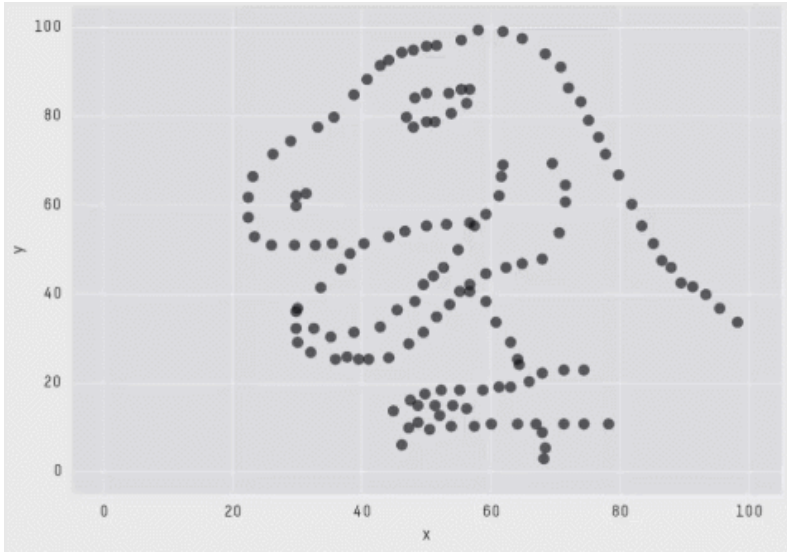
boxplot(anscombe)



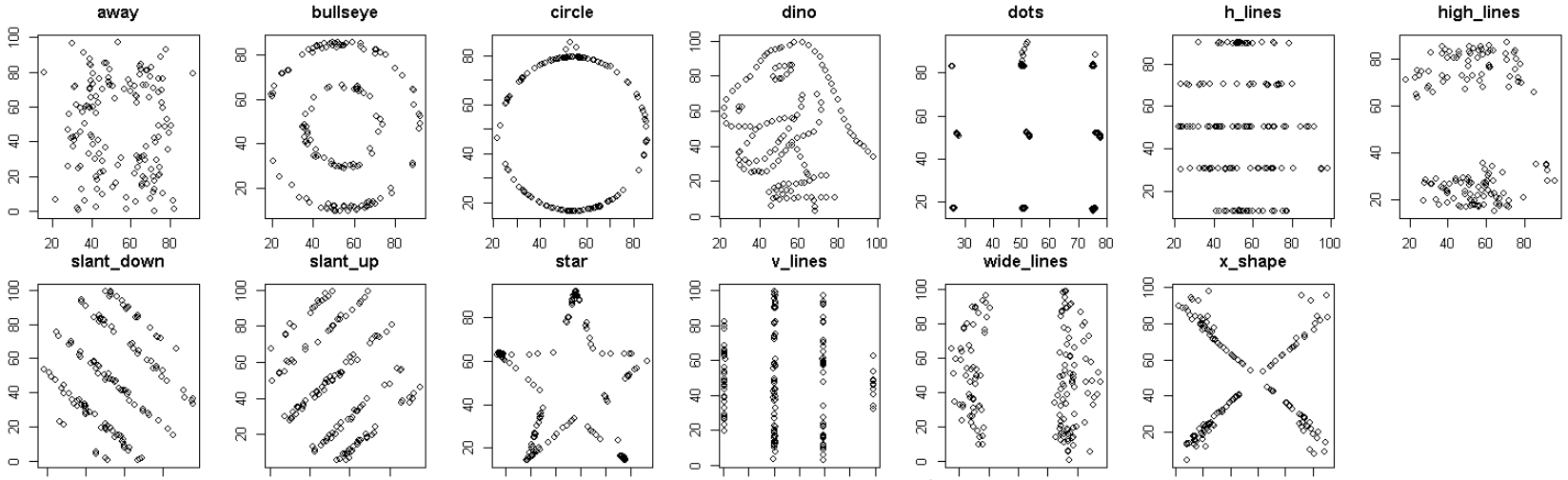


The Datasaurus Dozen

```
install.packages("datasauRus")
```

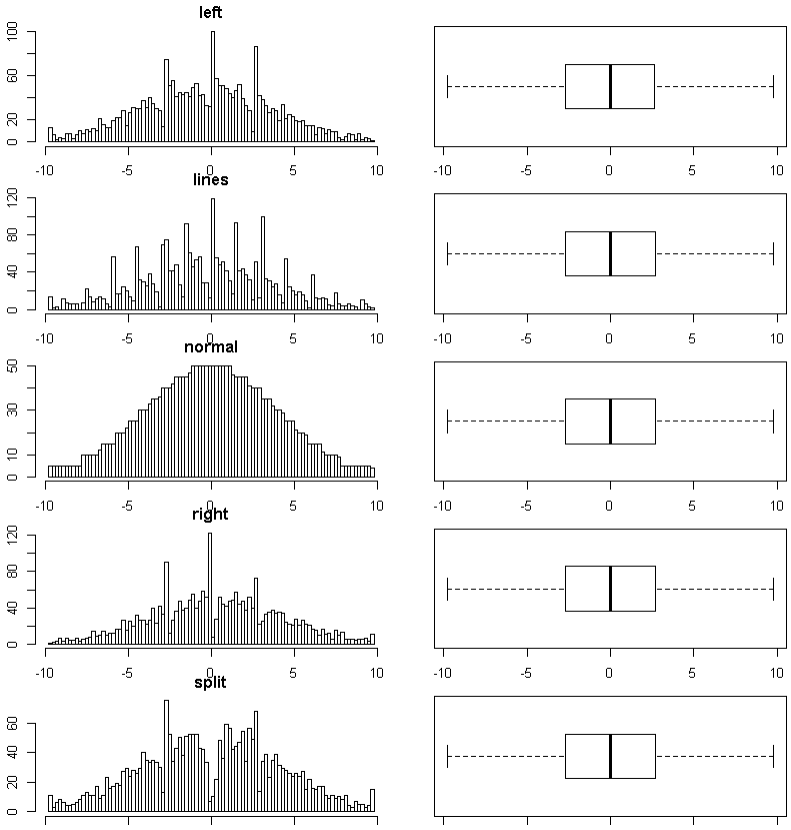


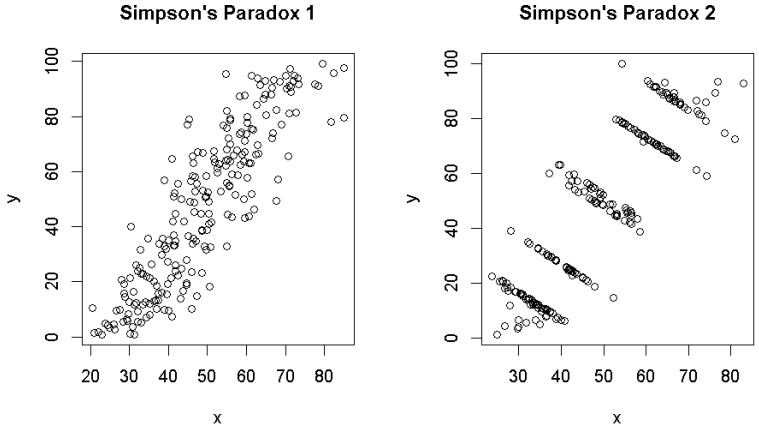
Justin Matejka and George Fitzmaurice, Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. https://www.autodeskresearch.com/publications/samestats



See also: https://www.r-bloggers.com/data-fun-inspired-by-darasaurus/

The Datasaurus Dozen
More examples



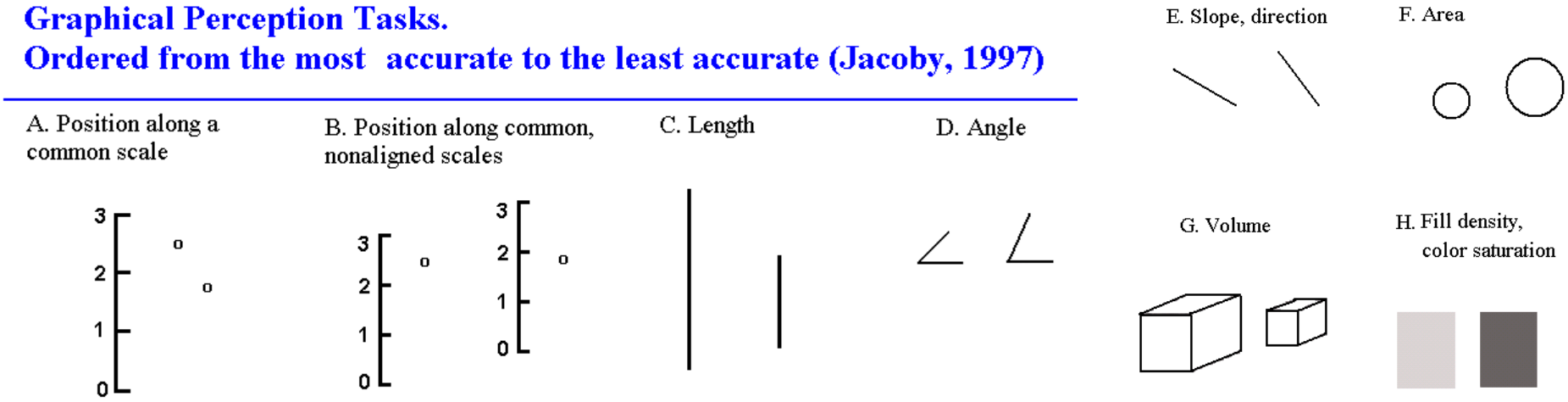


Graphical Perception

Human reception and comprehension of graphical information involves three fundamental perceptual task:

- Detection: the visual recognition of a geometric aspect that encodes a physical value. The basic information from the data must be discernible in the graph.
- Assembly: the process of discerning patterned regularities among the discrete elements of a graphical display.
- Estimation: the visual assessment of the relative magnitudes of two or more quantitative physical values.

Graphical Perception Tasks.
Ordered from the most accurate to the least accurate (Jacoby, 1997)



CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

- Plotting: `plotrix, vcd, hexbin, gclus, gplots, aplpack, lattice, scatterplot3d, misc3d, onion.`
- Graphic Applications:
  - Effect ordering: `gclus*, cba, seriation, biclust.`
  - Large Data Sets: `ash, hexbin*, scagnostics.`
  - Trees and Graphs: `ade4, ape, igraph, diagram, Rgraphviz, igraph.`
- Graphics Systems: `lattice*, ggplot2.`
- Devices: `cairoDevice, RGtk2, RSvgDevice, rgl, JavaGD.`
- Colors: `colorspace, vcd*, RColorBrewer, dichromat.`
- Interactive Graphics: `rggobi, iplots, JavaGD*, playwith, cairoDevice*, RGtk2*, rgl*.`
- Development: `rgl*, gridBase.`
- Others: `animation, Cairo, IDPmisc, klaR, latticeExtra, RGraphics, RSVGTipsDevice, tkrplot, vioplot, xgobi.`

`ash`: averaged shifted histogram
`hexbin`: Hexagonal Binning Routines
`scagnostics`: scatterplot diagnostics

plotly: Create Interactive Web Graphics via 'plotly.js'
https://plot.ly/r/

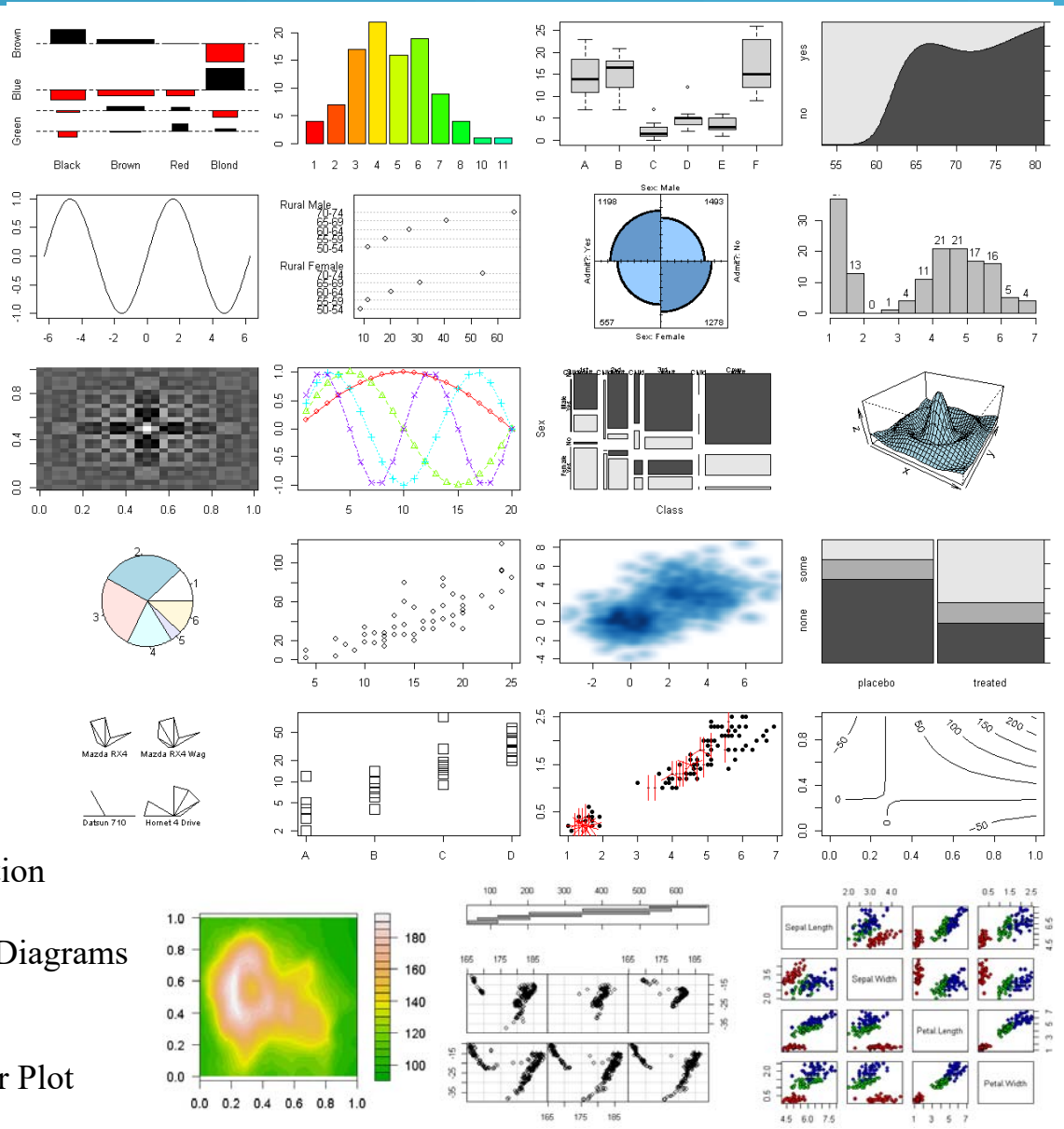
(Version: 2015-01-07)

http://cran.r-project.org/web/views/Graphics.html

graphics: The R Graphics Package

Plots:

assocplot: Association Plots
barplot: Bar Plots
boxplot: Box Plots
cdplot: Conditional Density Plots
contour: Display Contours
coplot: Conditioning Plots
curve: Draw Function Plots
dotchart: Cleveland's Dot Plots
filled.contour: Level (Contour) Plots
fourfoldplot: Fourfold Plots
hist: Histograms
image: Display a Color Image
matplot: Plot Columns of Matrices
mosaicplot: Mosaic Plots
pairs: Scatterplot Matrices
persp: Perspective Plots
pie: Pie Charts
plot: Generic X-Y Plotting
smoothScatter: Scatterplots with Smoothed Densities Color Representation
spineplot: Spine Plots and Spinograms
stars: Star (Spider/Radar) Plots and Segment Diagrams
stem: Stem-and-Leaf Plots
stripchart: 1-D Scatter Plots
sunflowerplot: Produce a Sunflower Scatter Plot



graphics: The R Graphics Package

Decoration:

abline: Add Straight Lines to a Plot
arrows: Add Arrows to a Plot
axis.POSIXct: Date and Date-time Plotting Functions
axis: Add an Axis to a Plot
box: Draw a Box around a Plot
grid: Add Grid to a Plot
legend Add Legends to Plots
lines: Add Connected Line Segments to a Plot
matlines: Plot Columns of Matrices
matpoints: Plot Columns of Matrices
mtext: Write Text into the Margins of a Plot
panel.smooth: Simple Panel Plot
points: Add Points to a Plot
polygon: Polygon Drawing
polypath: Path Drawing
rasterImage: Draw One or More Raster Images
rect: Draw One or More Rectangles
rug: Add a Rug to a Plot
segments: Add Line Segments to a Plot
symbols: Draw Symbols (Circles, Squares, Stars, Thermometers, Boxplots)
text: Add Text to a Plot
title: Plot Annotation
xspline: Draw an X-spline

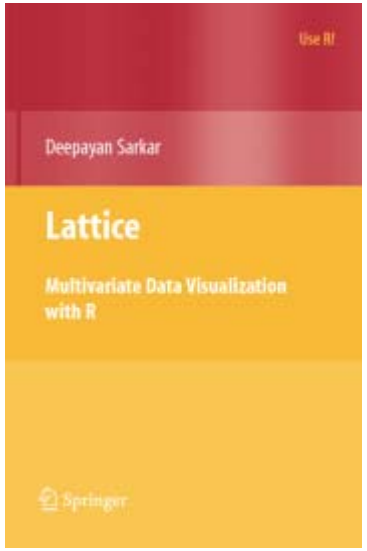
Utilities:

axTicks: Compute Axis Tickmark Locations
close.screen: Creating and Controlling Multiple Screens on a Single Device
erase.screen: Creating and Controlling Multiple Screens on a Single Device
frame: Create/Start a New Plot Frame
grconvertX: Convert between Graphics Coordinate Systems
grconvertY: Convert between Graphics Coordinate Systems
identify: Identify Points in a Scatter Plot
layout: Specifying Complex Plot Arrangements
lcm: Specifying Complex Plot Arrangements
locator: Graphical Input
screen: Creating and Controlling Multiple Screens on a Single Device
split.screen: Creating and Controlling Multiple Screens on a Single Device
strheight: Plotting Dimensions of Character Strings and Math Expressions
strwidth: Plotting Dimensions of Character Strings and Math Expressions

Parameters:

asp: Set up World Coordinates for Graphics Window
clip: Set Clipping Region
par: Set or Query Graphical Parameters
pch: Add Points to a Plot
xinch: Graphical Units
xlim: Set up World Coordinates for Graphics Window
xyinch: Graphical Units
yinch: Graphical Units
ylim: Set up World Coordinates for Graphics Window

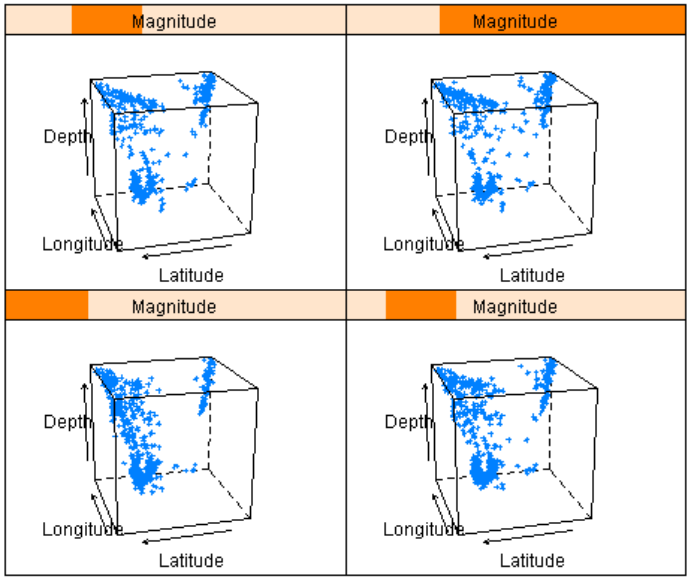
lattice: Trellis Graphics for R

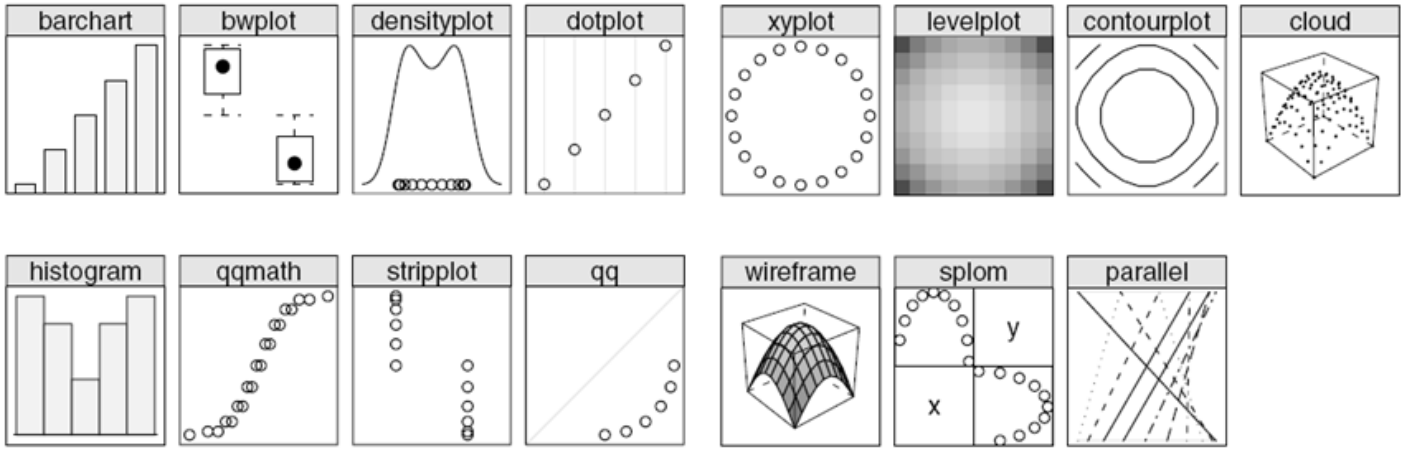


Publisher: Springer;
1st edition (March 12, 2008)

Table 4.1
The plotting functions available in lattice

Lattice Function	Description	Traditional Analogue
`barchart()`	Barcharts	`barplot()`
`bwplot()`	Boxplots Box-and-whisker plots	`boxplot()`
`densityplot()`	Conditional kernel density plots Smoothed density estimate	*none*
`dotplot()`	Dotplots Continuous versus categorical	`dotchart()`
`histogram()`	Histograms	`hist()`
`qqmath()`	Quantile–quantile plots Data set versus theoretical distribution	`qqnorm()`
`stripplot()`	Stripplots One-dimensional scatterplot	`stripchart()`
`qq()`	Quantile–quantile plots Data set versus data set	`qqplot()`
`xyplot()`	Scatterplots	`plot()`
`levelplot()`	Level plots	`image()`
`contourplot()`	Contour plots	`contour()`
`cloud()`	3-dimensional scatterplot	*none*
`wireframe()`	3-dimensional surfaces	`persp()`
`splom()`	Scatterplot matrices	`pairs()`
`parallel()`	Parallel coordinate plots	*none*

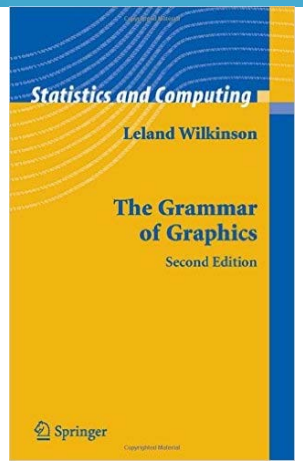


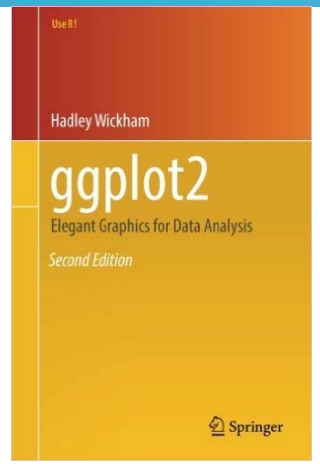


ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics

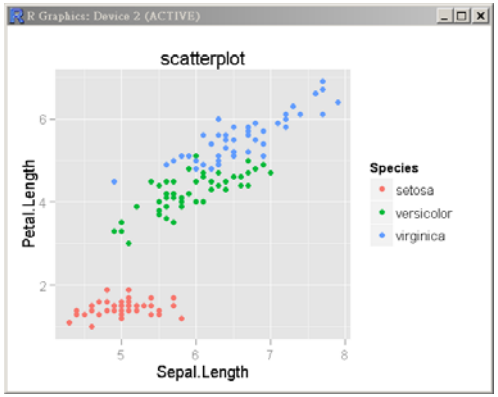
- Hadley Wickham, ggplot2: Elegant Graphics for Data Analysis: http://ggplot2.org/

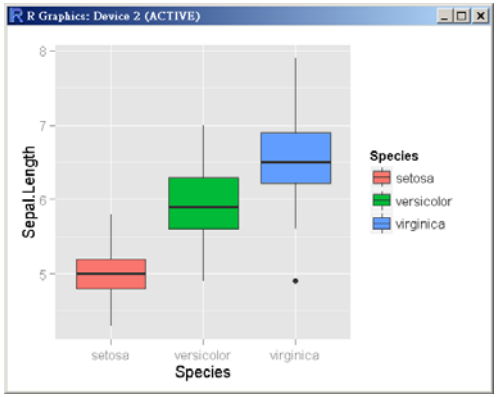
```
qplot(x, y = NULL, ..., data, facets = NULL, margins = FALSE,
  geom = "auto", stat = list(NULL), position = list(NULL),
  xlim = c(NA,NA), ylim = c(NA, NA), log = "", main = NULL,
  xlab = deparse(substitute(x)),
  ylab = deparse(substitute(y)), asp = NA)
```

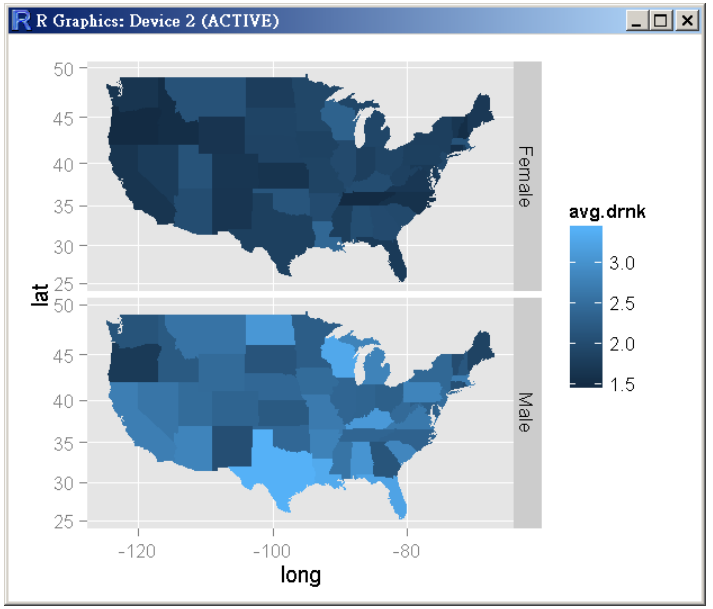





```
library(ggplot2)
qplot(Sepal.Length, Petal.Length, geom="point",
 data=iris, colour = Species, main="scatterplot")
qplot(Species, Sepal.Length, geom="boxplot",
 fill=Species, data=iris)
```







ggplot2 Version of Figures in Lattice:
https://learnr.files.wordpress.com/2009/08/latbook.pdf

http://www.youtube.com/watch?v=HeqHMM4ziXA
http://www.youtube.com/watch?v=n8kYa9vu1l8

Data Visualization Cheat Sheet by RStudio

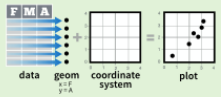
https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf

Data Visualization with ggplot2 Cheat Sheet

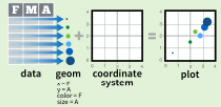
RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

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

Required

Not required, sensible defaults supplied

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**
Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, data = mpg, geom = "point"**)**
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot()
Returns the last plot

ggsave("plot.png", width = 5, height = 5)
Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

Graphical Primitives

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```


a + **geom_blank()**
(Useful for expanding limits)


b + **geom_curve(**aes(yend = lat + 1, xend=long+1,curvature=z)**)** - x, xend, y, yend, alpha, angle, color, curvature, linetype, size


a + **geom_path(**lineend="butt", linejoin="round', linemitre=1**)**
x, y, alpha, color, group, linetype, size


a + **geom_polygon(**aes(group = group)**)**
x, y, alpha, color, fill, group, linetype, size


b + **geom_rect(**aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size


a + **geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)**)** - x, ymax, ymin alpha, color, fill, group, linetype, size

Line Segments

common aesthetics: x, y, alpha, color, linetype, size


b + **geom_abline(**aes(intercept=0, slope=1)**)**
b + **geom_hline(**aes(yintercept = lat)**)**
b + **geom_vline(**aes(xintercept = long)**)**

b + **geom_segment(**aes(yend=lat+1, xend=long+1)**)**
b + **geom_spoke(**aes(angle = 1:1155, radius = 1)**)**

One Variable

Continuous

```
c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)
```


c + **geom_area(stat = "bin")**
x, y, alpha, color, fill, linetype, size


c + **geom_density(**kernel = "gaussian"**)**
x, y, alpha, color, fill, group, linetype, size, weight


c + **geom_dotplot()**
x, y, alpha, color, fill


c + **geom_freqpoly()**
x, y, alpha, color, group, linetype, size


c + **geom_histogram(**binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight


c2 + **geom_qq(**aes(sample = hwy)**)**
x, y, alpha, color, fill, linetype, size, weight

Discrete

```
d <- ggplot(mpg, aes(fl))
```


d + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

Two Variables

Continuous X, Continuous Y

```
e <- ggplot(mpg, aes(cty, hwy))
```


e + **geom_label(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust


e + **geom_jitter(**height = 2, width = 2**)**
x, y, alpha, color, fill, shape, size


e + **geom_point()**
x, y, alpha, color, fill, shape, size, stroke


e + **geom_quantile()**
x, y, alpha, color, group, linetype, size, weight


e + **geom_rug(**sides = "bl"**)**
x, y, alpha, color, linetype, size


e + **geom_smooth(**method = lm**)**
x, y, alpha, color, fill, group, linetype, size, weight


e + **geom_text(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

```
f <- ggplot(mpg, aes(class, hwy))
```


f + **geom_col()**
x, y, alpha, color, fill, group, linetype, size


f + **geom_boxplot()**
x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight


f + **geom_dotplot(**binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill, group


f + **geom_violin(**scale = "area"**)**
x, y, alpha, color, fill, group, linetype, size, weight

Discrete X, Discrete Y

```
g <- ggplot(diamonds, aes(cut, color))
```


g + **geom_count()**
x, y, alpha, color, fill, shape, size, stroke

Continuous Bivariate Distribution

```
h <- ggplot(diamonds, aes(carat, price))
```


h + **geom_bin2d(**binwidth = c(0.25, 500)**)**
x, y, alpha, color, fill, linetype, size, weight


h + **geom_density2d()**
x, y, alpha, colour, group, linetype, size


h + **geom_hex()**
x, y, alpha, colour, fill, size

Continuous Function

```
i <- ggplot(economics, aes(date, unemploy))
```


i + **geom_area()**
x, y, alpha, color, fill, linetype, size


i + **geom_line()**
x, y, alpha, color, group, linetype, size


i + **geom_step(**direction = "hv"**)**
x, y, alpha, color, group, linetype, size

Visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```


j + **geom_crossbar(**fatten = 2**)**
x, y, ymax, ymin, alpha, color, fill, group, linetype, size


j + **geom_errorbar()**
x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)


j + **geom_linerange()**
x, ymin, ymax, alpha, color, group, linetype, size


j + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

Maps

```
data <- data.frame(murder = USArrests$Murder,
  state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```


k + **geom_map(**aes(map_id = state), map = map**)** + **expand_limits(**x = map$long, y = map$lat**)**
map_id, alpha, color, fill, linetype, size

Three Variables

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
l <- ggplot(seals, aes(long, lat))
```


l + **geom_contour(**aes(z = z)**)**
x, y, z, alpha, colour, group, linetype, size, weight

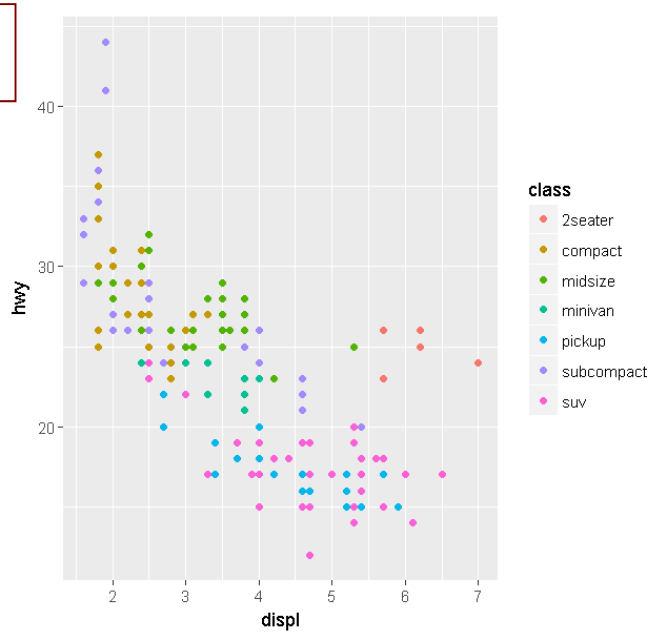
l + **geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill

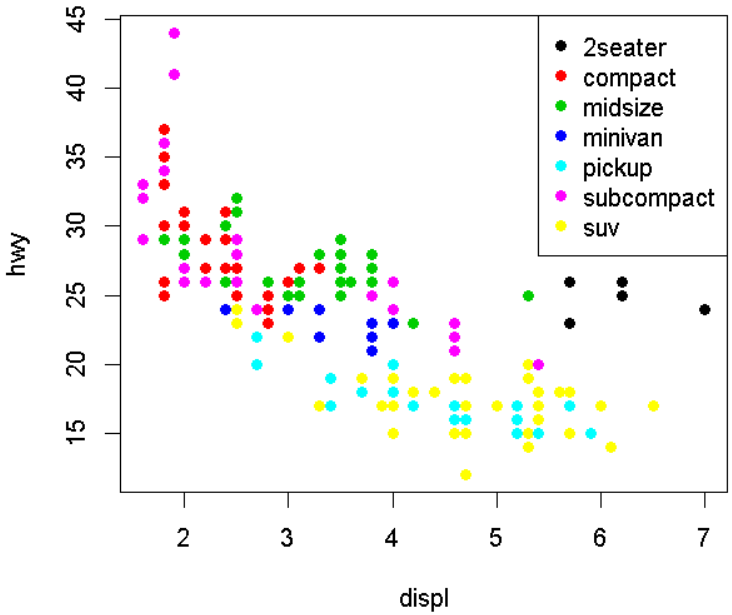
l + **geom_tile(**aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size, width

Base Graphics or ggplot2 ?

```
> ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```

```
> mpg.df <- as.data.frame(mpg)
> attach(mpg.df)
> group <- as.factor(class)
> plot(displ, hwy, col=group, pch=16)
> legend("topright", legend=levels(group),
+  col=1:length(levels(group)), pch=16)
> detach(mpg.df)
```





10 reasons to switch to ggplot
https://mandymejia.wordpress.com/2013/11/13/10-reasons-to-switch-to-ggplot-7/

Comparing Base Graphics with ggplot2
https://sakai.duke.edu/access/content/group/7a48cfac-b05c-4291-8e13-0091b5cd1479/Reference/BaseGraphicsGGPlotComparison.html

Why I use ggplot2
http://varianceexplained.org/r/why-I-use-ggplot2/

Why I don't use ggplot2
http://simplystatistics.org/2016/02/11/why-i-dont-use-ggplot2/

Chart Suggestions—A Thought-Starter

What would you like to show?

Comparison

- Among Items
  - Two Variables per Item
    - Variable Width Column Chart
    - Table or Table with Embedded Charts
  - One Variable per Item
    - Many Categories
      - Few Categories
        - Many Items: Bar Chart
        - Few Items: Column Chart
- Over Time
  - Many Periods
    - Cyclical Data: Circular Area Chart
    - Non-Cyclical Data: Line Chart
  - Few Periods
    - Single or Few Categories: Column Chart
    - Many Categories: Line Chart

Relationship

- Two Variables
  - Scatter Chart
- Three Variables
  - Bubble Chart

Scatterplot Matrices
Correlogram
PCP

Distribution

- Single Variable
  - Few Data Points: Column Histogram
  - Many Data Points: Line Histogram
- Two Variables: Scatter Chart
- Three Variables: 3D Area Chart

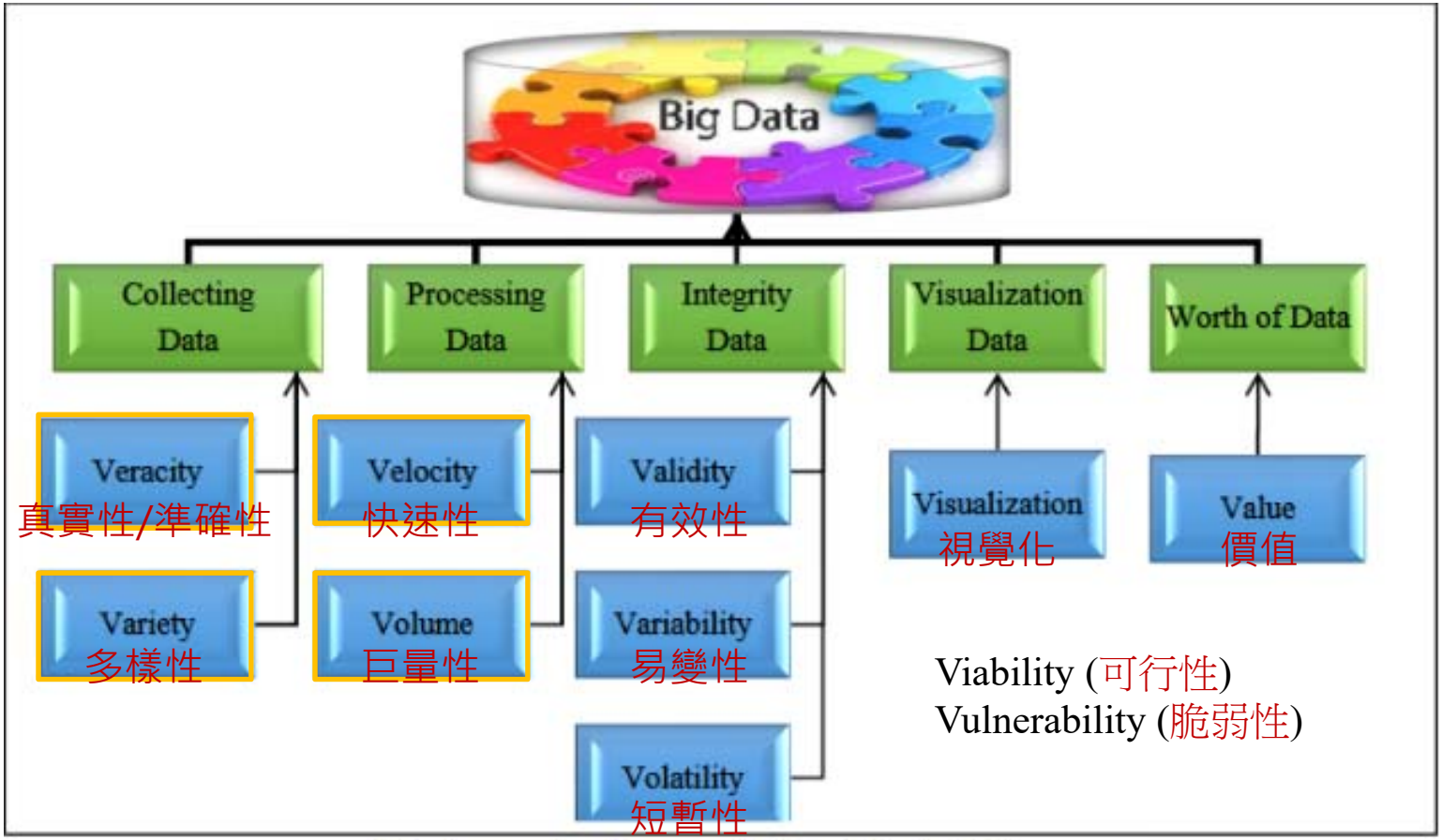
Box Plot
Heat Map
Star Plot

Composition

- Changing Over Time
  - Few Periods
    - Only Relative Differences Matter: Stacked 100% Column Chart
    - Relative and Absolute Differences Matter: Stacked Column Chart
  - Many Periods
    - Only Relative Differences Matter: Stacked 100% Area Chart
    - Relative and Absolute Differences Matter: Stacked Area Chart
- Static
  - Simple Share of Total: Pie Chart
  - Accumulation or Subtraction to Total: Waterfall Chart
  - Components of Components: Stacked 100% Column Chart with Subcomponents

Big Data: The Era of 9 Vs

- Visualization:
  - Visualization will be key to making big data an integral part of decision making.
  - Visualization will be the only way to make big data accessible to a large audience.
  - Visualization will be essential to the analysis of big data so it can be of highest value.



Categorization of Big Data V's

http://blogs.systweak.com/2017/03/big-data-vs-represents-characteristics-or-challenges-of-big-data/

Big Data Visualization

- **Definition - What does Big Data Visualization mean?**
  - Big data visualization refers to the implementation of more contemporary visualization techniques to illustrate the relationships within data. Visualization tactics include applications that can display real-time changes and more illustrative graphics, thus going beyond pie, bar and other charts. These illustrations veer away from the use of hundreds of rows, columns and attributes toward a more artistic visual representation of the data.
- **Techopedia explains Big Data Visualization**
  - Normally when businesses need to present relationships among data, they use graphs, bars and charts to do it. They can also make use of a variety of colors, terms and symbols. The main problem with this setup, however, is that it doesn't do a good job of presenting very large data or data that includes huge numbers. Data visualization uses more interactive, graphical illustrations - including personalization and animation - to display figures and establish connections among pieces of information.

techopedia™

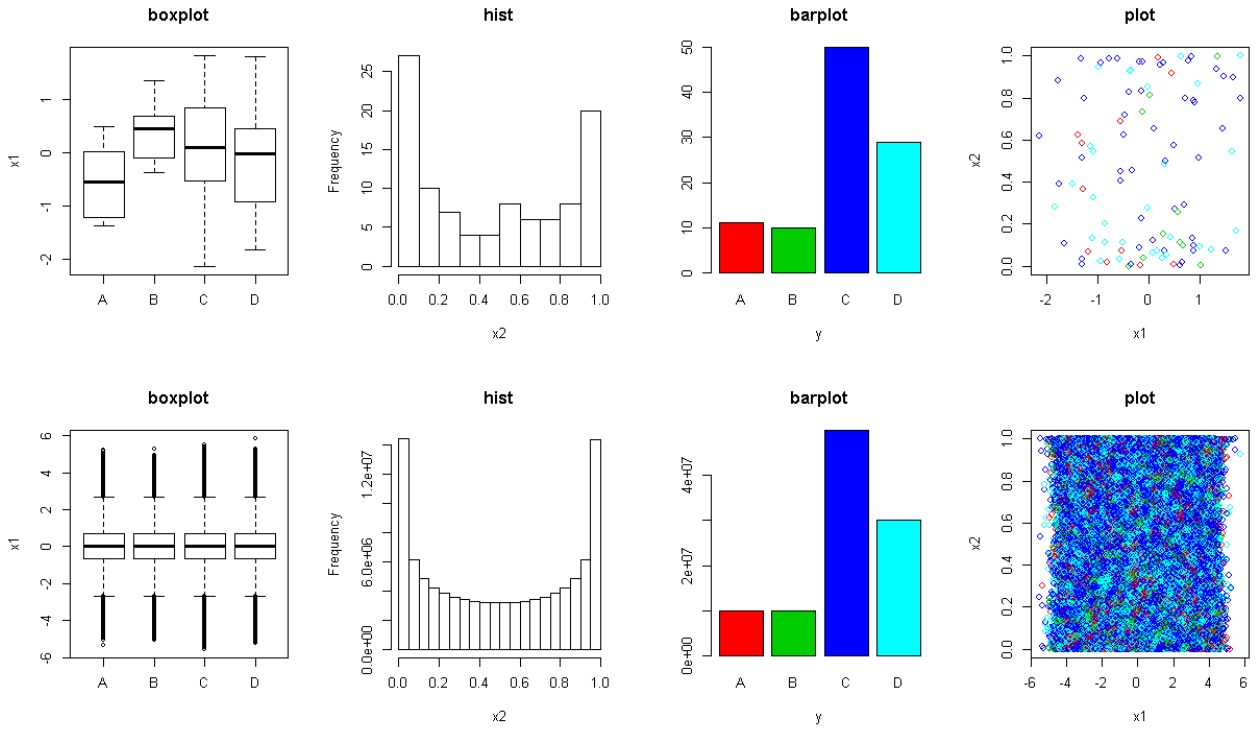
https://www.techopedia.com/definition/28988/big-data-visualization

The Challenge of Visualizing Big Data

```
> n <- 1e+02
```

a large p?

```
> n <- 1e+08
```




```
> n <- 1e+02
> y <- as.factor(sample(LETTERS[1:4], n, replace=T, prob=c(0.1, 0.1, 0.5, 0.3)))
> x1  <- rnorm(n)
> x2  <- rbeta(n, 0.5, 0.5)
> xydata <- data.frame(y, x1, x2)
> par(mfrow=c(1,4))
> boxplot(x1~y, data=xydata, ylab="x1", main="boxplot")
> hist(x2, xlab="x2", main="hist")
> barplot(table(y), xlab="y", col = 2:5, main="barplot")
> plot(x1, x2, main="plot", col=as.integer(y)+1)
```

Two principles:
Look at Less Data;
or Look at Data Faster

The Challenge of Visualizing Big Data

Visualising big data in R, April 2013 Birmingham R User Meeting, Alastair Sanderson, www.AlastairSanderson.com, 23rd April 2013.

- Only a few million pixels on a screen, but many more data points.
- Therefore need to generate a suitable summary to plot instead.
- Directly visualizing raw big data is probably pointless (at least for static graphics).
- A typical 1D/2D plot of big data will have lots of overlapping & therefore obscured points: these different values will be visually indistinguishable.

Lidong Wang, Guanghui Wang, Cheryl Ann Alexander. Big Data and Visualization: Methods, Challenges and Technology Progress. Digital Technologies. Vol. 1, No. 1, 2015, pp 33-38. http://pubs.sciepub.com/dt/1/1/7

- Scalability and dynamics.
- Visualization of big data with diversity and heterogeneity (structured, semi-structured, and unstructured) is a big problem
- Effective visualization for the high complexity and high dimensionality in big data.
- Other problems for big data visualization: Visual noise, Information loss, Large image perception, High rate of image change, High performance requirements.
- Perceptual and interactive scalability are also challenges of big data visualization.

Graphics of Large Datasets

Scatterplot Matrix Techniques for Large *N*

D. B. CARR, R. J. LITTLEFIELD, W. L. NICHOLSON, and J. S. LITTLEFIELD*

High-performance interaction with scatterplot matrices is a powerful approach to exploratory multivariate data analysis. For a small number of data points, real-time interaction is possible and overplotting is usually not a major problem. When the number of plotted ... ever, display techniques that deal with overplotting ... are important. This article addresses these two prob... density representation by gray scale or by symbol ... brushing, and animation sequences. We also discus... generally applicable, including interactive graphical ... any plot in a collection of scatterplots and comp... matrices.

KEY WORDS: Density representations; Animation sequences; Graphical subset selection; Interactive graphics.

500 points can seem large. For our purposes, *N* is large if plotting or computation times are long, or if plots can have an extensive amount of overplotting. Figure 1 provides an

each scatterplot would contain 4,109 points. With 14 plots, the total number of points in the display is almost 50,000.

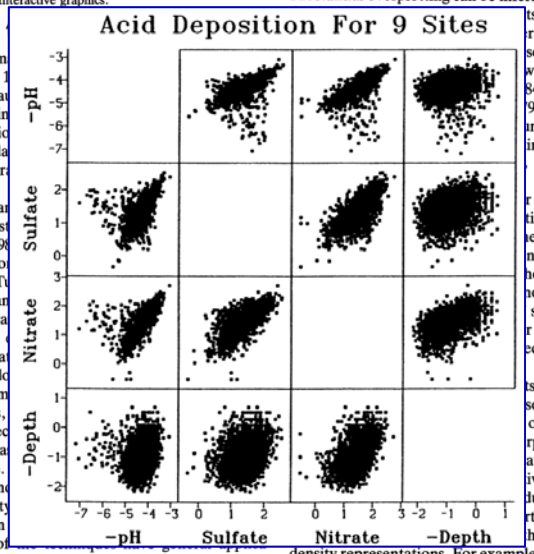
large. The exhibit also fits the other definitions of large. Substantial overplotting can be inferred from plotted area, ... density representations. For example, simple random sampling omits most outliers and obscures some local ... while introducing others. In other words, samplin... alter features that are frequently of interest in ter... subsetting, identification, and comparison. Samplin... powerful tool, but even sophisticated analysts wh... handle complex sampling will find it useful to look ... whole data set.

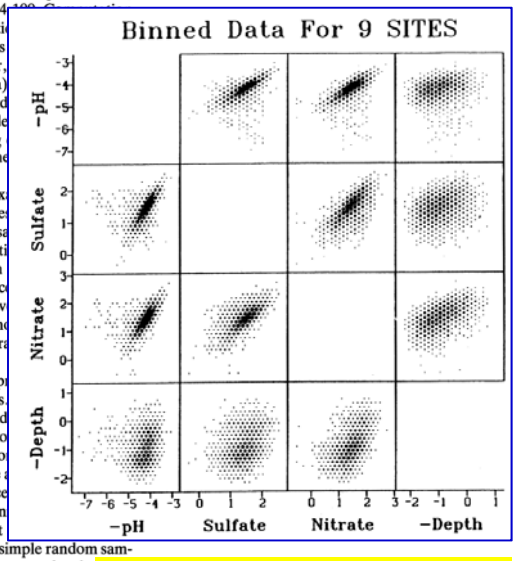
A scatterplot m... display of $p(p-1$... of Figure 1. Becau... two scatterplots in... whole is a collectio... the background da... and interactive gra... from any plot.

Since 1980, mar... have appeared in st... Cleveland 1985, 19... Carr and Nicholso... and Tukey 1983; Tu... With different nam... tant themes preva... effective tool for ... (b) Scatterplot ma... information. Undo... riety of enhancem... tions, smoothings, ... active subset selec... way into an increa... into common use. ... sequel to Carr and ... interaction, density ... that are helpful in ... Although some of ... bility, the context here is exploratory analysis using a graphics terminal with a color lookup table.

2. LARGE *N*

What is large depends on the frame of reference. If available plotting space for a scatterplot is a one-inch square,

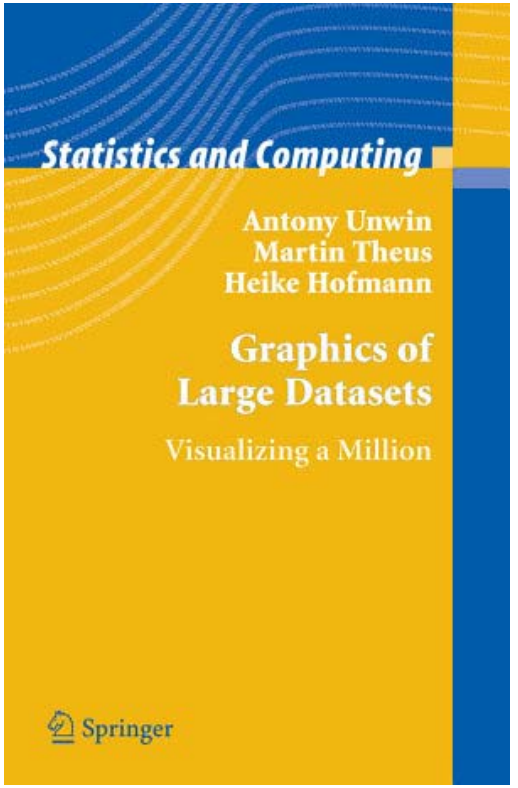




Hexagon area density representation

* D. B. Carr is Senior Research Scientist, Statistics; R. J. Littlefield is Senior Research Scientist, Computers and Information Sciences; W. L. Nicholson is Lead Scientist, Statistics; and J. S. Littlefield is Research Scientist, Statistics; all in the Computational Sciences Department, Pacific Northwest Laboratory, Richland, WA 99352. The authors would like to thank Vern Crow for substantial software contributions and Paul Tukey for suggesting the hexagonal binning algorithm in the Appendix. This work was supported by the Applied Mathematical Sciences division of the U.S. Department of Energy under Contract DE-ACO6-76RLO 1830.

Journal of the American Statistical Association
June 1987, Vol. 82, No. 398, Statistical Graphics



Antony Unwin, Martin Theus, Heike Hofmann, Publisher: Springer; 2006 edition (July 24, 2006)

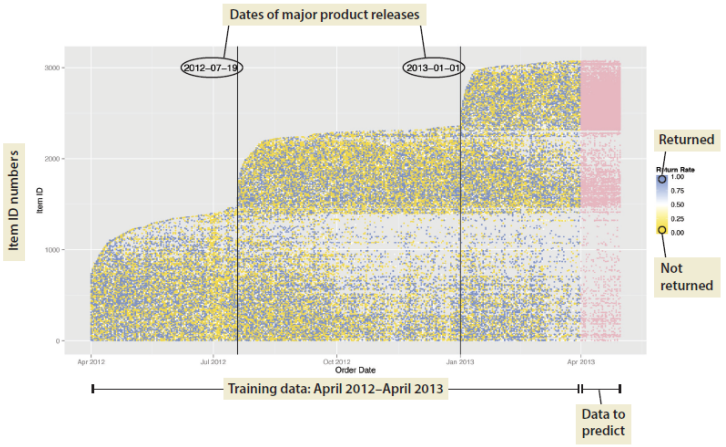
Data Visualization and Statistical Graphics in Big Data Analysis

ANNUAL REVIEWS
For Librarians & Agents For Authors
JOURNALS A-Z MULTIMEDIA JOURNAL INFO PRICING & SU
Home / Annual Review of Statistics and Its Application / Volume 3, 2016 / Cook, pp 133-159

Data Visualization and Statistical Graphics in Big Data Analysis

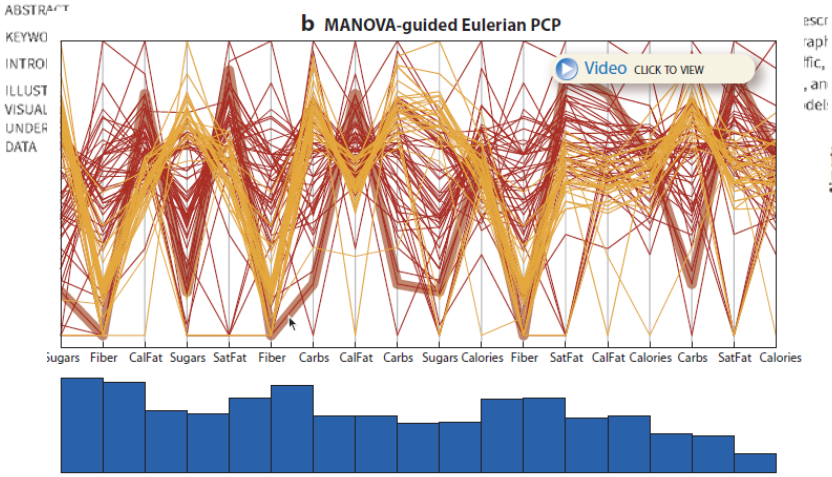
Annual Review of Statistics and Its Application
Vol. 3:133-159 (Volume publication date June 2016)
DOI: 10.1146/annurev-statistics-041715-033420

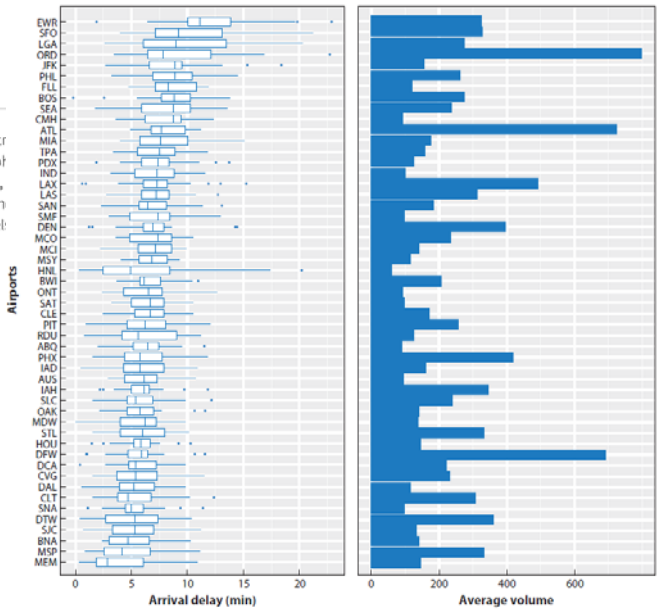
Dianne Cook,[1] Eun-Kyung Lee,[2] and Mahbubul Majumder[3]
[1]Department of Econometrics and Business Statistics, Monash University, Clayton, Victoria 3800, Australia; email: dicook@monash.edu
[2]Department of Statistics, Ewha Womans University, Seoul 120-750, Korea; email: lee.eunk@ewha.ac.kr
[3]Department of Mathematics, University of Nebraska, Omaha, Nebraska 68182; email: mmajumder@unomaha.edu

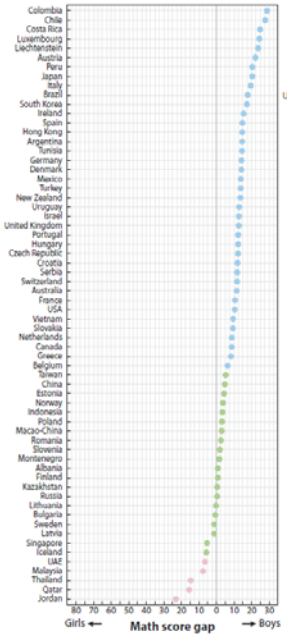


Sections

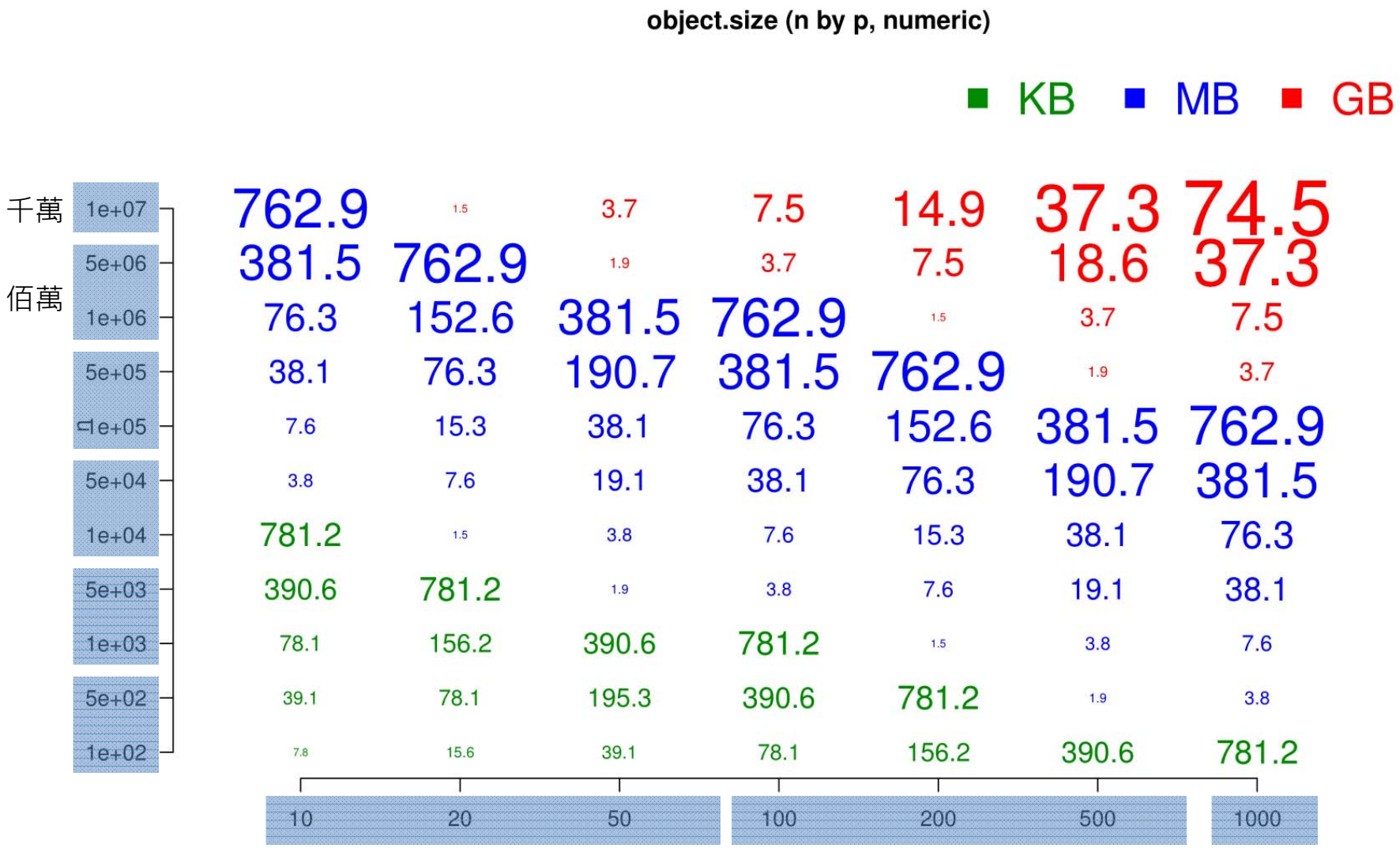
Abstract







object.size{utils}

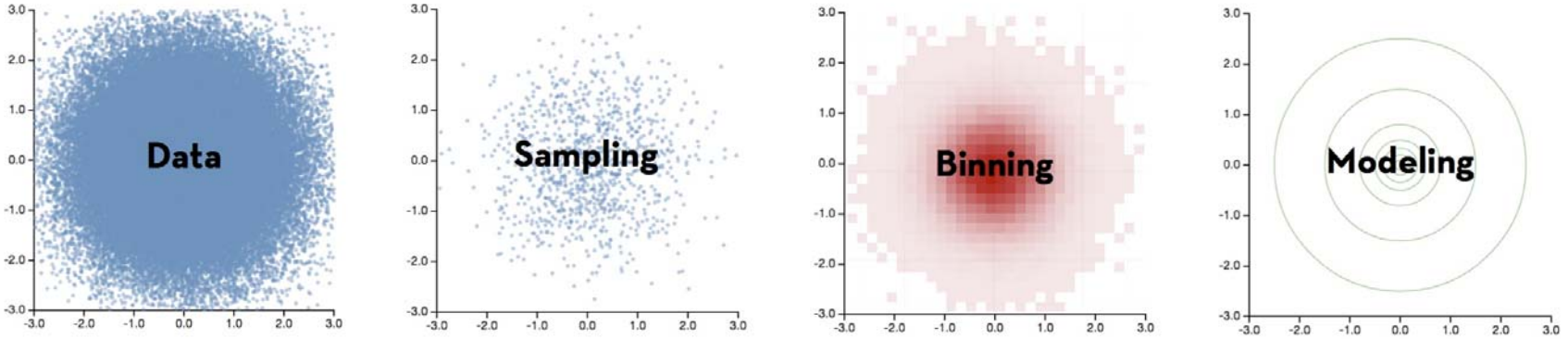


(n*p*8)/(1024*1024) MB

1 Bit = Binary Digit; 8 Bits = 1 Byte; 1024 Bytes = 1 Kilobyte; 1024 Kilobytes = 1 Megabyte
1024 Megabytes = 1 Gigabyte; 1024 Gigabytes = 1 Terabyte; 1024 Terabytes = 1 Petabyte

How Can We Visualize and Interact with Billion+ Record Databases in Real-time?

- Two central challenges:
  - need to keep visualizations perceptually effective regardless of the number of input data points.
  - need to support real-time interaction to enable rapid and iterative exploratory analysis.
- Perceptual and interactive scalability should be limited by the chosen resolution of the visualized data, not the number of records.



http://skandel.github.io/slides/strata2013/part1

Reduce the Original Massive Dataset to a More Manageable Summary

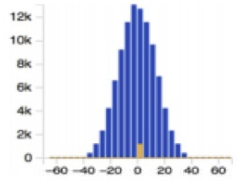
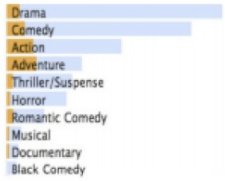
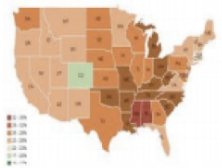
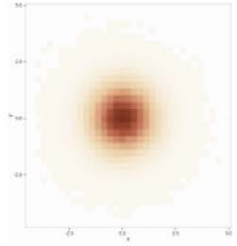
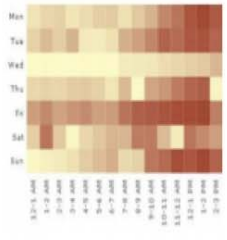
- Bin => Aggregate => Smooth => Plot
  - Bin: number of bins, bin shape: rectangular bins, hexagonal bins,
  - Aggregate: a summary within each bin: counts, sum, average, ...
  - Smoothing: (e.g., convolution with a kernel) on aggregated data to better approximate an underlying continuous density.
  - Plot: create visualizations.

Bin: divide data domain into discrete "buckets"
- categories: already discrete (but check cardinality)
- numbers: choose bin intervals (uniform, quantile, ...)
- time: choose time unit: hour, day, month, etc.
- geo: bin x, y coordinates after cartographic projection.

- Plot: visual encoding
  - choose most effective encoding
  - 1D plot: position or length encoding: histogram, line charts, etc
  - 2D plot: area or color encoding, spatial dimensions (x, y) already allocated.
    - area for magnitude estimation.
    - color (per-pixel level) provides an overall gestalt.
- Color encoding:
  - Keep small non-zero values visible (outliers)
  - Match color ramp to perceptual distances
  - Enable exploration across values ranges

Design Space of Binned Plots

Design Space of Binned Plots

	Numeric	Ordinal	Temporal	Geographic
1D	 Histogram	 Bar Chart	 Line Graph / Area Chart	 Choropleth Map
2D	 Binned Scatter Plot	 Heatmap	 Temporal Heatmap	 Geographic Heatmap

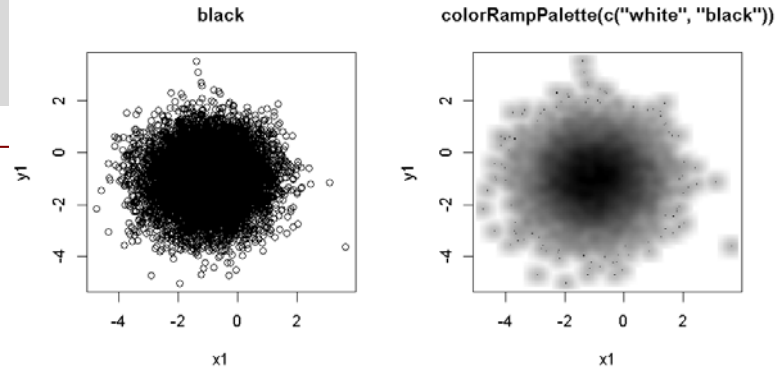
http://skandel.github.io/slides/strata2013/part1

smoothScatter {graphics}: Scatterplots with Smoothed Densities Color Representation

- **smoothScatter** produces a smoothed color density representation of a scatterplot, obtained through a (2D) kernel density estimate.
  - **nbin**: numeric vector of length one (for both directions) or two (for x and y separately) specifying the number of equally spaced grid points for the density estimation; directly used as gridsize in bkde2D().
  - **bandwidth**: numeric vector (length 1 or 2) of smoothing bandwidth(s). If missing, a more or less useful default is used. bandwidth is subsequently passed to function bkde2D.
  - **nrpoints**: number of points to be superimposed on the density image. The first nrpoints points from those areas of lowest regional densities will be plotted. Adding points to the plot allows for the identification of outliers. If all points are to be plotted, choose nrpoints = Inf.

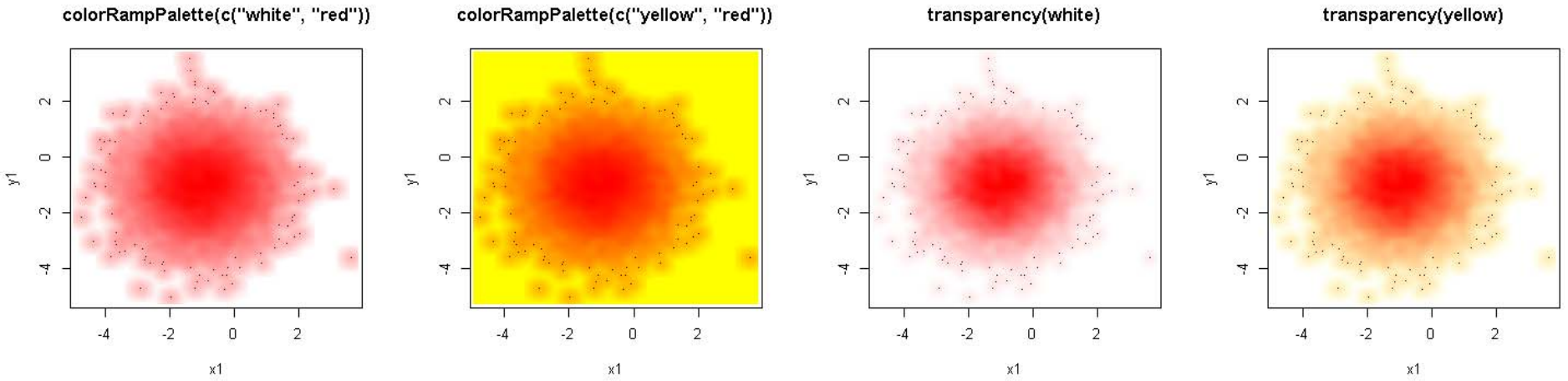
```
smoothScatter(x, y = NULL, nbin = 128, bandwidth,
              colramp = colorRampPalette(c("white", blues9)),
              nrpoints = 100, ret.selection = FALSE,
              pch = ".", cex = 1, col = "black",
              transformation = function(x) x^.25,
              postPlotHook = box,
              xlab = NULL, ylab = NULL, xlim, ylim,
              xaxs = par("xaxs"), yaxs = par("yaxs"),
```

```
> n <- 1e+04
> x1 <- rnorm(n, mean = -1, sd = 1)
> y1 <- rnorm(n, mean = -1, sd = 1)
> x2 <- rnorm(n, mean = 2, sd = 1)
> y2 <- rnorm(n, mean = 2, sd = 1)
> par(mfrow=c(1, 2))
> plot(x1, y1, main="black")
> smoothScatter(x1, y1, col="black", colramp=colorRampPalette(c("white", "black")),
main='colorRampPalette(c("white", "black"))')
```



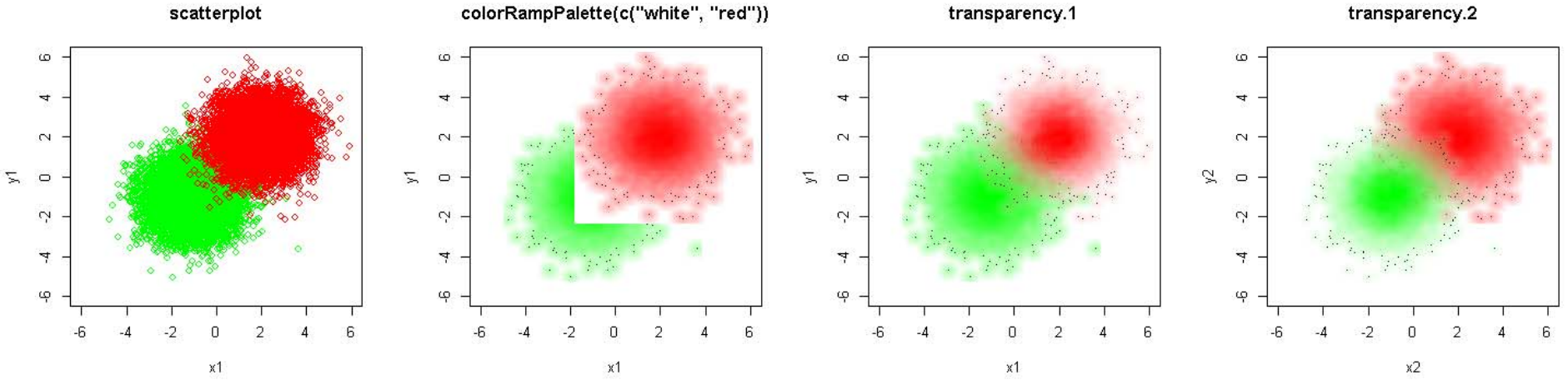
Transparency in smoothScatter {graphics}

```
> par(mfrow=c(1, 4))
> smoothScatter(x1, y1, col="black", colramp=colorRampPalette(c("white", "red")),
main='colorRampPalette(c("white", "red"))')
>
> smoothScatter(x1, y1, col="black", colramp=colorRampPalette(c("yellow", "red")),
main='colorRampPalette(c("yellow", "red"))')
>
> transparency.white <- colorRampPalette(c(rgb(1, 1, 1, 0), rgb(1, 0, 0, 1)), alpha = TRUE)
> smoothScatter(x1, y1, col="black", colramp=transparency.white, main="transparency(white)")
>
> transparency.yellow <- colorRampPalette(c(rgb(1, 1, 0, 0), rgb(1, 0, 0, 1)), alpha = TRUE)
> smoothScatter(x1, y1, col="black", colramp=transparency.yellow,
main="transparency(yellow)")
```



Points Overlapping by Transparency

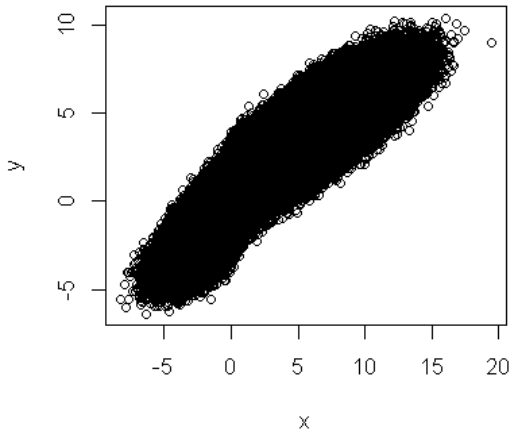
```
> par(mfrow=c(1, 4))
> plot(x1, y1, col="green", xlim=c(-6, 6), ylim=c(-6, 6), main="scatterplot")
> points(x2, y2, col="red")
>
> smoothScatter(x1, y1, col="black", colramp=colorRampPalette(c("white", "green")),
xlim=c(-6, 6), ylim=c(-6, 6), main='colorRampPalette(c("white", "red"))')
> smoothScatter(x2, y2, col="black", colramp=colorRampPalette(c("white", "red")), add=T)
>
> smoothScatter(x1, y1, col="black", colramp=colorRampPalette(c("white", "green")),
xlim=c(-6, 6), ylim=c(-6, 6), main="transparency.1")
> transparency.1 <- colorRampPalette(c(rgb(1, 1, 1, 0), rgb(1, 0, 0, 1)), alpha = TRUE)
> smoothScatter(x2, y2, col="black", colramp=transparency.1, add=T)
>
> smoothScatter(x2, y2, col="black", colramp=colorRampPalette(c("white", "red")),
xlim=c(-6, 6), ylim=c(-6, 6), main="transparency.2")
> transparency.2 <- colorRampPalette(c(rgb(1, 1, 1, 0), rgb(0, 1, 0, 1)), alpha = TRUE)
> smoothScatter(x1, y1, col="black", colramp=transparency.2, add=T)
```



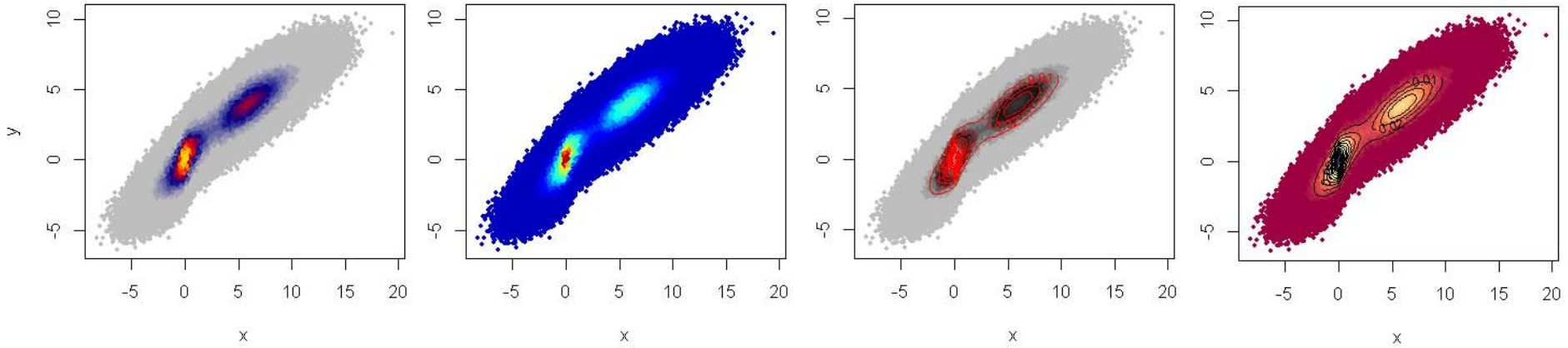
heatscatter {LSD}: A colored scatterplot based on a two-dimensional Kernel Density Estimation

```
> n <- 1e+07 # use n <- 1e+04 for simplicity
> x <- c(rnorm(n/2), rnorm(n/2)+4)
> y <- x + rnorm(n, sd=0.8)
> x <- sign(x)*abs(x)^1.3
> plot(x, y)
> library(LSD) # install.packages("LSD")
> par(mfrow=c(2, 2))
> heatscatter(x, y)
> heatscatter(x, y, colpal="bl2gr2rd", cor=FALSE)
> heatscatter(x, y, cor=FALSE, add.contour=TRUE,
color.contour="red", greyscale=TRUE)
> heatscatter(x, y, colpal="spectral", cor=FALSE,
add.contour=TRUE)
```

LSD: Lots of Superior Depictions



heatscatter heatscatter heatscatter heatscatter

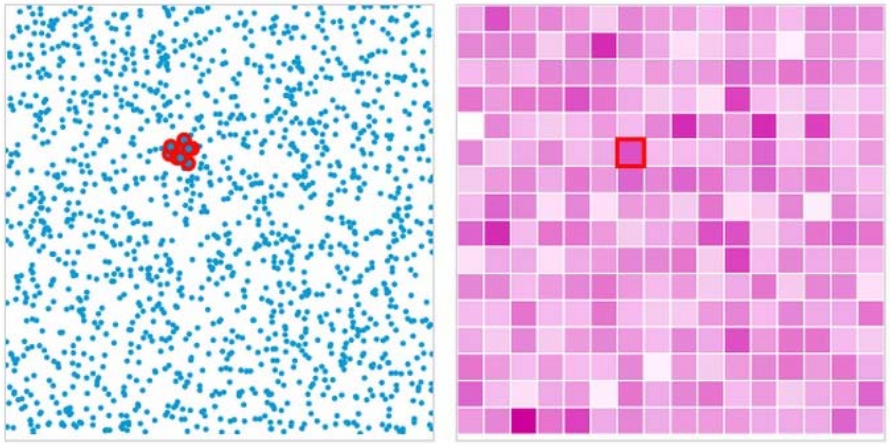



```
> demotour()
```

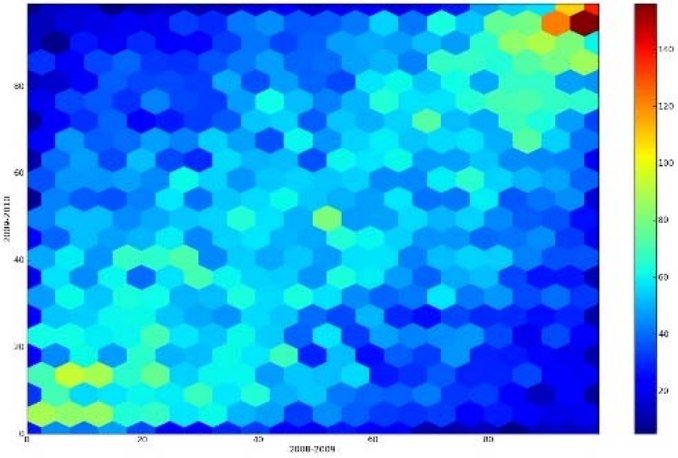
Binning Technique

- Binning is a technique of data aggregation used for grouping a dataset of N values into less than N discrete groups.
  - the XY plane is uniformly tiled with polygons (squares, rectangles or hexagons).
  - the number of points falling in each bin (tile) are counted and stored in a data structure.
  - the bins with count > 0 are plotted using a color range (heatmap) or varying their size in proportion to the count.

Rectangular binning



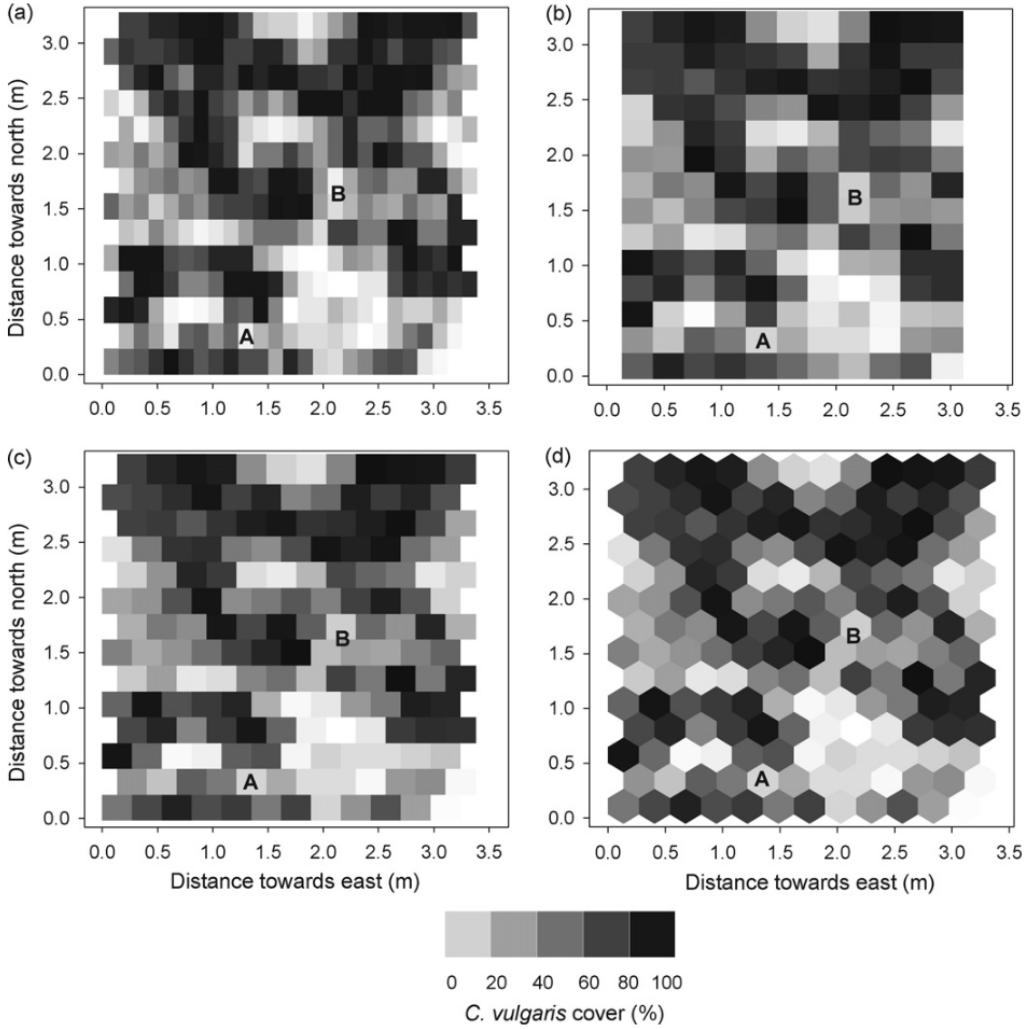
Hexagonal binning



http://www.meccanismocomplesso.org/en/hexagonal-binning/

hexbin Package: Hexagonal Binning Routines

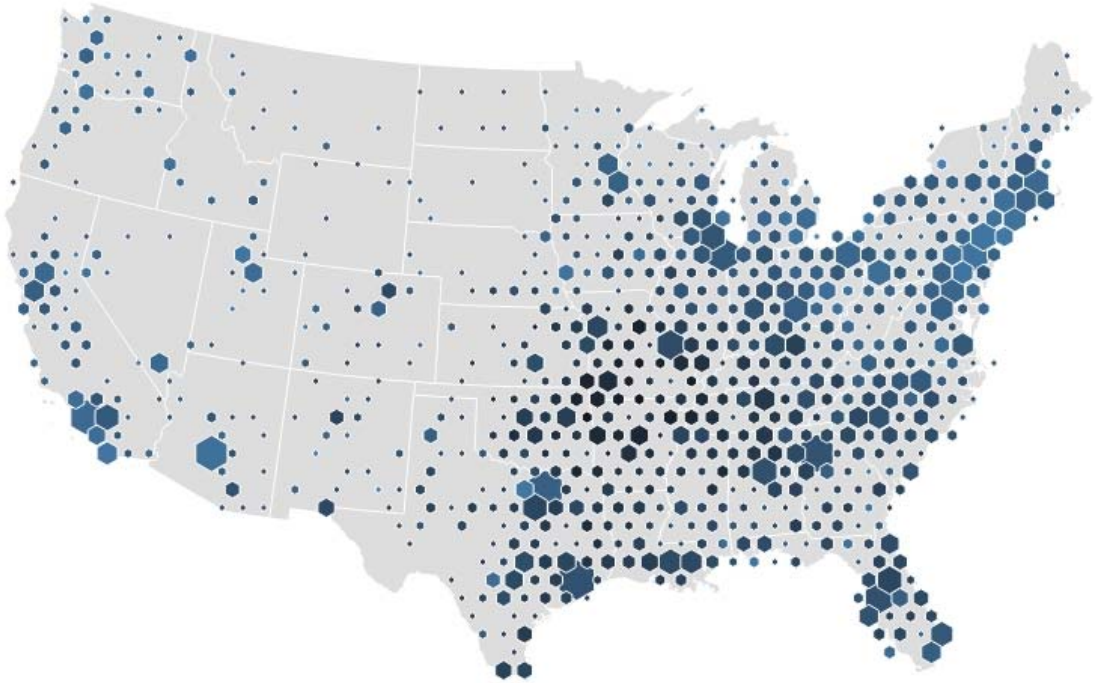
Colin P.D. Birch, Sander P. Oom, Jonathan A. Beecham, Rectangular and hexagonal grids used for observation, experiment and simulation in ecology, Ecological Modelling 206(2007) 347–359



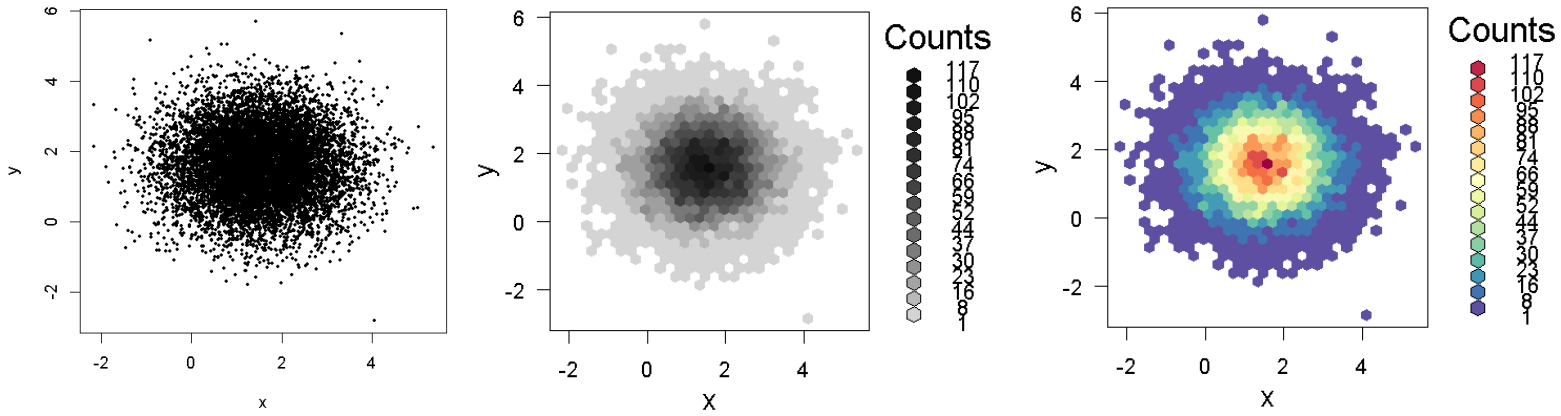
- The rectangular grid is generally preferred because of its symmetrical, orthogonal coordinate system.
- The rectangular grid is also convenient for studies varying resolution, such as hierarchical grids, because squares can easily be combined to form larger squares with the same alignment.

Why hexagons?

- Working over a larger area, a square grid will suffer more from distortion due to curvature than hexagons.
- Hexagons have symmetry of nearest neighbors which is lacking in square bins.
- Hexagons are visually less biased for displaying densities than other regular tesselations.
- The hexagon is the most complex regular polygon that can fill a plane (without gaps or overlap).

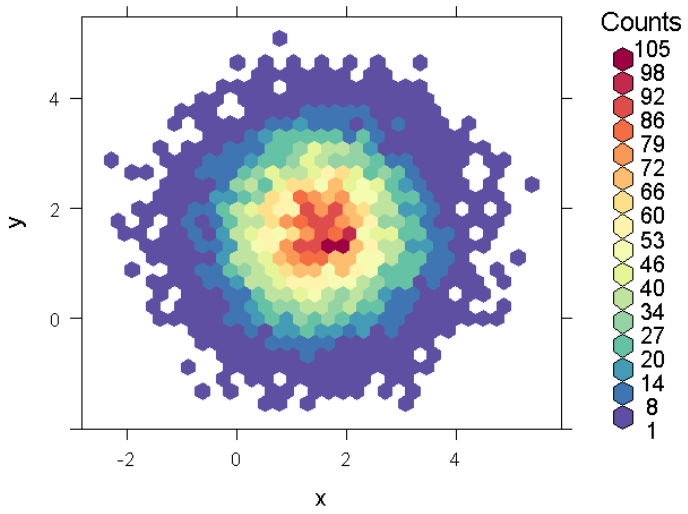


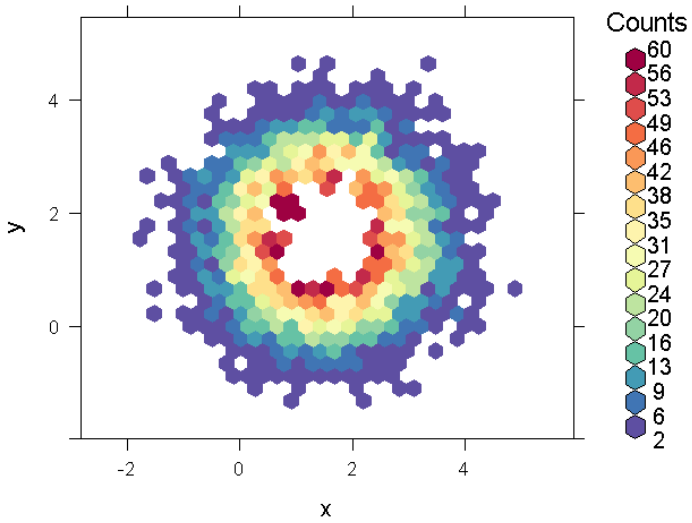
Example




```
> x <- rnorm(mean=1.5, 10000)
> y <- rnorm(mean=1.6, 10000)
> my.data <- data.frame(x, y)
>
> pk <- c("RColorBrewer", "hexbin", "gplots")
> install.packages(pk, repos="http://cran.csie.ntu.edu.tw")
> library(RColorBrewer)
> # create rainbow color
> col_rb <- colorRampPalette(rev(brewer.pal(11, 'Spectral')))
> # scatterplot
> plot(my.data, pch=16, col='black', cex=0.5)
> library(hexbin)
> h <- hexbin(my.data) # create a hexbin object
> h
'hexbin' object from call: hexbin(x = my.data)
n = 10000  points in    nc = 598  hexagon cells in grid dimensions  36 by 31
> plot(h) # in grey level
> plot(h, colramp=col_rb) # rainbow color
```

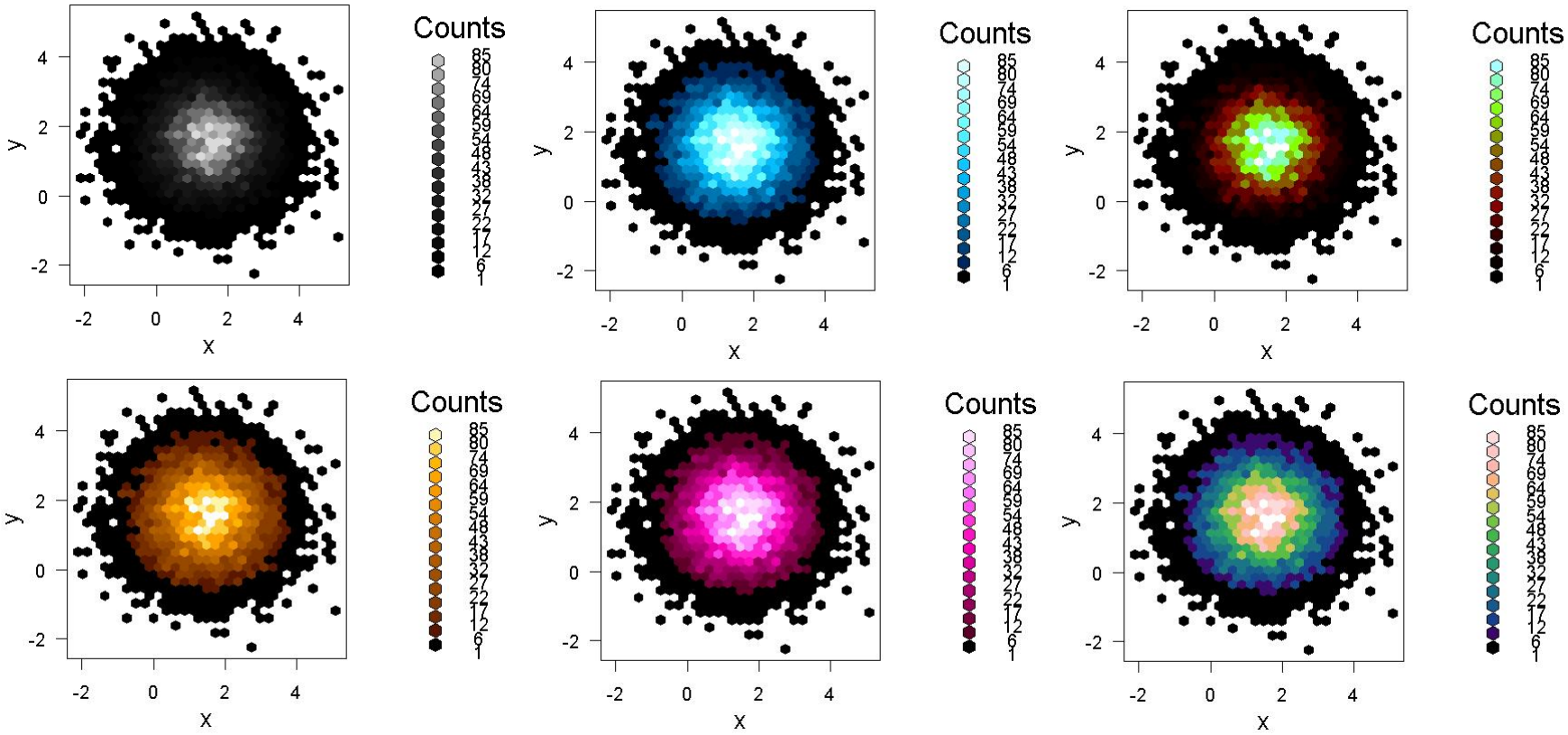
hexbinplot {hexbin}






```
> hexbinplot(y ~ x, data=my.data, colramp=col_rb) # more flexible
> # set max and min counts
> hexbinplot(y ~ x, data=my.data, colramp=col_rb, mincnt=2, maxcnt=60)
```

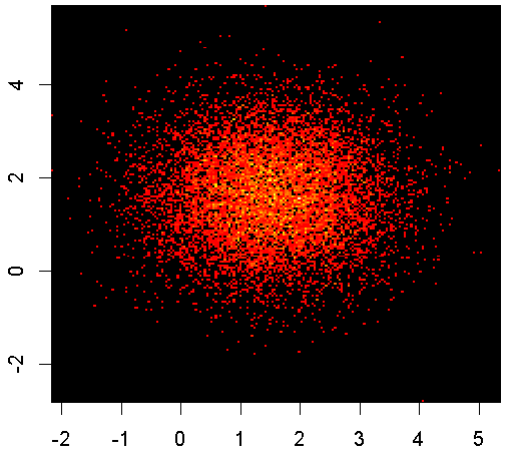
Various colramp

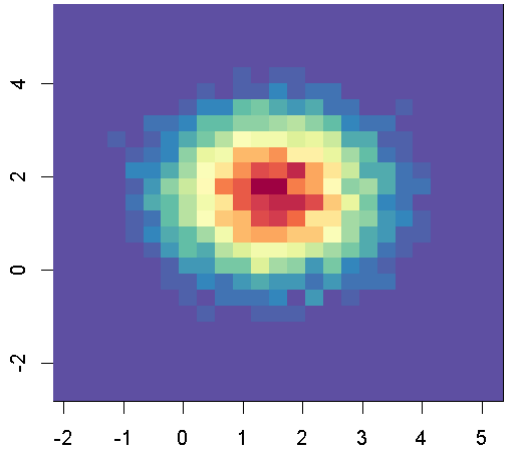


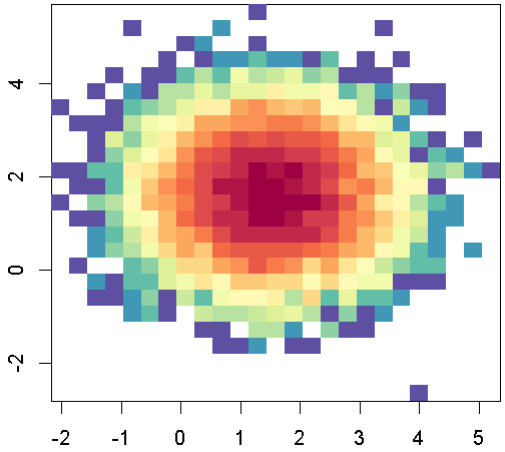

```
plot(h, colramp=LinGray)
plot(h, colramp=BTC)
plot(h, colramp=LinOCS)
plot(h, colramp=heat.ob)
plot(h, colramp=magent)
plot(h, colramp=plinrain)
```

Using hist2d{gplots}

- Another simple way to get a quick 2D histogram is to use the **hist2d** function from the **gplots** package. Again, the default invocation leaves a lot to be desired:



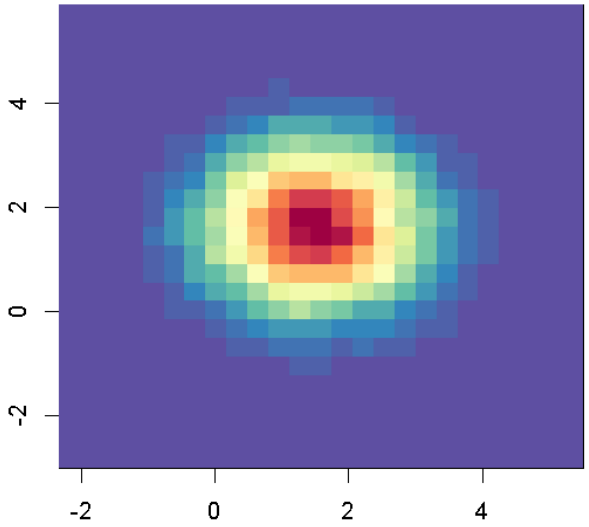


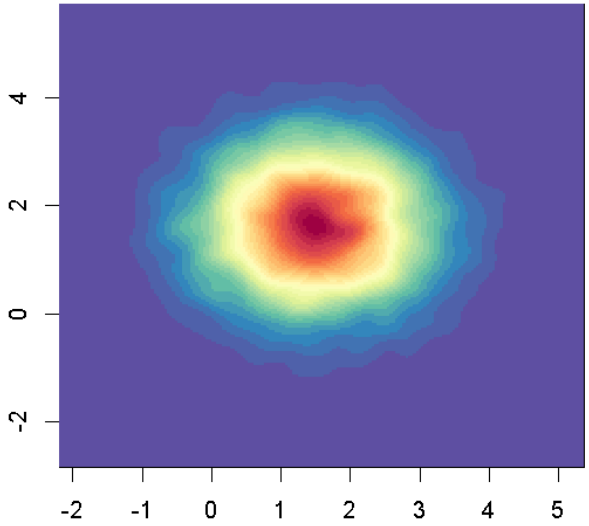



```
> h2d <- hist2d(my.data)
> # binsizing and coloring
> h2d <- hist2d(my.data, nbins=25, col=col.rb(32))
> # log scale
> h2d <- hist2d(my.data, nbins=25, col=col.rb(32), FUN=function(x) log(length(x)))
```

using kde2d {MASS} + image

- There are over 20 packages with which to do (kernel) density estimation.

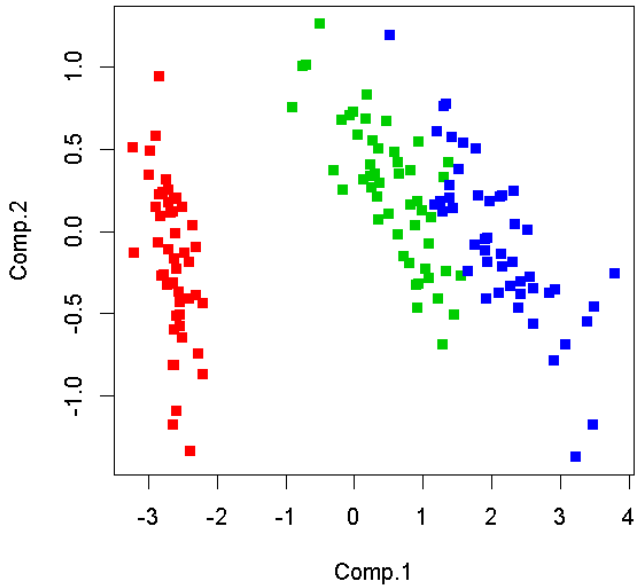


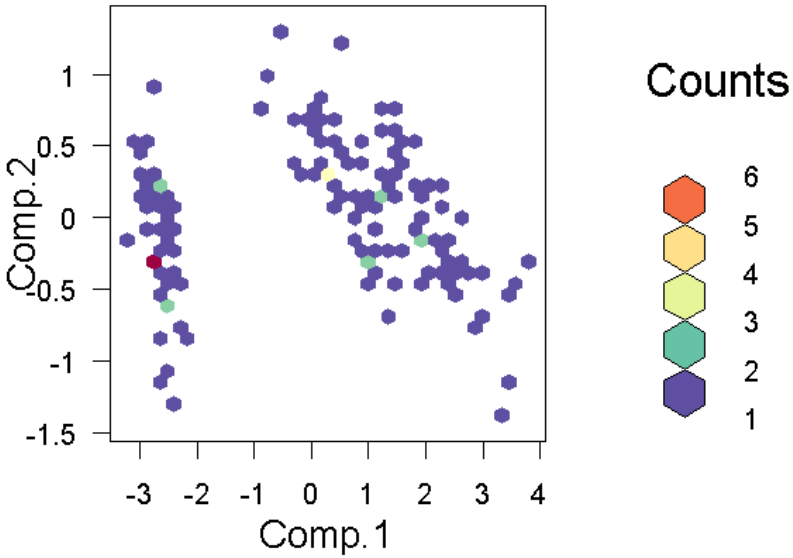



```
> library(MASS)
> k <- kde2d(my.data$x, my.data$y)
> image(k, col=col_rb(32))
> k <- kde2d(my.data$x, my.data$y, n=200) # Adjust binning
> image(k, col=col_rb(32))
```

```
> x <- rnorm(mean=1.5, 10000)
> y <- rnorm(mean=1.6, 10000)
> my.data <- data.frame(x, y)
```

hexbin for Small Dataset

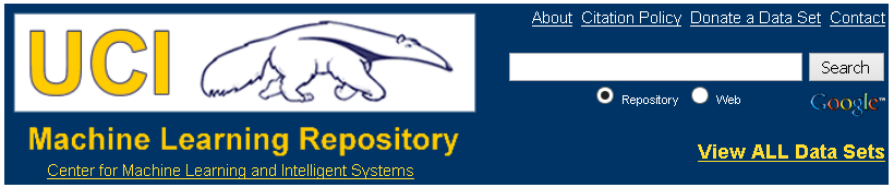





```
> iris.pca <- princomp(iris[,-5])
> plot(iris.pca$scores[,1:2], pch=15, col=as.integer(iris[,5])+1)
> iris.hex <- hexbin(iris.pca$scores[,1:2])
> plot(iris.hex, colramp=col_rb)
```

hexbin for Large Dataset

https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption



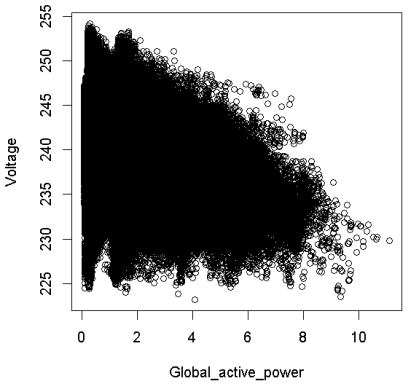
Individual household electric power consumption Data Set

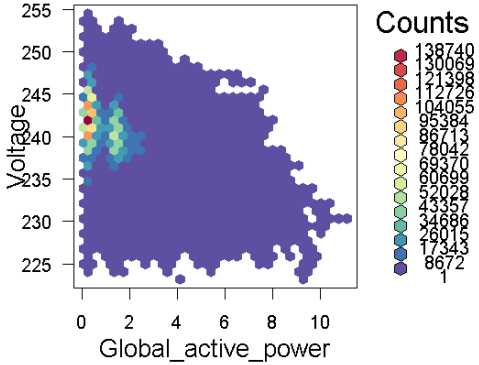
Download: Data Folder, Data Set Description

Abstract: Measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. Different electrical quantities and some sub-metering values are available.

Data Set Characteristics:	Multivariate, Time-Series	**Number of Instances:**	2075259	**Area:**	Physical
Attribute Characteristics:	Real	**Number of Attributes:**	9	**Date Donated**	2012-08-30
Associated Tasks:	Regression, Clustering	**Missing Values?**	Yes	**Number of Web Hits:**	128715

December 2006 and November 2010 (47 months).



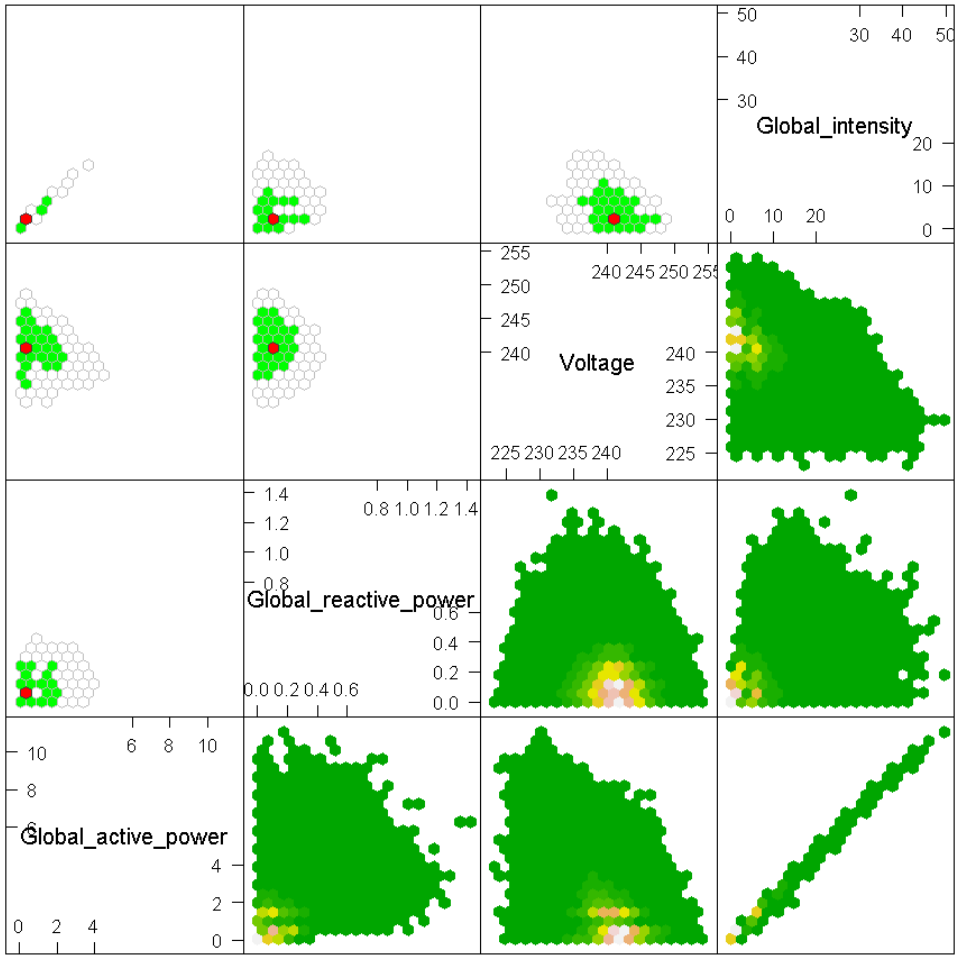



```
> zz <- unz(description="household_power_consumption.zip",
filename="household_power_consumption.txt")
> colC <- c(rep("NULL", 2), "numeric", "NULL", "numeric", rep("NULL", 4))
> power <- read.table(zz, header=T, sep=";", colClasses = colC, na.strings = "?")
> summary(power)
 Global_active_power    Voltage
 Min.   : 0.076      Min.   :223.2
 1st Qu.: 0.308      1st Qu.:239.0
 Median : 0.602      Median :241.0
 Mean   : 1.092      Mean   :240.8
 3rd Qu.: 1.528      3rd Qu.:242.9
 Max.   :11.122      Max.   :254.2
 NA's   :25979       NA's   :25979
```

```
> plot(power)
> power.hex <- hexbin(power)
> plot(power.hex, colramp=col_rb)
```

hexplom {hexbin}: Hexbin Plot Matrices

```
> zz <- unz(description="household_power_consumption.zip", filename="household_power_consumption.txt")
> colC <- c(rep("NULL", 2), rep("numeric", 4), rep("NULL", 3))
> power.num <- read.table(zz, header=T, sep=";", colClasses = colC, na.strings = "?")
> hexplom(power.num, xbins = 20, colramp = terrain.colors, upper.panel = panel.hexboxplot)
```



Scatter Plot Matrix

bigvis: Exploratory data analysis for large datasets (10-100 million observations)

- Hadley Wickham, 2013, Bin-summarise-smooth: a framework for visualising large data. https://github.com/hadley/bigvis

- The aim is to have most operations take less than 5 seconds on commodity hardware, even for 100,000,000 data points.

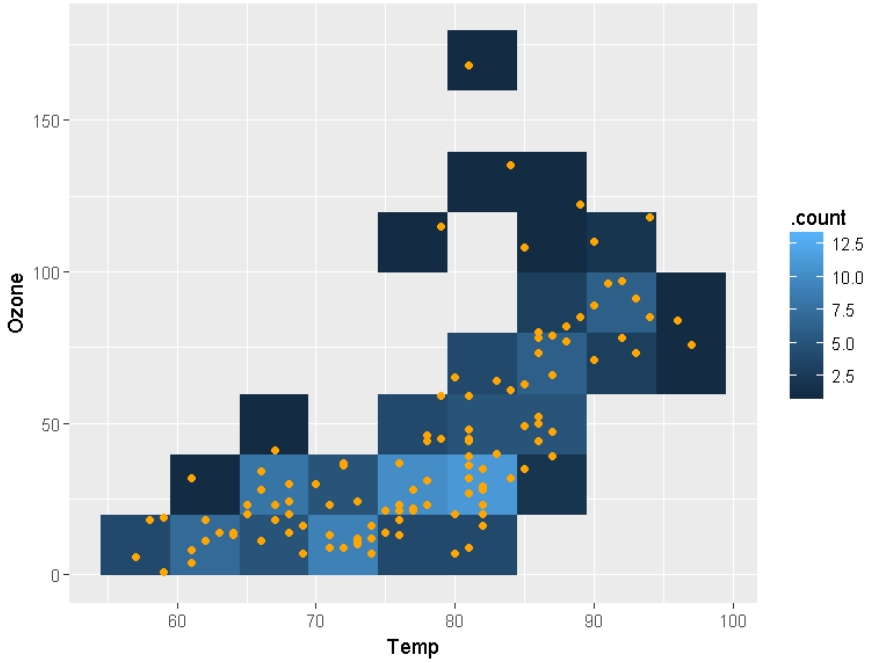
- **Workflow**:
  - **Binning**: binning is an injective mapping from the real numbers to a fixed and finite set of integers. (fixed width binning: fast, easily extended from 1d to nd).
  - **Summarizing**: to collapse the points in each bin into a small number of summary statistics. (count, sum, mean, median or sd)
  - **Smoothing**: if the estimates are rough, you might want to `smooth()`.
  - **Visualizing**: visualize the results with `autoplot {ggplot2}`

- `bigvis` provides outlier removal and smoothing:
  - big data means very rare cases can occur ⇒ outliers may be more of a problem
  - smoothing very important to highlight trends & suppress noise

```
# install.packages("devtools")
devtools::install_github("hadley/bigvis")
```

bigvis applied to a small dataset

```
> library(bigvis)
> library(ggplot2)
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> par(mfrow=c(1, 2))
> hist(airquality$Ozone)
> hist(airquality$Temp)
> #ggplot(data=airquality) +
> #  geom_point(mapping = aes(x = Temp, y = Ozone)
>
> binData <- with(airquality, condense(bin(Ozone, 20), bin(Temp, 5)))
Summarising with count
> binData
   Ozone Temp .count
1     NA   57      4
2     NA   67      2
...
35 129.5   87      1
36 169.5   82      1
> ggplot(data=binData, aes(Temp, Ozone, fill=.count))+
+   geom_tile() +
+   geom_point(data=airquality, aes(fill=NULL), colour="orange")
```



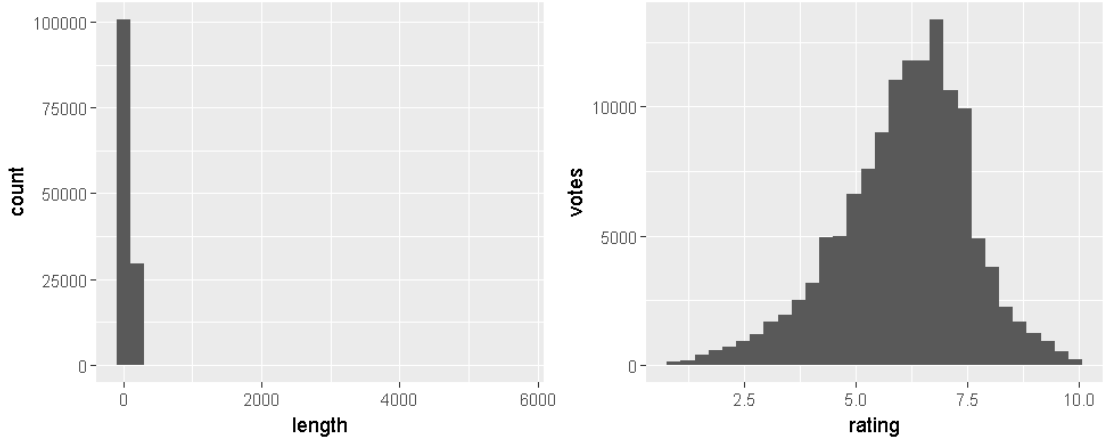
bigvis Applied to a Large Dataset with Outliers

movies {bigvis}: Movie information and user ratings from IMDB.com.

```
> data(movies)
> dim(movies)
[1] 130456     14
> head(movies)
```

```
> head(movies)
                      title year length budget rating votes mpaa Action Animation Comedy Drama Documentary Romance Short
1               Falling Cat 1890      1     NA    5.3    27 <NA>  FALSE     FALSE  FALSE FALSE       FALSE   FALSE FALSE
2          Dickson Greeting 1891      1     NA    5.8   414 <NA>  FALSE     FALSE  FALSE FALSE       FALSE   FALSE FALSE
3                  Fencing 1892      1     NA    5.1    81 <NA>  FALSE     FALSE  FALSE FALSE       FALSE   FALSE FALSE
4            Pauvre Pierrot 1892      4     NA    6.7   204 <NA>  FALSE     FALSE  FALSE FALSE       FALSE   FALSE FALSE
5          Blacksmith Scene 1893      1     NA    6.4   679 <NA>  FALSE     FALSE  FALSE FALSE       FALSE   FALSE FALSE
6 Annabelle Butterfly Dance 1894      1     NA    6.1   212 <NA>  FALSE     FALSE  FALSE FALSE       FALSE   FALSE FALSE
```

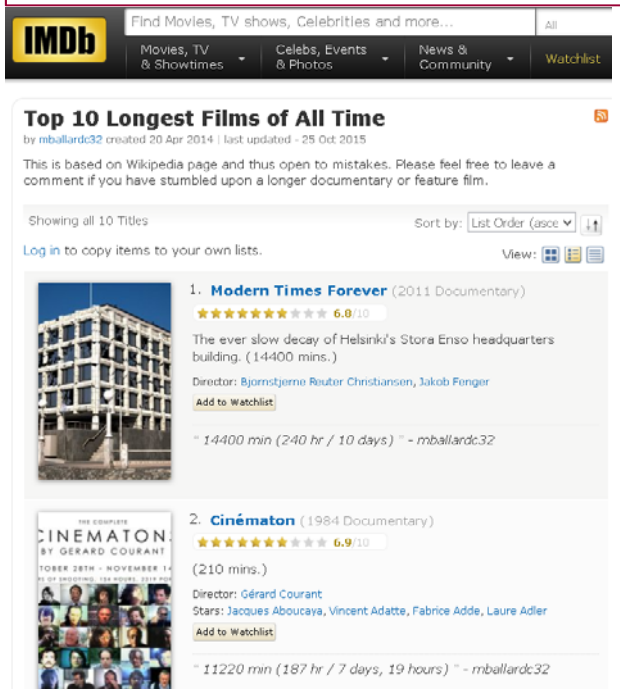
```
> install.packages("gridExtra")
> library(gridExtra)
> g1 <- ggplot(data=movies, aes(length)) + geom_histogram()
> g2 <- ggplot(data=movies, aes(rating)) + geom_histogram() + ylab("votes")
> grid.arrange(g1, g2, nrow=1, ncol=2)
```

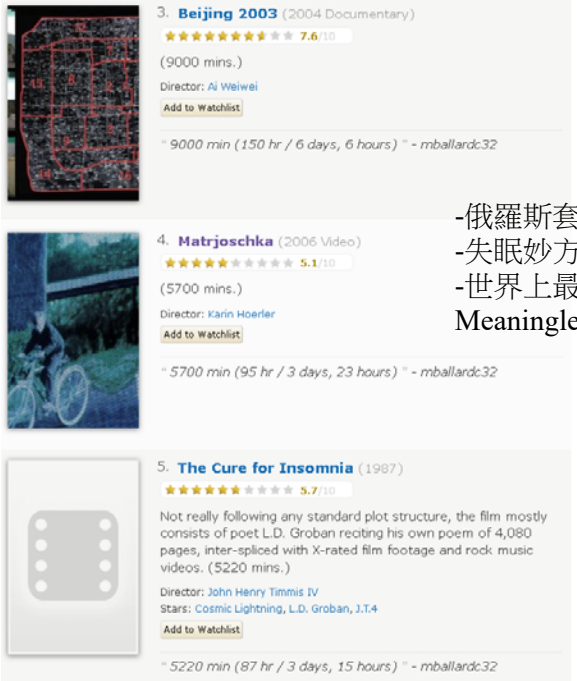


Top 10 longest films

```
> arrange(movies, desc(length))[1:10,] %>% select(title, year, length, rating, votes)
                                                 title year length rating votes
1                                           Matrjoschka 2006   5700    8.5     8
2                                The Cure for Insomnia 1987   5220    5.9   293
3  The Longest Most Meaningless Movie in the World 1970   2880    7.3   143
4                                The Hazards of Helen 1914   1428    6.6    48
5                                                **** 1967   1100    6.9    49
6                                              Resan 1987    873    6.7    40
7                                       Caiyou riji 2008    840    9.2    10
8                          Out 1, noli me tangere 1971    773    7.7   201
9                               Daii jan Napelon 1976    770    7.3   338
10                                 Broken Saints 2003    720    7.5   359
```

See also: 史上超長超無聊電影挑戰人類的耐心極限！
https://read01.com/j6eL0z.html



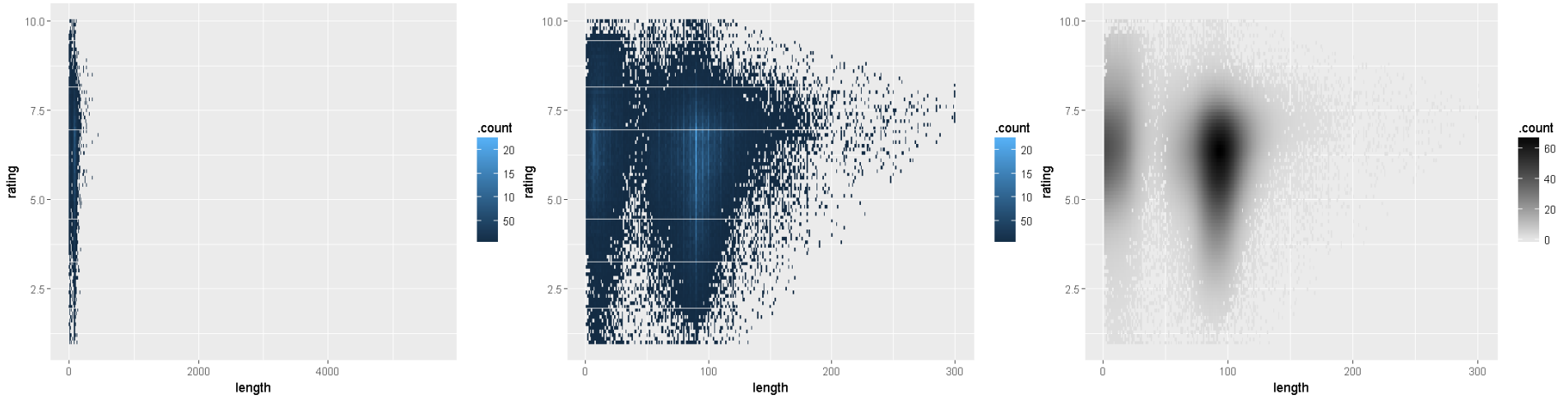


-俄羅斯套 Matrjoschka 5700分鐘/3天23小時
-失眠妙方 The Cure for Insomnia 5220分鐘/3天15小時
-世界上最長最沒意義的電影The Longest Most Meaningless Movie in the World 2880分鐘/2天

bigvis plot with outliers removed

```
> nobin <- 1e4
> binData <- with(movies, condense(bin(length, find_width(length, nobin)),
+                                  bin(rating, find_width(rating, nobin))))
Summarising with count
> ggplot(data=binData, aes(length, rating, fill = .count)) + geom_tile()
> last_plot() %+% peel(binData)
> smoothBinData <- smooth(peel(binData), h=c(20, 1))
> autoplot(smoothBinData)
```

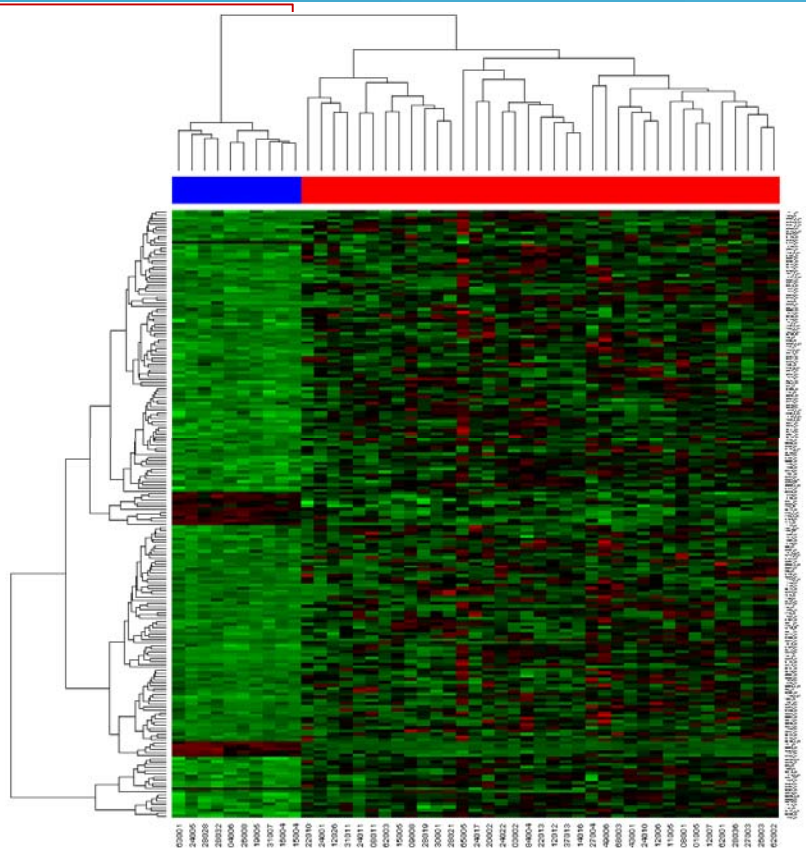
peel {bigvis}: Peel off low density regions of the data.
Description: Keeps specified proportion of data by removing the lowest density regions, either anywhere on the plot, or for 2d, just around the edges.
Usage: `peel(x, keep = 0.99, central = NULL)`



heatmap {stats}

```
> source("https://bioconductor.org/biocLite.R")
> biocLite("ALL")
> library(ALL)
> data(ALL)
> ALL
> str(ALL)
> dim(exprs(ALL))
[1] 12625   128
> exprs(ALL)[1:3, 1:5]
             01005    01010    03002    04006    04007
1000_at   7.597323 7.479445 7.567593 7.384684 7.905312
1001_at   5.046194 4.932537 4.799294 4.922627 4.844565
1002_f_at 3.900466 4.208155 3.886169 4.206798 3.416923
> table(ALL$mol.biol)

ALL1/AF4  BCR/ABL E2A/PBX1      NEG   NUP-98  p15/p16
      10       37        5       74        1        1
> eset <- ALL[, ALL$mol.biol %in%
                c("BCR/ABL", "ALL1/AF4")]
> dim(exprs(eset))
[1] 12625    47
> f <- factor(as.character(eset$mol.biol))
> eset.p <- apply(exprs(eset), 1, function(x) t.test(x ~ f)$p.value)
> selected.eset <- eset[eset.p < 0.00001, ]
> dim(selected.eset)
Features  Samples
     200       47
> ma.col <- colorRampPalette(c("green", "black", "red"))(200)
> var.col <- ifelse(f=="ALL1/AF4", "blue", "red")
> heatmap(exprs(selected.eset), col=ma.col, ColSideColors=var.col,
          scale="row")
```



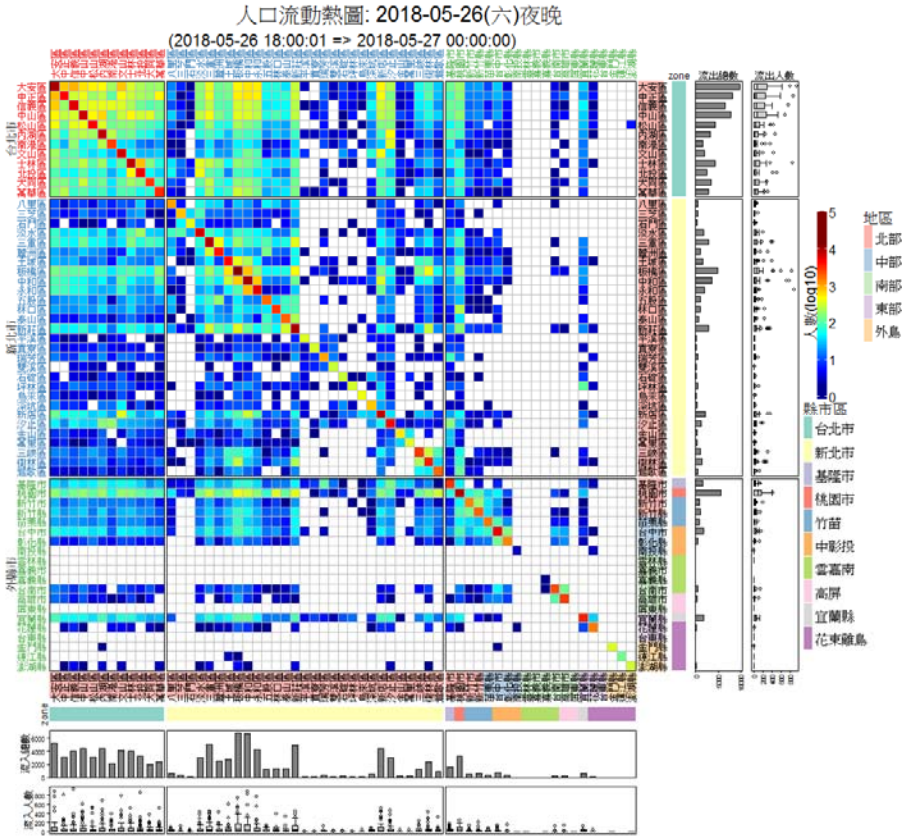
ComplexHeatmap

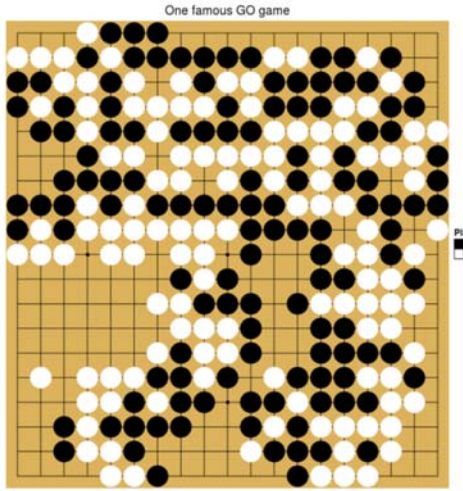
https://jokergoo.github.io/ComplexHeatmap-reference/book/

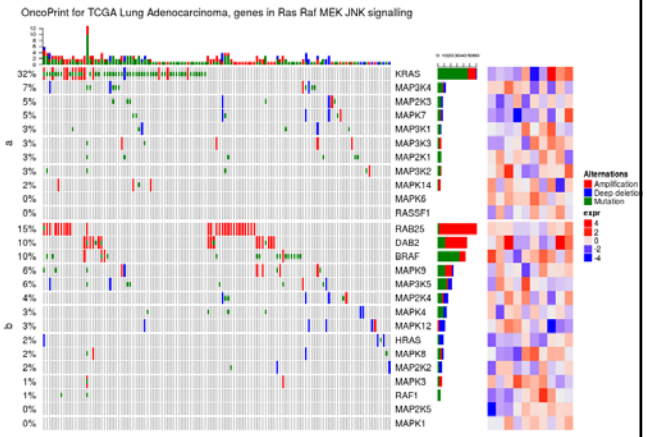
http://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html

Zuguang Gu, Roland Eils, Matthias Schlesner, Complex heatmaps reveal patterns and correlations in multidimensional genomic data, Bioinformatics, Volume 32, Issue 18, 15 September 2016, Pages 2847–2849.

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
> BiocManager::install("ComplexHeatmap")
> library(ComplexHeatmap)
> Heatmap(exprs(selected.eset))
```







visualize multiple genomic alteration events by heatmap

Example: ComplexHeatmap

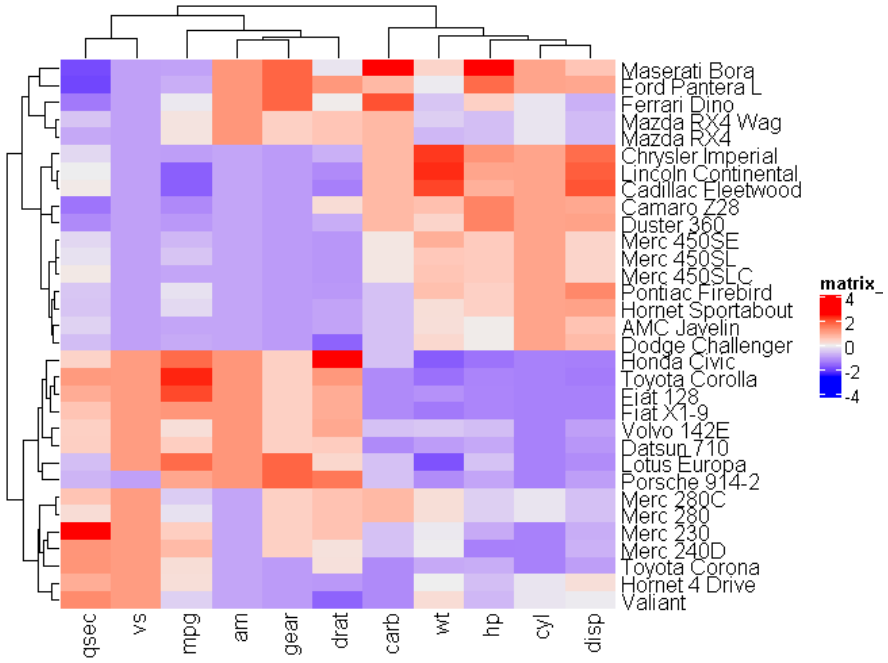
```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
> str(mtcars)
'data.frame':    32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
> mtcars.df <- scale(mtcars)
> class(mtcars.df)
[1] "matrix" "array"
> library(ComplexHeatmap)
> ? Heatmap
```

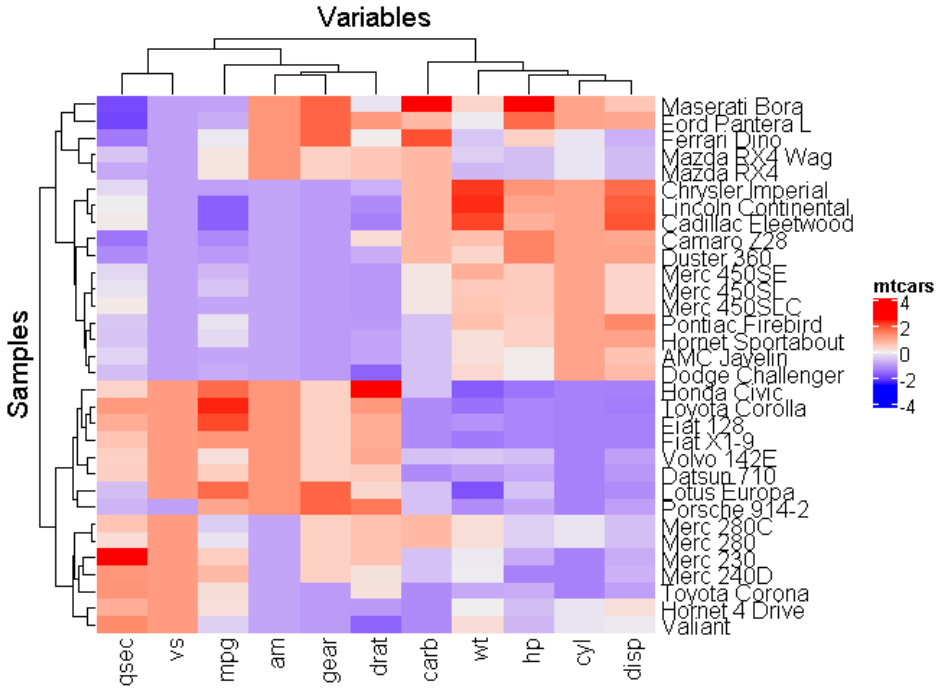
Reference
https://www.datanovia.com/en/lessons/heatmap-in-r-static-and-interactive-visualization/

ComplexHeatmap: 列、行標題

```
Heatmap(mtcars.df)
```

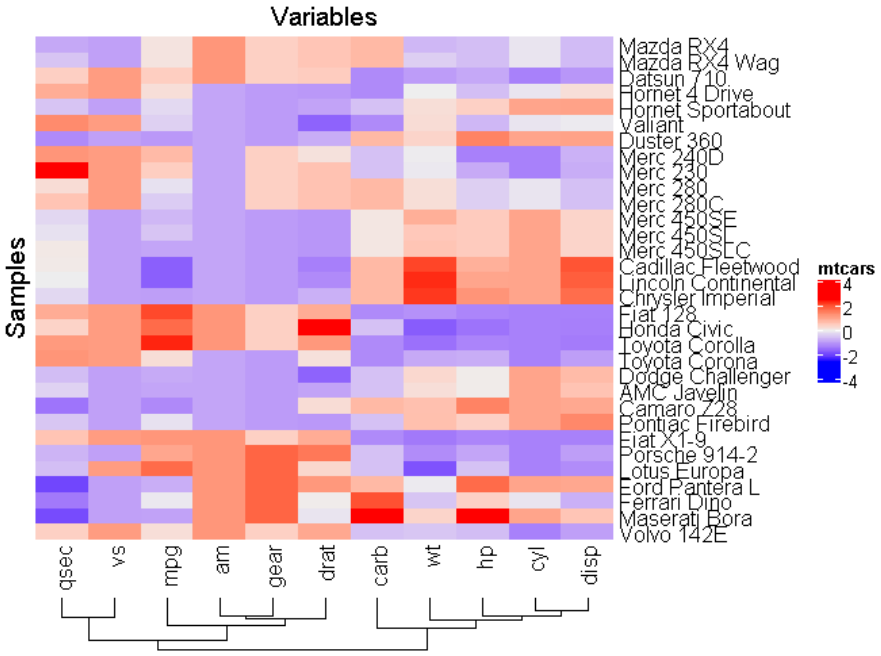
```
Heatmap(mtcars.df, name = "mtcars",
        row_title = "Samples",
        column_title = "Variables")
```



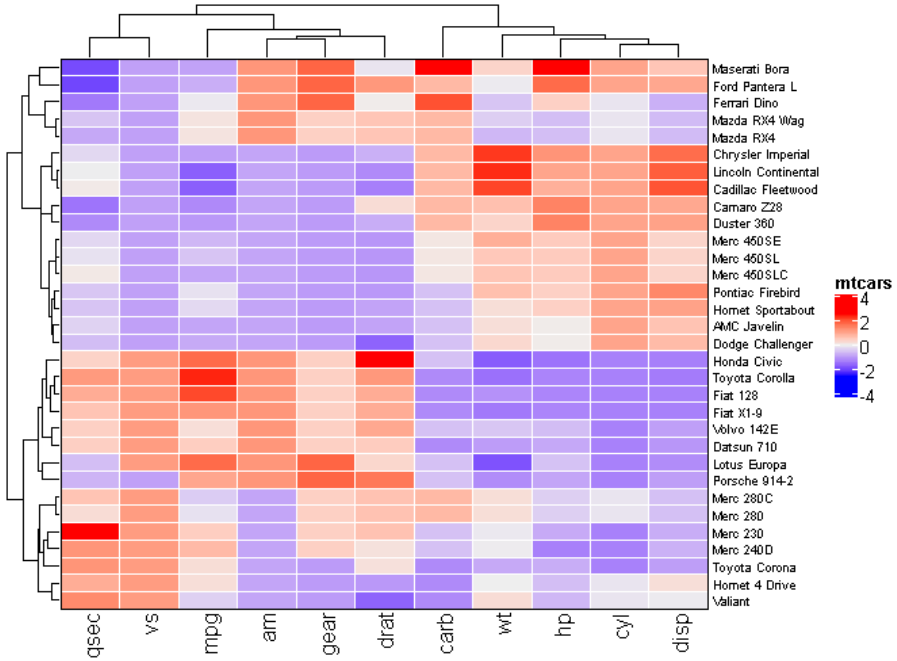


ComplexHeatmap:
樹狀圖位置、格線及字體大小

```
Heatmap(mtcars.df, name = "mtcars",
        row_title = "Samples",
        column_title = "Variables",
        cluster_rows = FALSE,
        column_dend_side = "bottom")
```

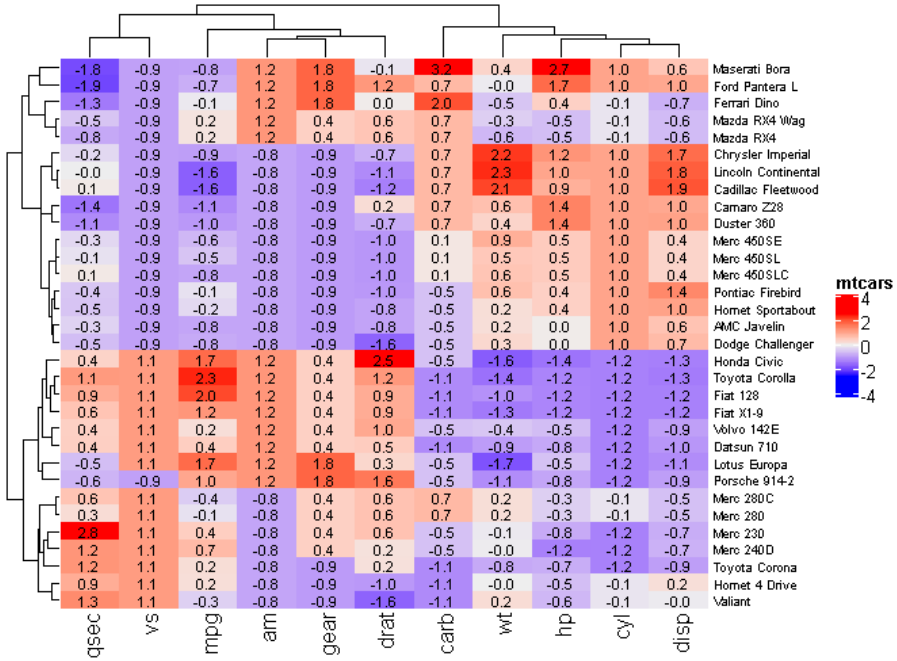



```
Heatmap(mtcars.df, name = "mtcars",
        rect_gp = gpar(col = "white", lwd = 1),
        border = TRUE,
        row_names_gp = gpar(fontsize = 7))
```

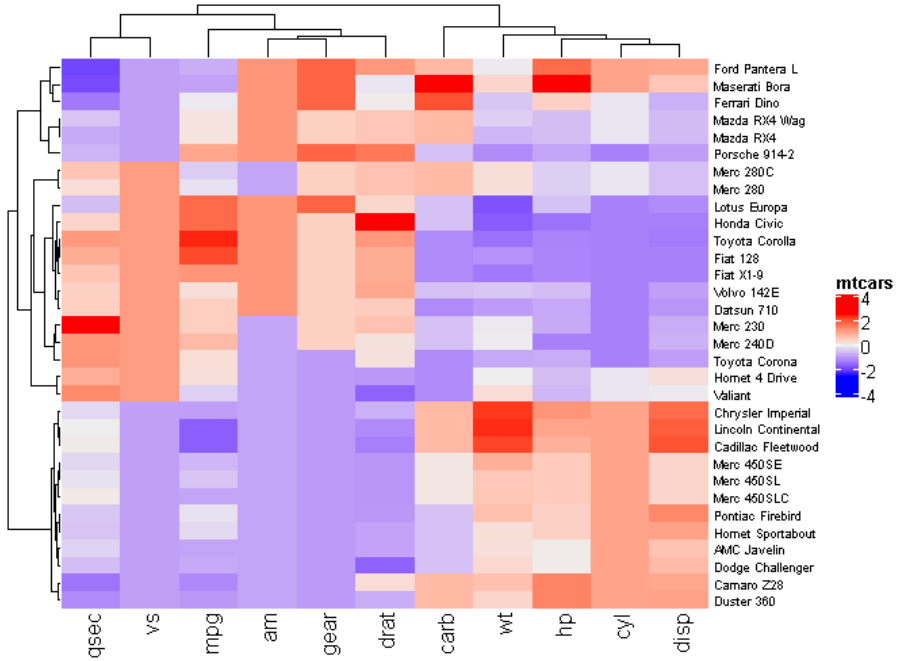


ComplexHeatmap: 顯示數字、指定分群法

```
Heatmap(mtcars.df, name = "mtcars",
        cell_fun = function(j, i, x, y, width, height, fill) {
          grid.text(sprintf("%.1f", mtcars.df[i, j]), x, y,
                    gp = gpar(fontsize = 8))},
        row_names_gp = gpar(fontsize = 7))
```



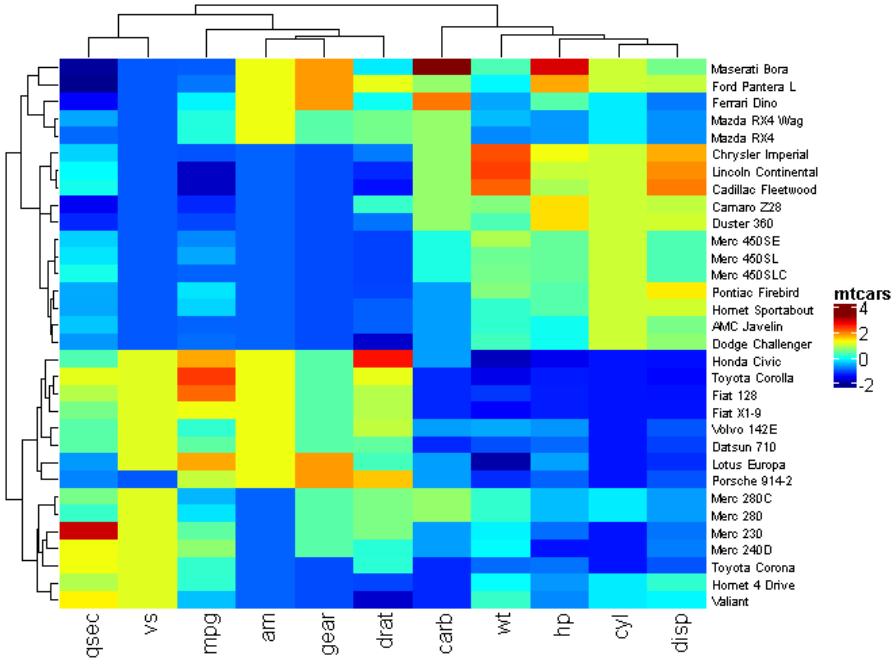

```
Heatmap(mtcars.df, name = "mtcars",
        clustering_distance_rows = "pearson",
        clustering_method_rows = "average",
        row_names_gp = gpar(fontsize = 7))
```

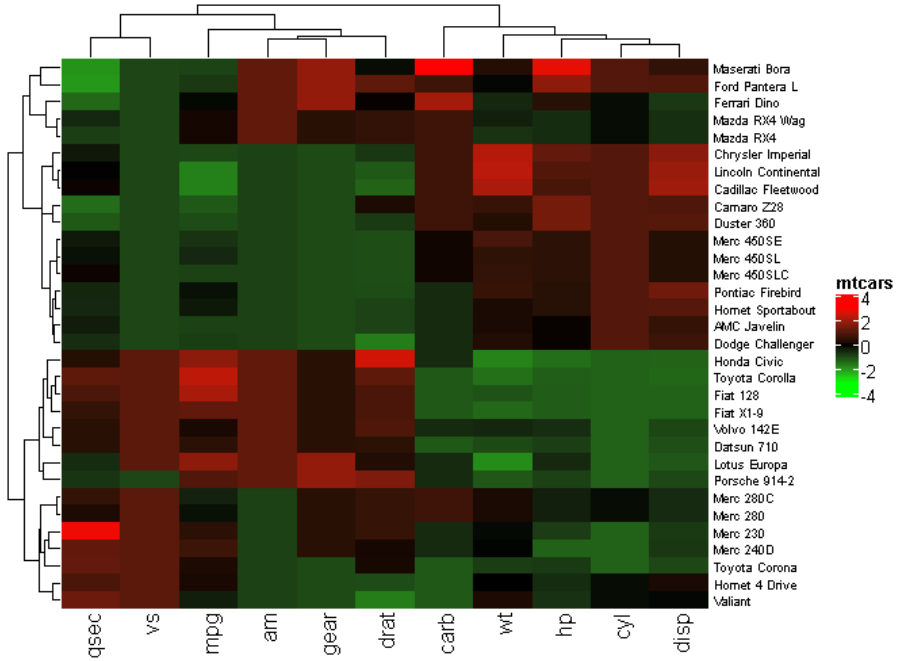


ComplexHeatmap: 自定色階

```
library(fields)
Heatmap(mtcars.df, name = "mtcars",
        col = tim.colors(),
        row_names_gp = gpar(fontsize = 7))
```

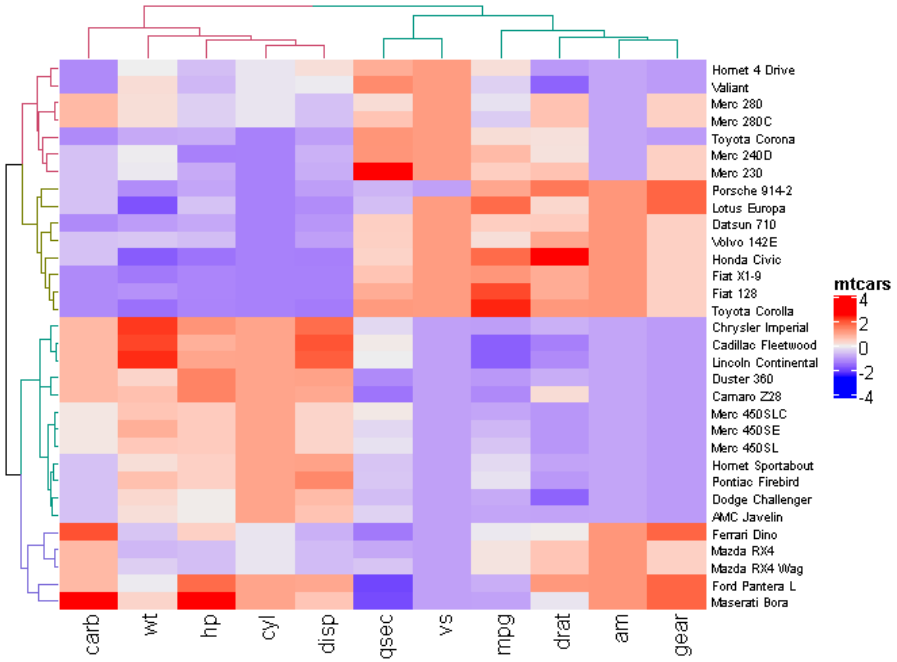
```
library(circlize) # Circular Visualization
mycolorRamp <- colorRamp2(c(-3, 0, 3), c("green", "black", "red"))
Heatmap(mtcars.df, name = "mtcars",
        col = mycolorRamp,
        row_names_gp = gpar(fontsize = 7))
```



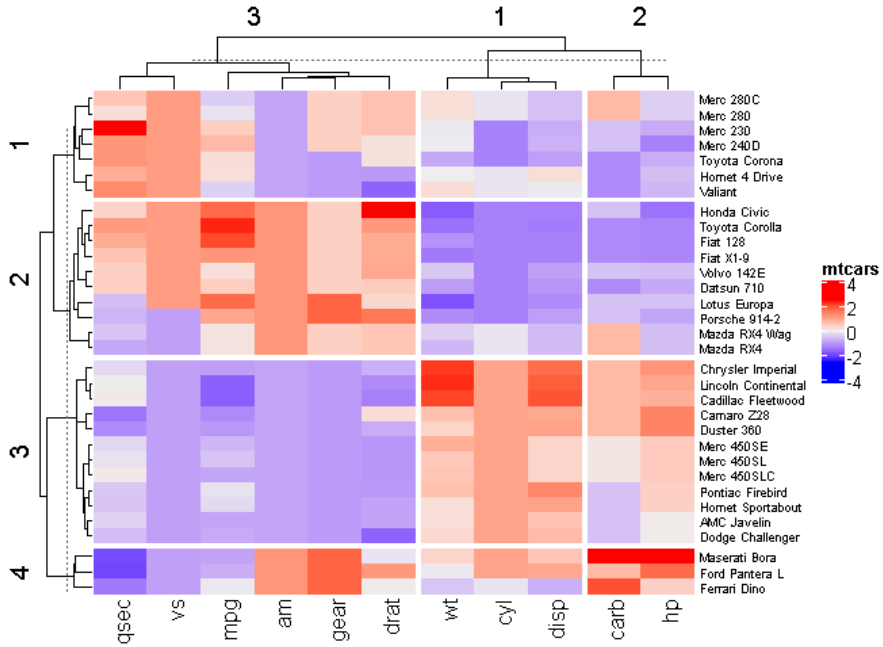


ComplexHeatmap: 以顏色標記樹狀圖、熱圖分割

```
library(dendextend)
row.dend <- hclust(dist(mtcars.df))
column.dend <- hclust(dist(t(mtcars.df)))
Heatmap(mtcars.df, name = "mtcars",
        cluster_rows = color_branches(row.dend, k = 4),
        cluster_columns = color_branches(column.dend, k = 2),
        row_names_gp = gpar(fontsize = 7))
```



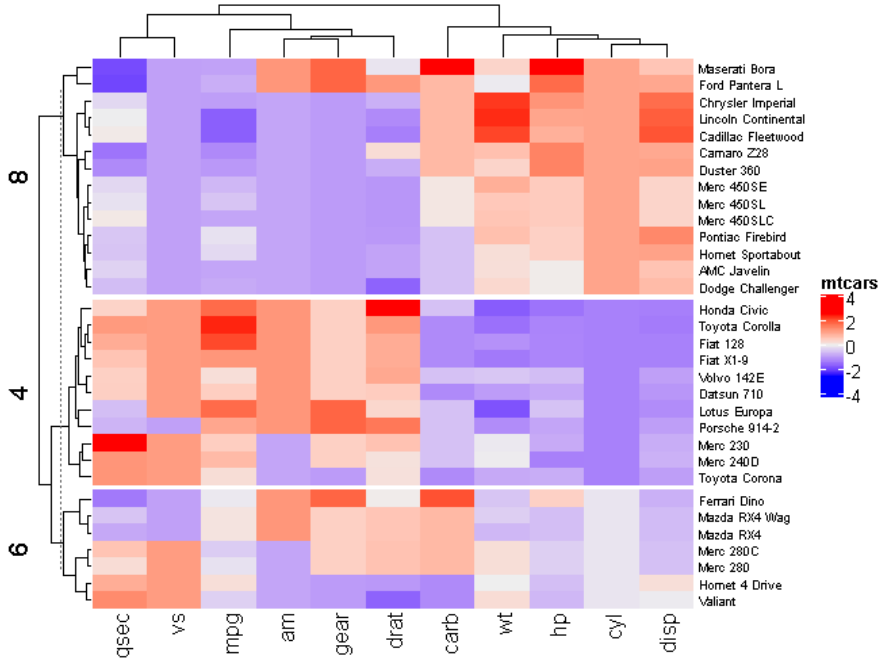

```
# plit the dendrogram using k-means
Heatmap(mtcars.df, name = "mtcars",
        row_km = 4,
        column_km = 3,
        row_names_gp = gpar(fontsize = 7))
```

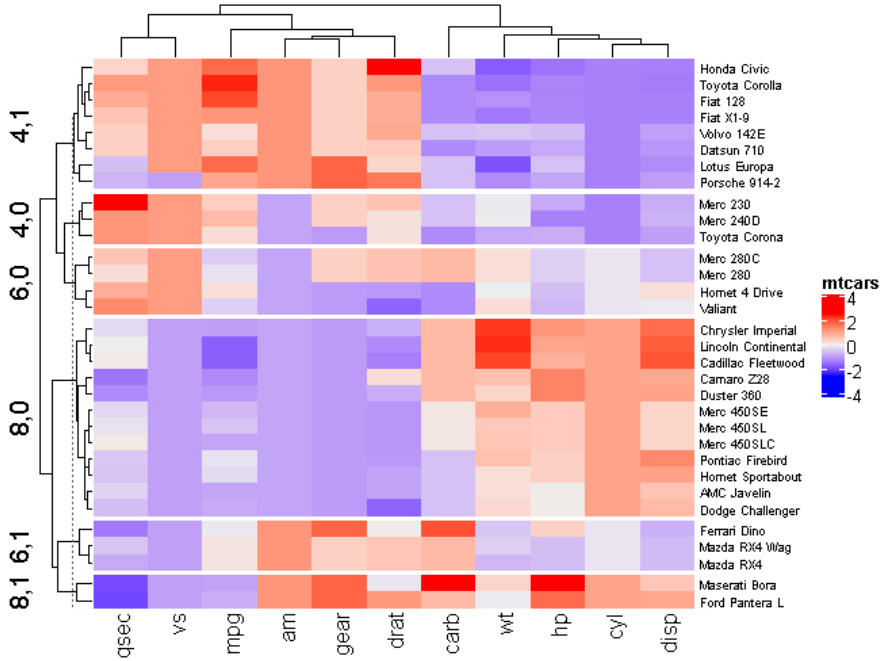


ComplexHeatmap: 依據類別變數分割熱圖

```
# split by a vector specifying rowgroups
Heatmap(mtcars.df, name = "mtcars",
        row_split = mtcars$cyl,
        row_names_gp = gpar(fontsize = 7))
```

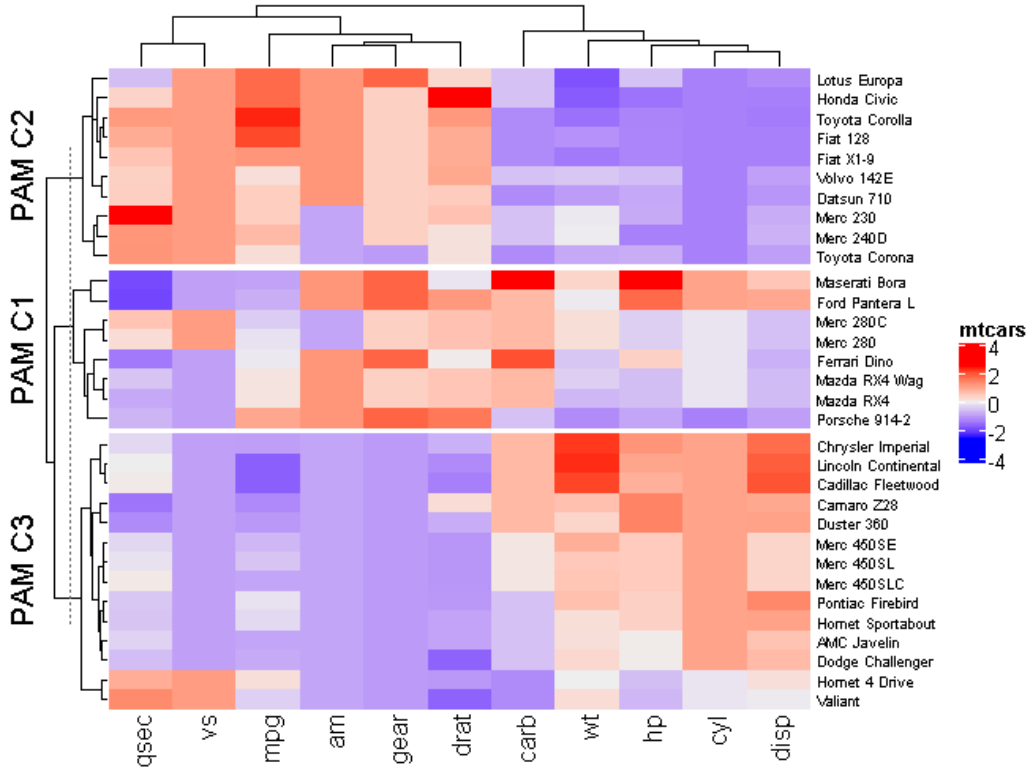
```
# Split by combining multiple variables
Heatmap(mtcars.df, name ="mtcars",
        row_split = data.frame(cyl = mtcars$cyl, am = mtcars$am),
        row_names_gp = gpar(fontsize = 7))
```





ComplexHeatmap: 依據群集分析法分割熱圖

```
# install.packages("cluster")
library("cluster")
set.seed(12345)
pa <- pam(mtcars.df, k = 3)
Heatmap(mtcars.df, name = "mtcars",
        row_split = paste0("PAM C", pa$clustering),
        row_names_gp = gpar(fontsize = 7))
```

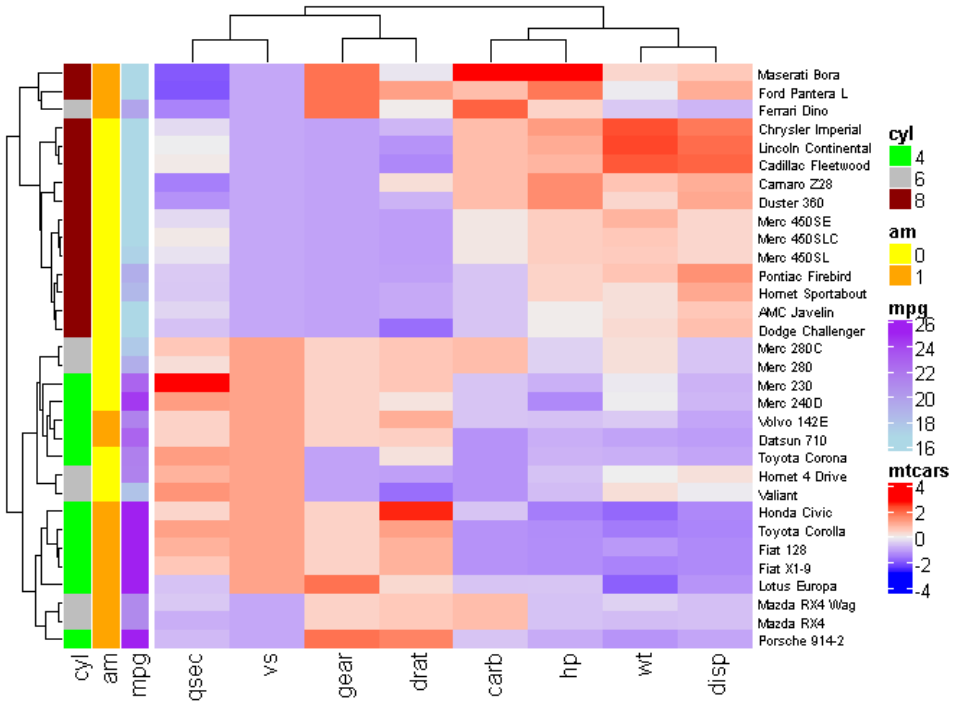


ComplexHeatmap: 以類別變數標記註解並指定顏色

```
# HeatmapAnnotation for "top_annotation", "bottom_annotation"
# rowAnnotation for "left_annotation", "right_annotation"
var.colors <- list(cyl = c("4" = "green", "6" = "gray", "8" = "darkred"),
                   am = c("0" = "yellow", "1" = "orange"),
                   mpg = colorRamp2(c(17, 25), c("lightblue", "purple")))

ha <- rowAnnotation(cyl = mtcars$cyl, am = mtcars$am, mpg = mtcars$mpg,
                    col = var.colors)

myvars <- colnames(mtcars.df) %in% c("cyl", "am", "mpg")
Heatmap(mtcars.df[, !myvars], name = "mtcars",
        left_annotation = ha,
        row_names_gp = gpar(fontsize = 7))
```

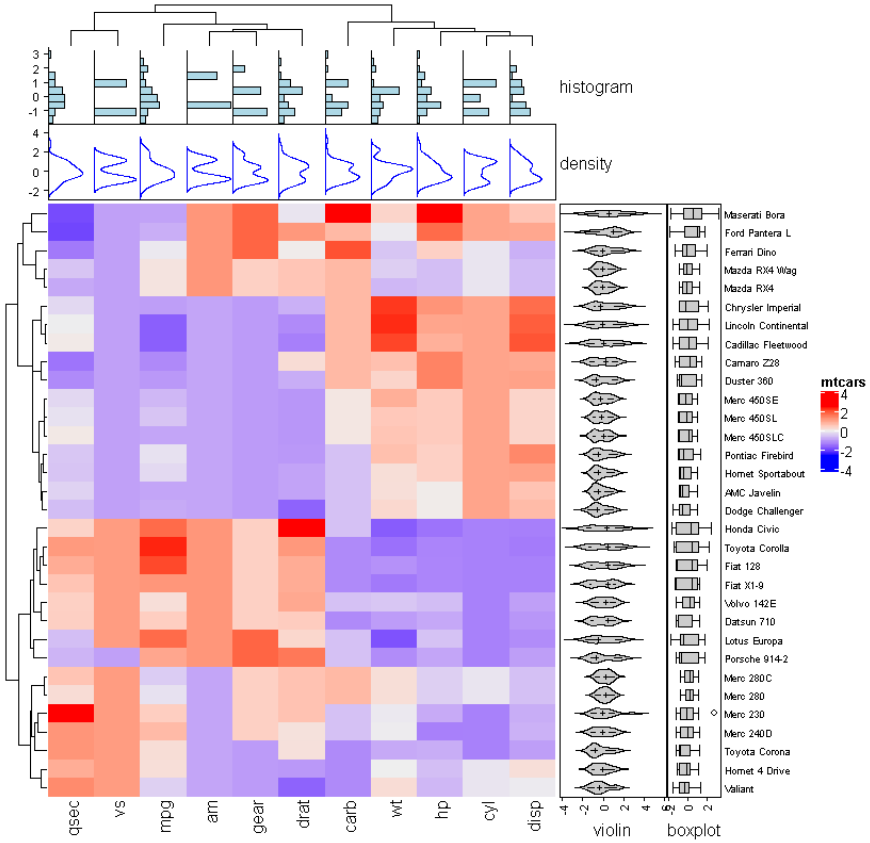


ComplexHeatmap: 標記統計圖為註解

```
# anno_points, anno_barplot, anno_boxplot, anno_density, anno_histogram
h <- anno_histogram(mtcars.df, gp = gpar(fill = "lightblue"))
d <- anno_density(mtcars.df, type = "line", gp = gpar(col = "blue"))
ha.top <- HeatmapAnnotation(histogram = h, density = d, height = unit(3.8, "cm"))

v <- anno_density(mtcars.df, type = "violin", which = "row")
b <- anno_boxplot(mtcars.df, which = "row")
ha.right <- HeatmapAnnotation(violin = v, boxplot = b, which = "row", width = unit(4, "cm"))

Heatmap(mtcars.df, name = "mtcars",
        top_annotation = ha.top,
        right_annotation = ha.right,
        row_names_gp = gpar(fontsize = 7))
```



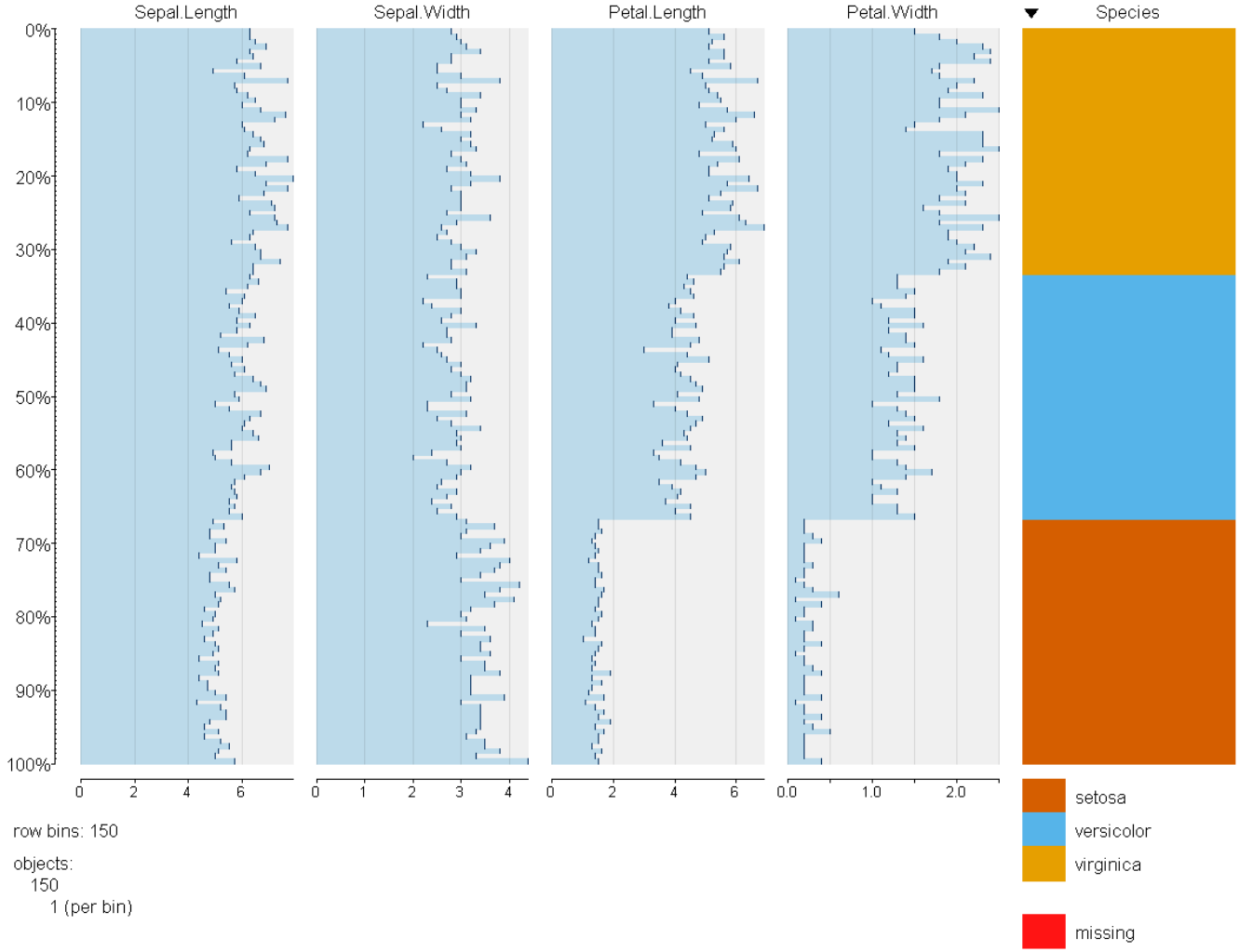
tabplot: Tableplot, a Visualization of Large Datasets

- A tableplot is a visualisation of a (large) dataset with a dozen of variables, both numeric and categorical.
  - Each column represents a variable and each row bin is an aggregate of a certain number of records.
  - Numeric variables are visualized as bar charts, and
  - categorical variables as stacked bar charts. Missing values are taken into account.
  - Also supports large '**ffdf**' datasets from the '**ff**' package.
  - https://github.com/mtennekes/tabplot
  - https://cran.r-project.org/web/packages/tabplot/vignettes/tabplot-vignette.html
- Tennekes, M., Jonge, E. de, Daas, P.J.H. (2013) Visualizing and Inspecting Large Datasets with Tableplots, Journal of Data Science 11 (1), 43-58.

```
tableplot(dat, select, subset = NULL, sortCol = 1, decreasing = TRUE,
  nBins = 100, from = 0, to = 100, nCols = ncol(dat), sample = FALSE,
  sampleBinSize = 1000, scales = "auto", numMode = "mb-sdb-ml",
  max_levels = 50, pals = list("Set1", "Set2", "Set3", "Set4"),
  change_palette_type_at = 20, rev_legend = FALSE, colorNA = "#FF1414",
  colorNA_num = "gray75", numPals = "OrBu", limitsX = NULL,
  bias_brokenX = 0.8, IQR_bias = 5, select_string = NULL,
  subset_string = NULL, colNames = NULL, filter = NULL, plot = TRUE,
  ...)
```

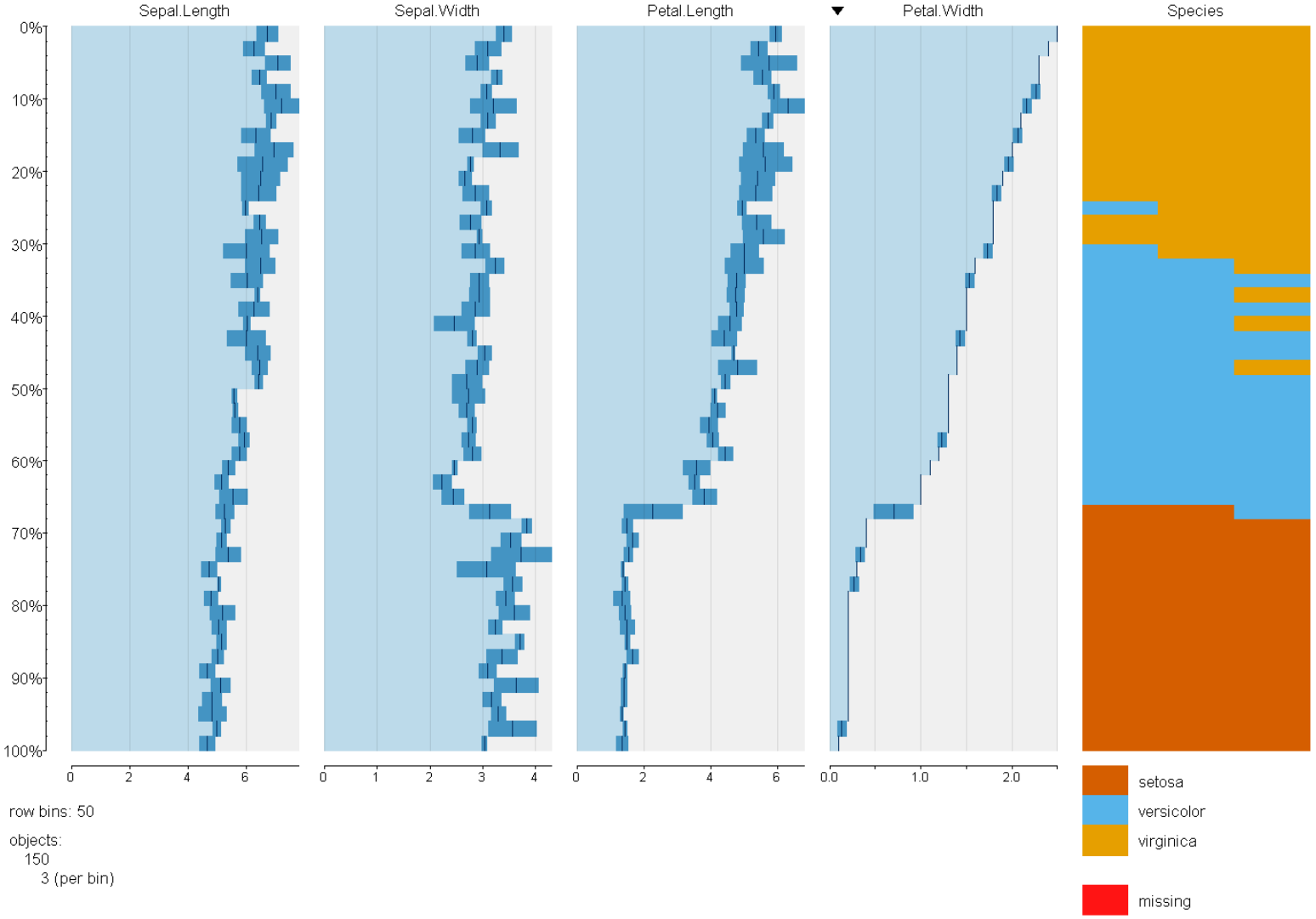
tableplot(iris, nBins=150, sortCol=5)

```
> install.packages("tabplot")
> library(tabplot)
> tableplot(iris, nBins=150, sortCol=5)
```



tableplot(iris, nBins=50, sortCol=4)

```
> tableplot(iris, nBins=50, sortCol=4)
```

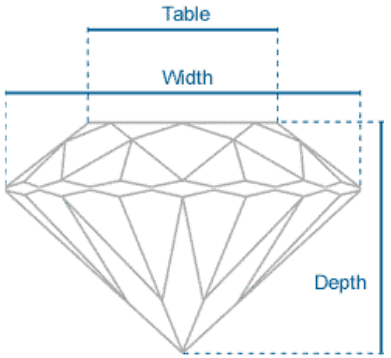


tableplot(diamonds)

```
> require(ggplot2)
> data(diamonds)
> dim(diamonds)
[1] 53940    10
> head(diamonds)
# A tibble: 6 × 10
  carat       cut color clarity depth table price     x     y     z
  <dbl>     <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.23     Ideal     E     SI2  61.5    55   326  3.95  3.98  2.43
2  0.21   Premium     E     SI1  59.8    61   326  3.89  3.84  2.31
3  0.23      Good     E     VS1  56.9    65   327  4.05  4.07  2.31
4  0.29   Premium     I     VS2  62.4    58   334  4.20  4.23  2.63
5  0.31      Good     J     SI2  63.3    58   335  4.34  4.35  2.75
6  0.24 Very Good     J    VVS2  62.8    57   336  3.94  3.96  2.48
> tableplot(diamonds)
```

Details

- price. price in US dollars (\$326--\$18,823)
- carat. weight of the diamond (0.2--5.01)
- cut. quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- colour. diamond colour, from J (worst) to D (best)
- clarity. a measurement of how clear the diamond is (I1 (worst), SI1, SI2, VS1, VS2, VVS1, VVS2, IF (best))
- x. length in mm (0--10.74)
- y. width in mm (0--58.9)
- z. depth in mm (0--31.8)
- depth. total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43--79)
- table. width of top of diamond relative to widest point (43--95)

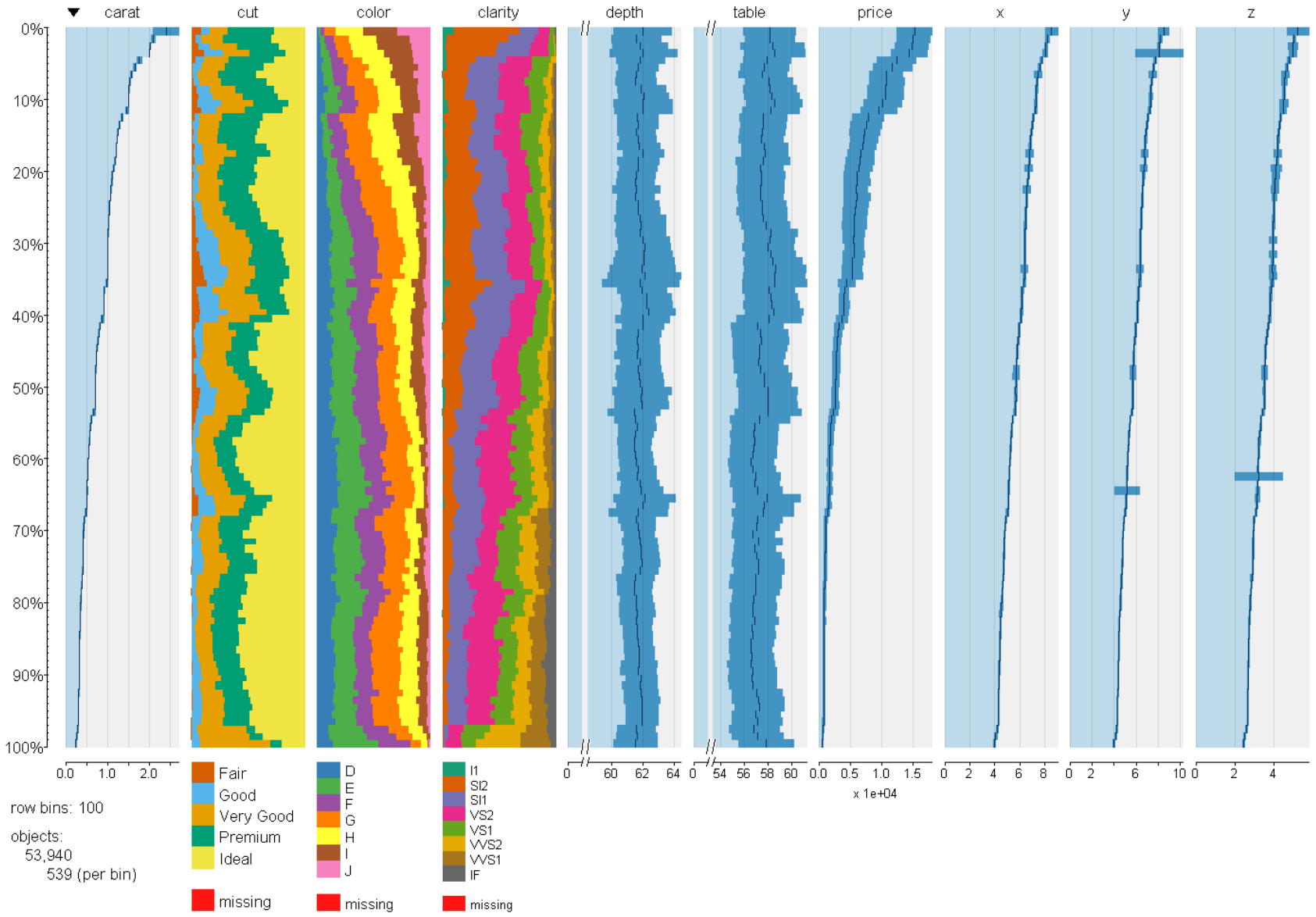


http://www.lumeradiamonds.com/diamond-education/diamond-cut

http://docs.ggplot2.org/0.9.3.1/diamonds.html

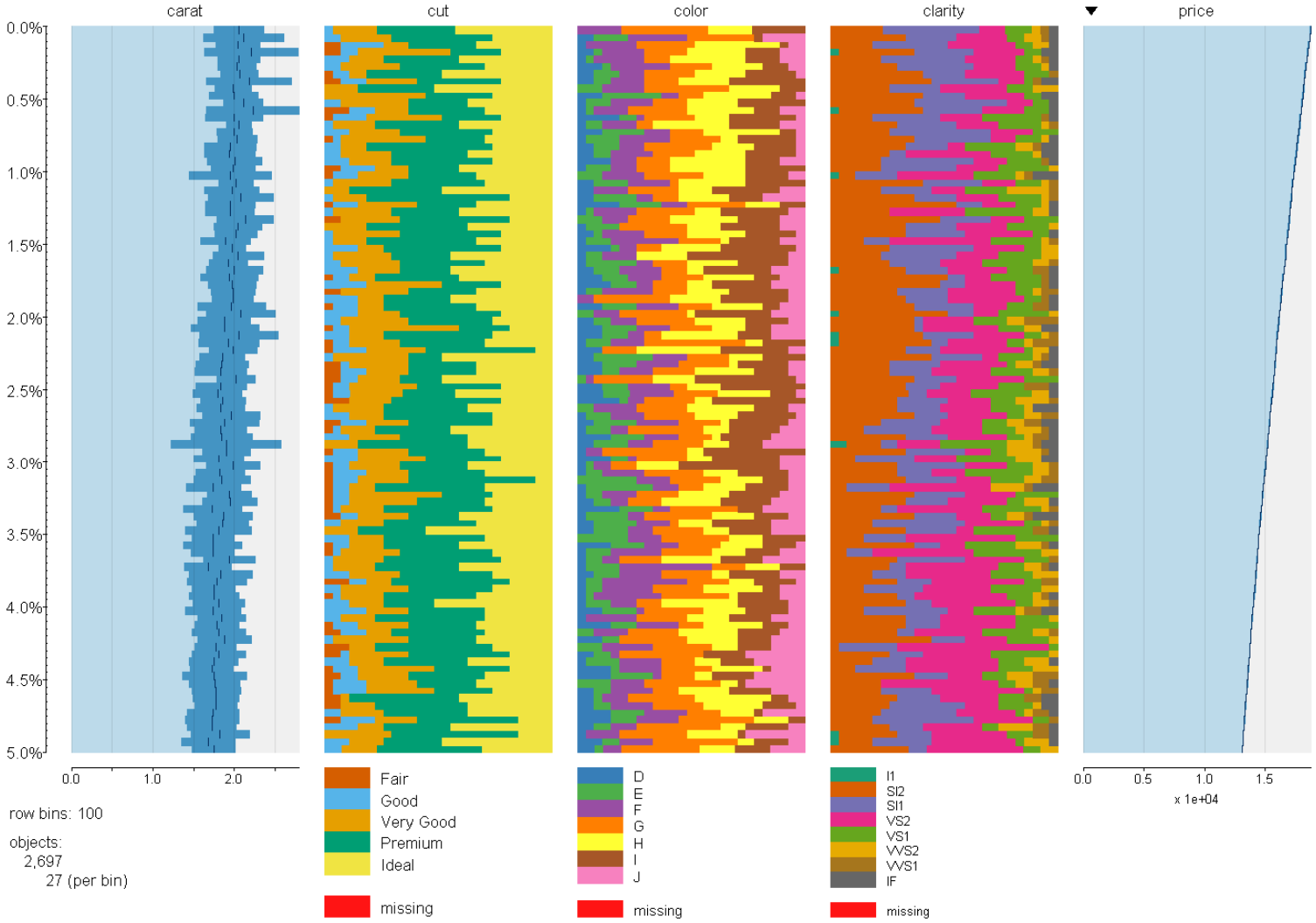
http://yourdiamondteacher.com/diamond-4cs/cut/

tableplot(diamonds)



Most Expensive Diamonds

```
> tableplot(diamonds, select=c(carat, cut, color, clarity, price),
+           sortCol=price, from=0, to=5)
```



ggplot2.SparkR: Rebooting ggplot2 for Scalable Big Data Visualization

ggplot2.SparkR

Welcome to ggplot2.SparkR

View project on GitHub

Overview

ggplot2.SparkR is an R package for scalable visualization of big data represented in Spark DataFrame.

It is an extension to the original ggplot2 package and can seamlessly handle both R data.frame and Spark DataFrame with no modifications to the original API.

Installation

SparkR Installation

Build Spark

Build Spark with Maven and include the `-PsparkR` profile to build the R package. For example to use the default Hadoop versions you can run

```
build/mvn -DskipTests -Psparkr package
```

Using SparkR from RStudio

If you wish to use SparkR from RStudio or other R frontends you will

Download .zip file

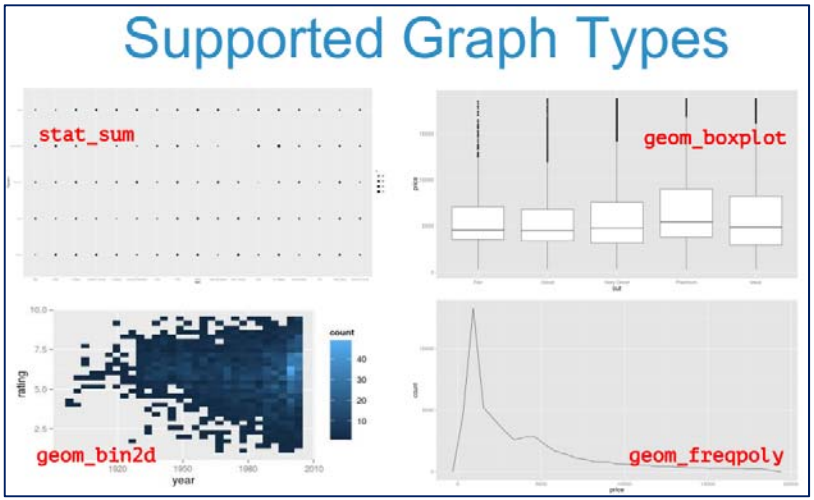
Download .tar.gz file

is maintained by SKKU-SKT.

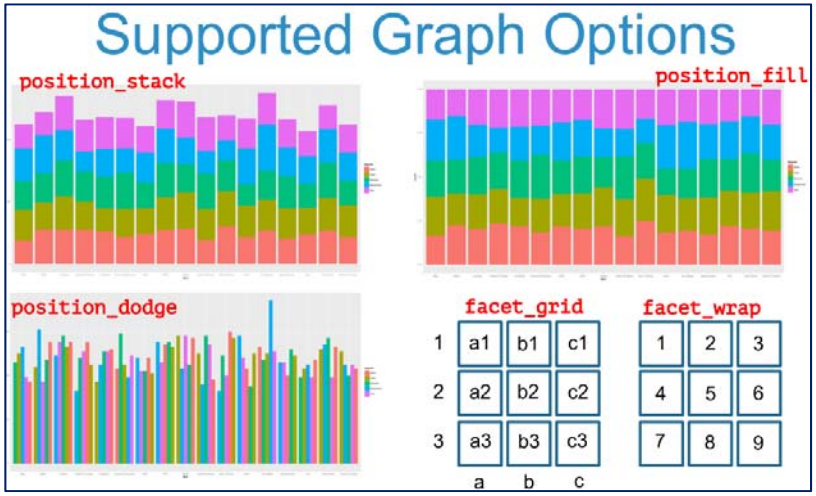
This page was generated by GitHub Pages using the Architect theme by Jason Long.

- Home
- Hello ggplot2.SparkR
- Supported Plot Types

Supported Graph Types

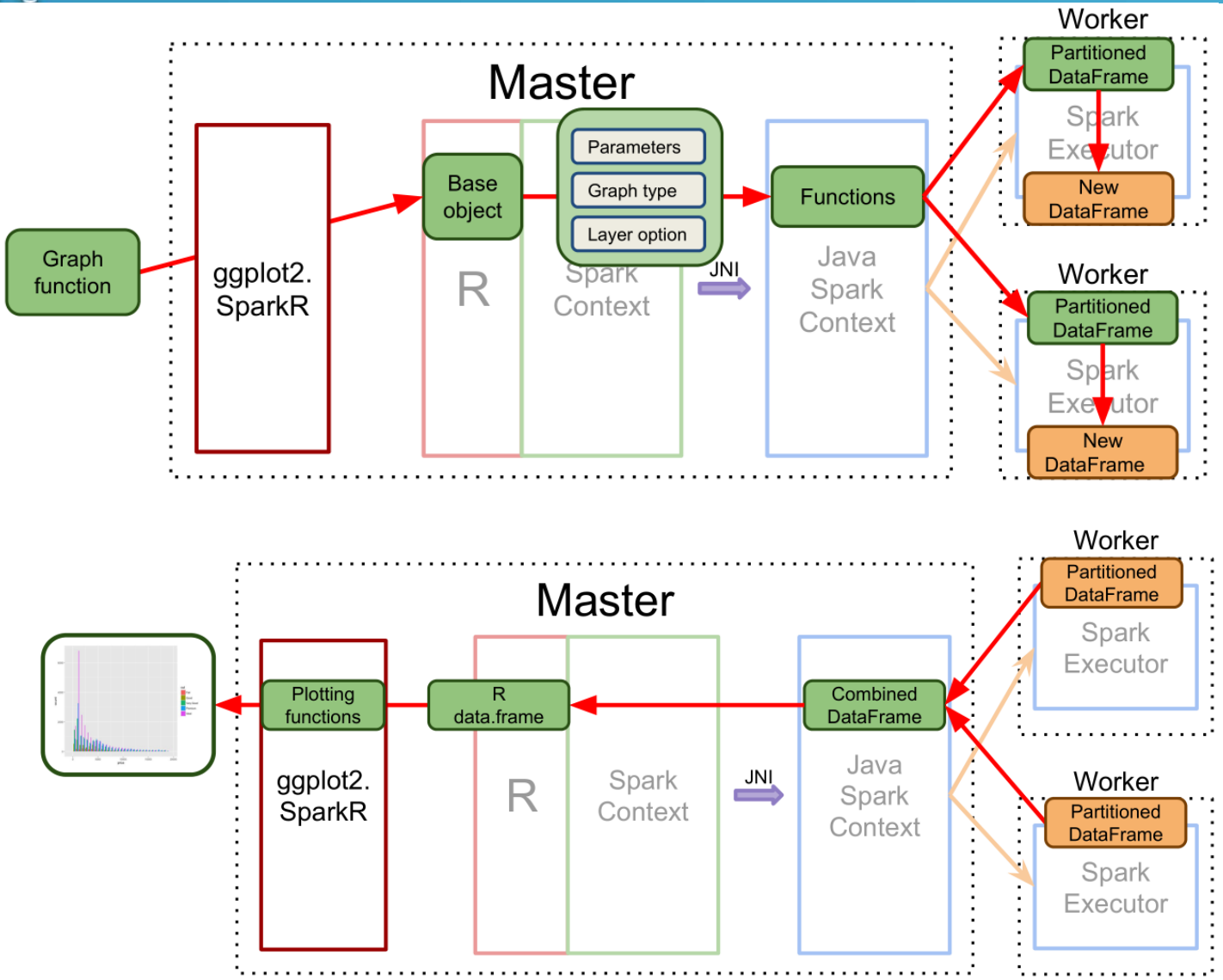


Supported Graph Options



http://skku-skt.github.io/ggplot2.SparkR/

ggplot2.SparkR: Data Flow



http://skku-skt.github.io/ggplot2.SparkR/

Symbolic Data Analysis (Billard and Diday, JASA 2003)

Symbolic data table (intervals)

The classical data table

	X_1	X_2	$\cdots$	X_3	$\cdots$	X_p
s_1	x_{11}	x_{12}	$\cdots$	x_{13}	$\cdots$	x_{1p}
s_2	x_{21}	x_{22}	$\cdots$	x_{23}	$\cdots$	x_{2p}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
s_{n_1}	$x_{n_1 1}$	$x_{n_1 2}$	$\cdots$	$x_{n_1 3}$	$\cdots$	$x_{n_1 p}$
s_{n_1+1}	$x_{(n_1+1)1}$	$x_{(n_1+1)2}$	$\cdots$	$x_{(n_1+1)3}$	$\cdots$	$x_{(n_1+1)p}$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
s_i	x_{i1}	x_{i2}	$\cdots$	x_{ij}	$\cdots$	x_{ip}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
s_N	x_{N1}	x_{N2}	$\cdots$	x_{N3}	$\cdots$	x_{Np}

aggregate

aggregate

Symbolic data table (histograms)

Symbolic Variable

- Numerical
  - Single-valued (real or integer) → Multi-valued (finite set of real)
  - Interval-valued $[a_i, b_i]$
  - Modal (η_i, π_i) → Multi-valued Modal $\{(\eta_{i1}, \pi_{i1}),\ldots, (\eta_{ik}, \pi_{ik})\}$ → Histogram $\{(I_{i1}(p_1),\ldots, (I_{ik}(p_k)\}$
  - Function (pf, cdf) → Time Series → Stochastic Process → Model
- Categorical
  - Single-valued (nominal or ordinal) → Multi-valued (finite set of categories)
  - Modal (ξ_i, π_i) → Multi-valued Modal $\{(\xi_{i1}, \pi_{i1}),\ldots, (\xi_{ik}, \pi_{ik})\}$

π_i: weight, frequency, probability
η_i: numerical
ξ_i: categorical

graphics.SDA

the R base graphics package companion for symbolic data analysis

The R Graphics Package Companion for Symbolic Data Analysis

Documentation for package 'graphics.SDA' version 0.0.0.9

- DESCRIPTION file.
- Code demos. Use demo() to run them.

Help Pages

boxplot.i	Box Plots For Interval Data
boxplot.sbs.i	The Side-by-Side Box Plots For Interval Data
cbind.h	Combine HistData Objects by Rows or Columns
cbind.i	Combine IntervalData Objects by Rows or Columns
get.subset	The Subset of the Histogram Data
hM.size	The Dimensions of the Histogram Data
image.i	The Image Plot For Interval Data
plot.index.i	The Index Plot For Interval Data

Not Yet Released!

plot.index.i {graphics.SDA}

The Index Plot For Interval Data

Description

The index plot for one sample of the interval data

Usage

```
## S3 method for class 'index.i'
plot(idata, vertical = FALSE, type = "seg", align = "d",
  fill.col = "lightcyan", col = "black", cex = 0.85,
  show.mean.value = F, ...)
```

Arguments

idata	one sample of an IntervalData object, or the data matirx with the (min, max) format
vertical	logical.
type	"seg": segments, "rect": rectangular
align	the direction (x-axis or y-axis) of the indics default(d), initial(i), left(l), right(r).
fill.col	the filled color of the interval bars
col	the color of the interval segments
...	additional plotting parameters

Details

...

Author(s)

Han-Ming Wu

See Also

plot.2d.i

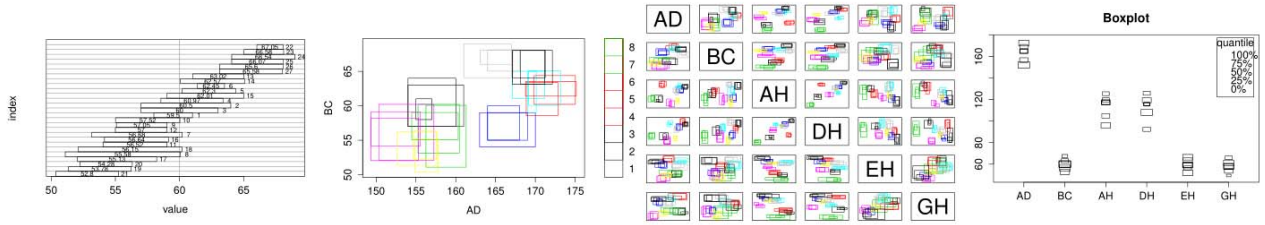
Examples

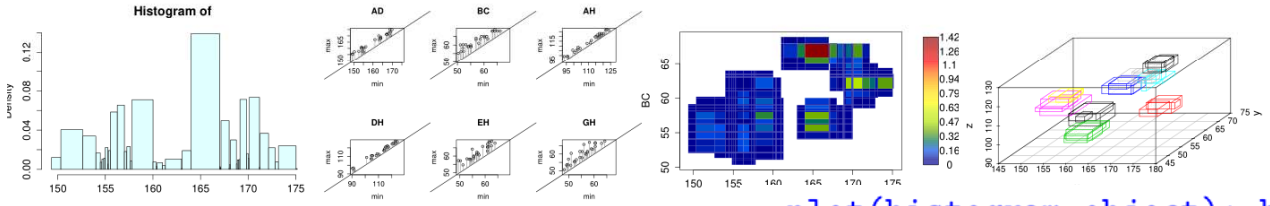
```
data(face)
idata.x <- face$x
y.C <- face$y
title <- "face data"
plot.index.i(idata.x)
plot.index.i(idata.x, vertical=F)
plot.index.i(idata.x, vertical=F, col=y.C)
plot.index.i(idata.x)
```

graphics.SDA

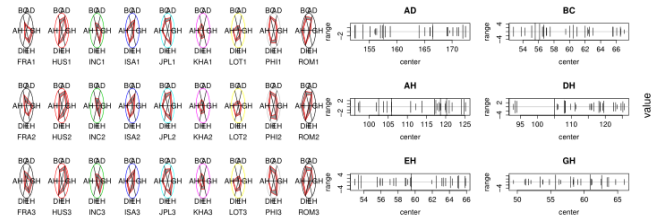
the R base graphics package companion for symbolic data analysis

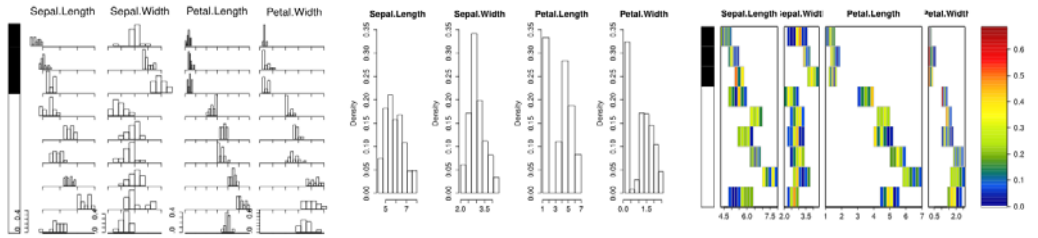
```
plot(interval_object); hist(interval_object); ...
```

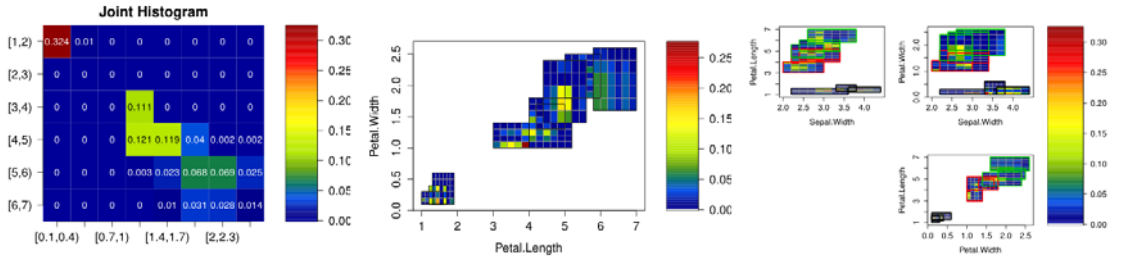





```
plot(histogram_object); hist(histogram_object); ...
```

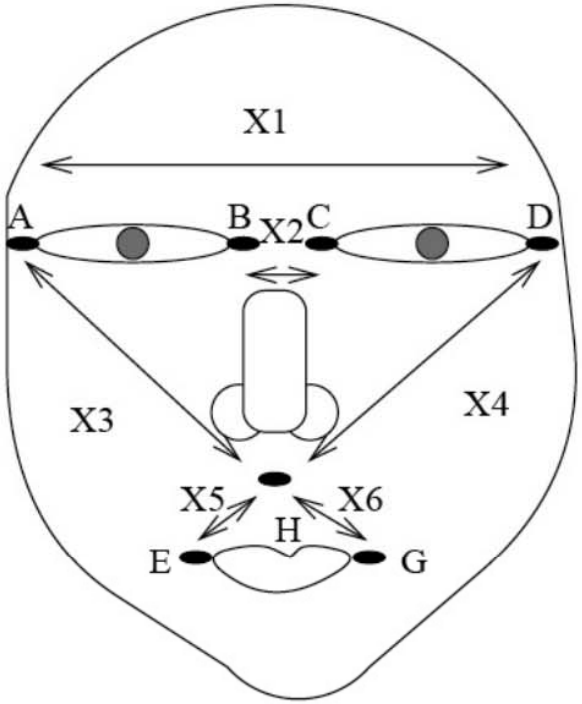






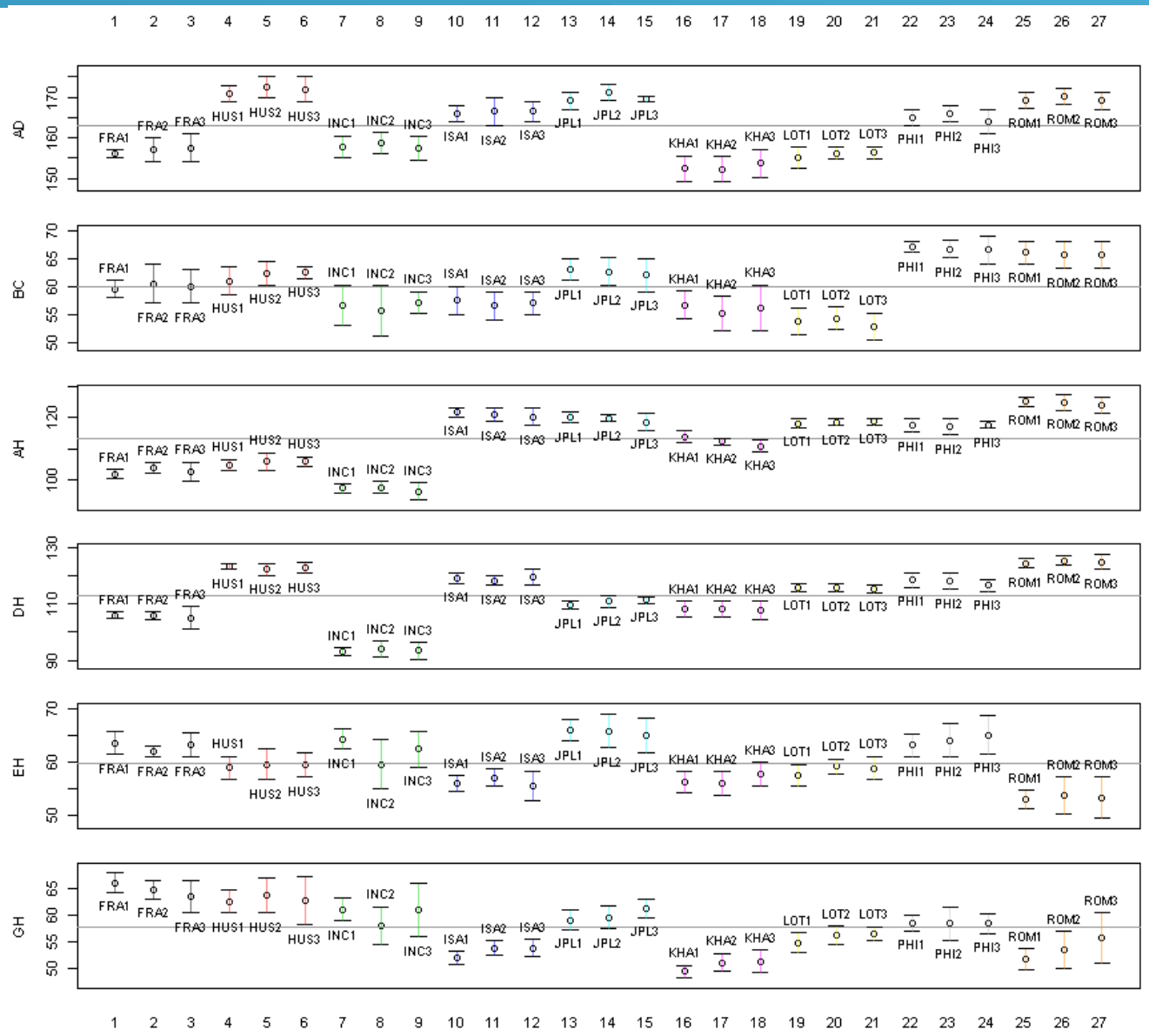
Face Recognition Data

1. Details: Leroy et al. (1996), Douzal-Chouakria, Billard and Diday (2011), Le-Rademacher and Billard (2012)
2. The dataset gives six face measurements of nine men, each with three observations, resulting in a total 27 observations. The measurements for each observation came from a sequence of images.

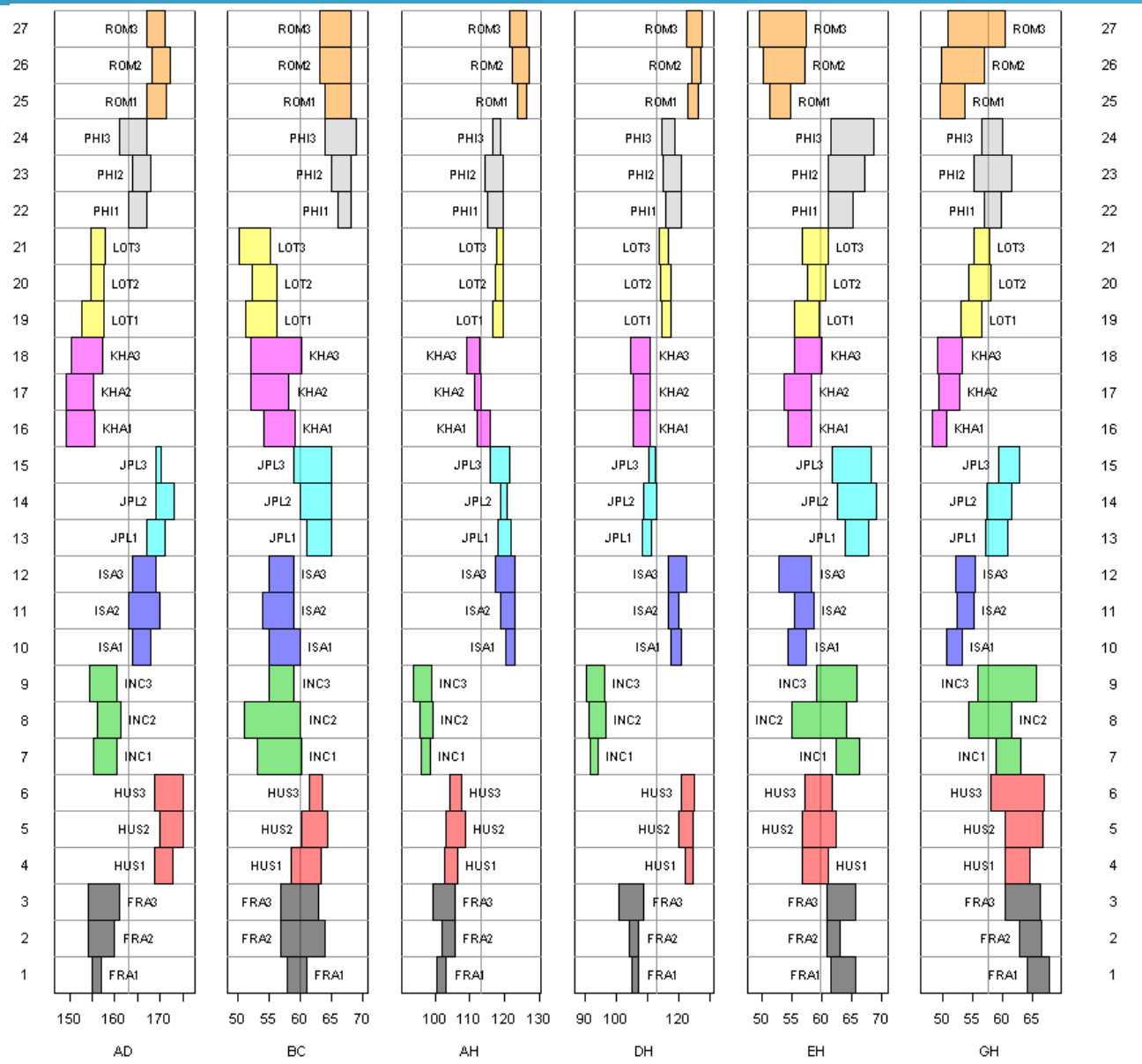


name	AD	BC	AH	DH	EH	GH
FRA1	(155, 157)	(58, 61.01)	(100.45, 103.28)	(105, 107.3)	(61.4, 65.73)	(64.2, 67.8)
FRA2	(154, 160.01)	(57, 64)	(101.98, 105.55)	(104.35, 107.3)	(60.88, 63.03)	(62.94, 66.47)
FRA3	(154.01, 161)	(57, 63)	(99.36, 105.65)	(101.04, 109.04)	(60.95, 65.6)	(60.42, 66.4)
HUS1	(168.86, 172.84)	(58.55, 63.39)	(102.83, 106.53)	(122.38, 124.52)	(56.73, 61.07)	(60.44, 64.54)
⋮			⋮			
ROM3	(167.11, 171.19)	(63.13, 68.03)	(121.62, 126.57)	(122.58, 127.78)	(49.41, 57.28)	(50.99, 60.46)

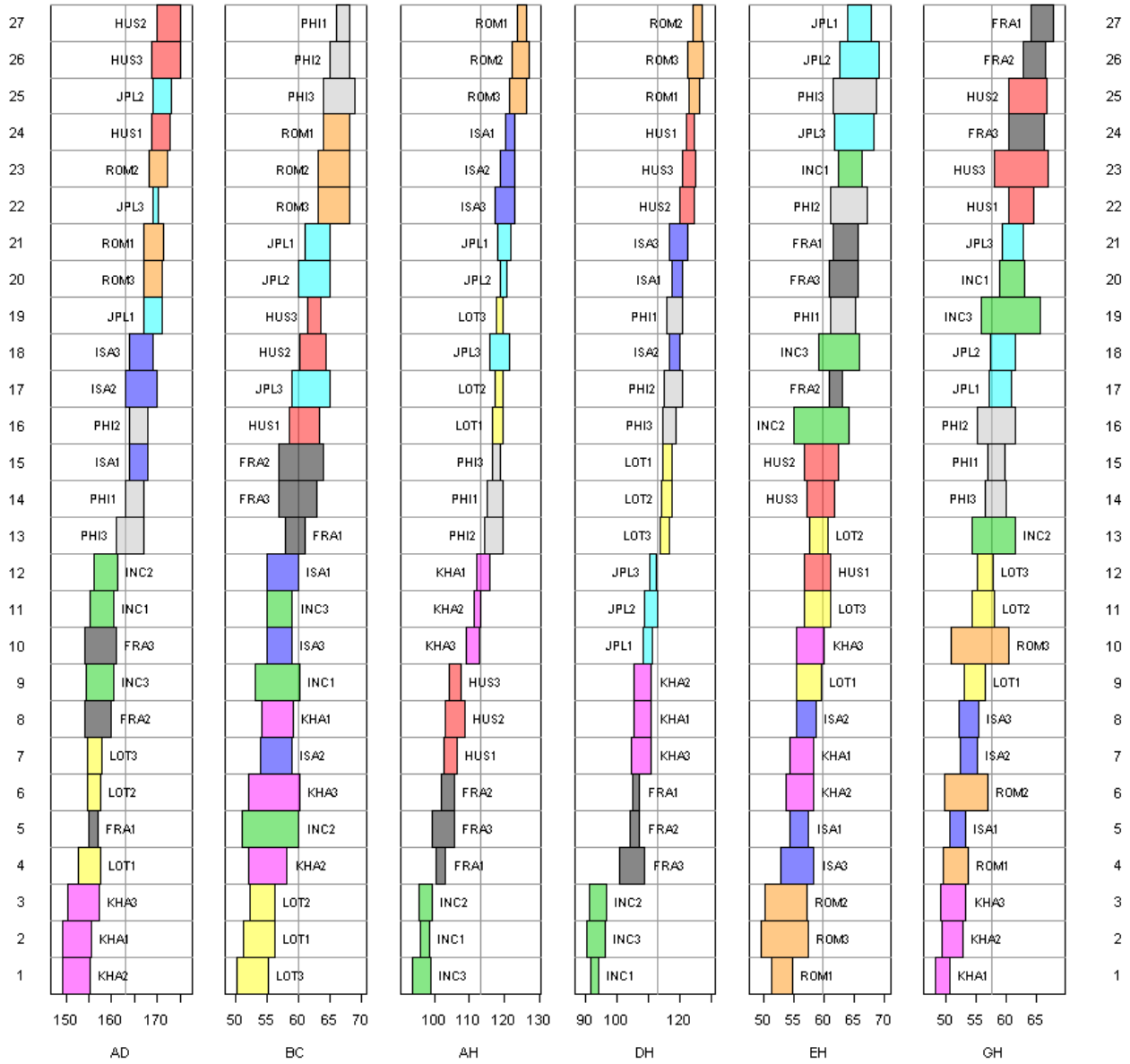
plot.index.i(idata.x, fill.col=y.C)



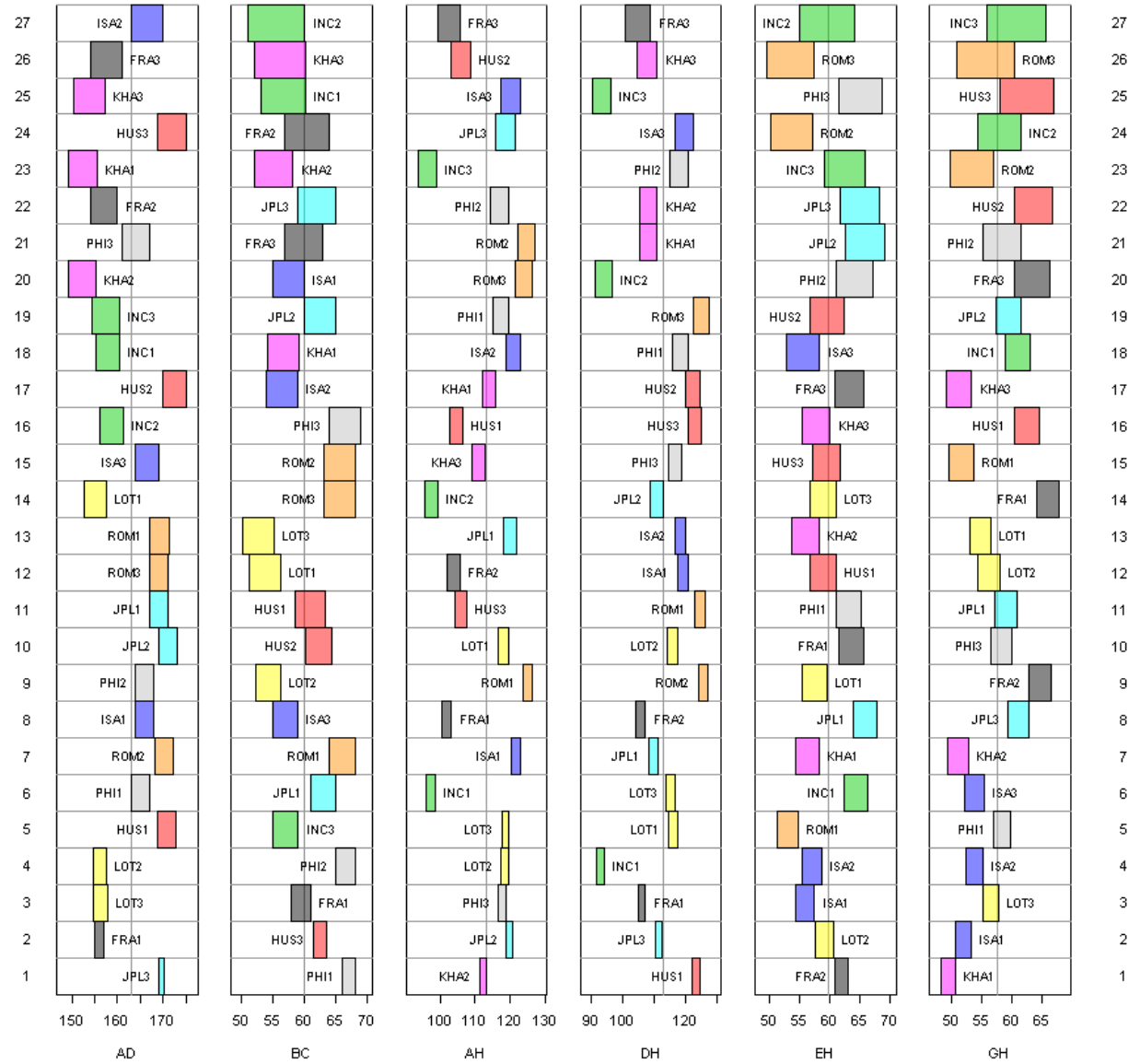
plot.index.i(idata.x, type="rect", vertical = T, fill.col=y.C)



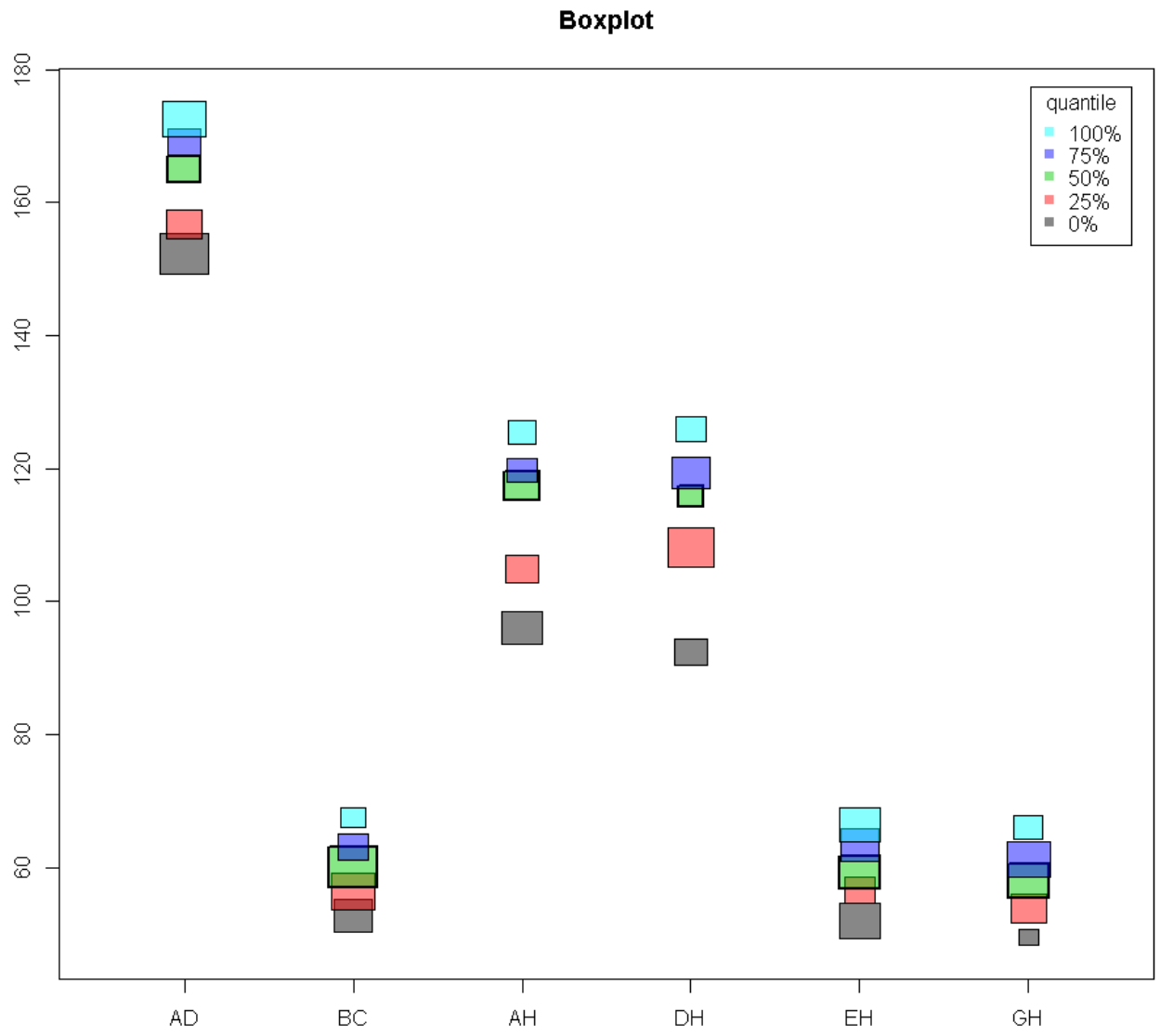
plot.index.i(idata.x, type="rect", vertical = T, align="c", fill.col=y.C)



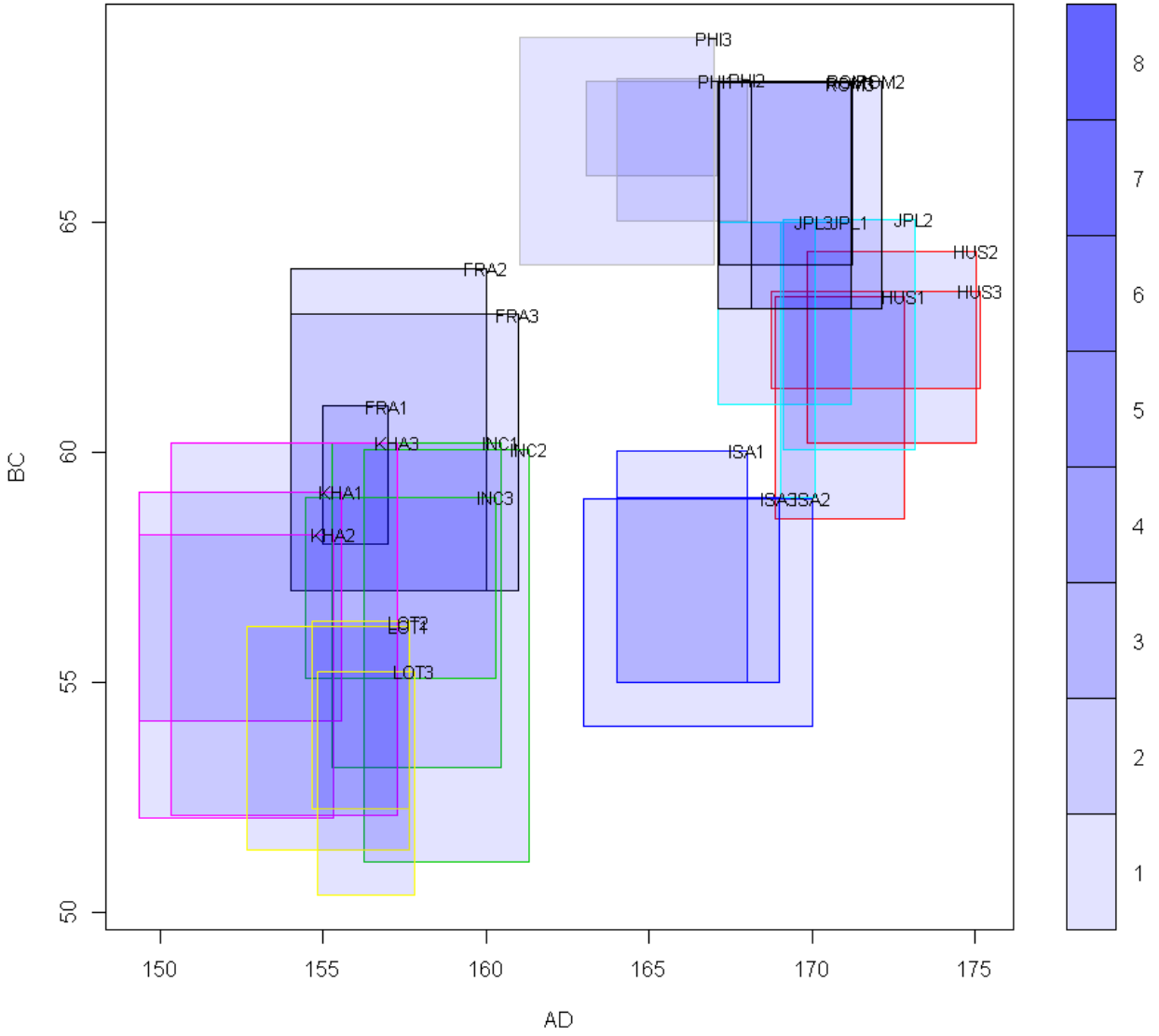
plot.index.i(idata.x, type="rect", vertical = T, align="range", fill.col=y.C)



boxplot.sbs.i(idata.x)

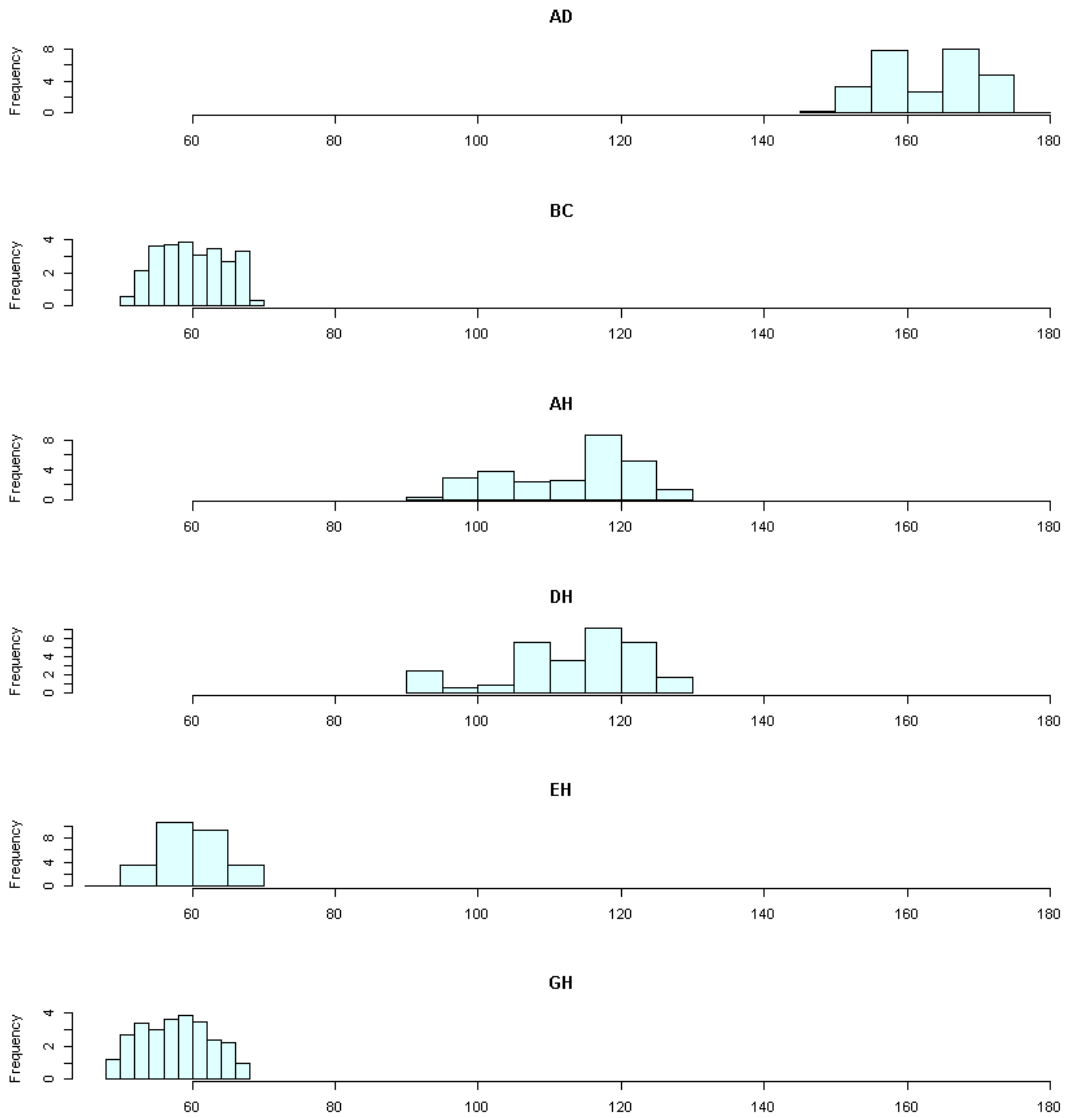


plot.2d.i(idata.x[,1:2,], border.col=y.C)

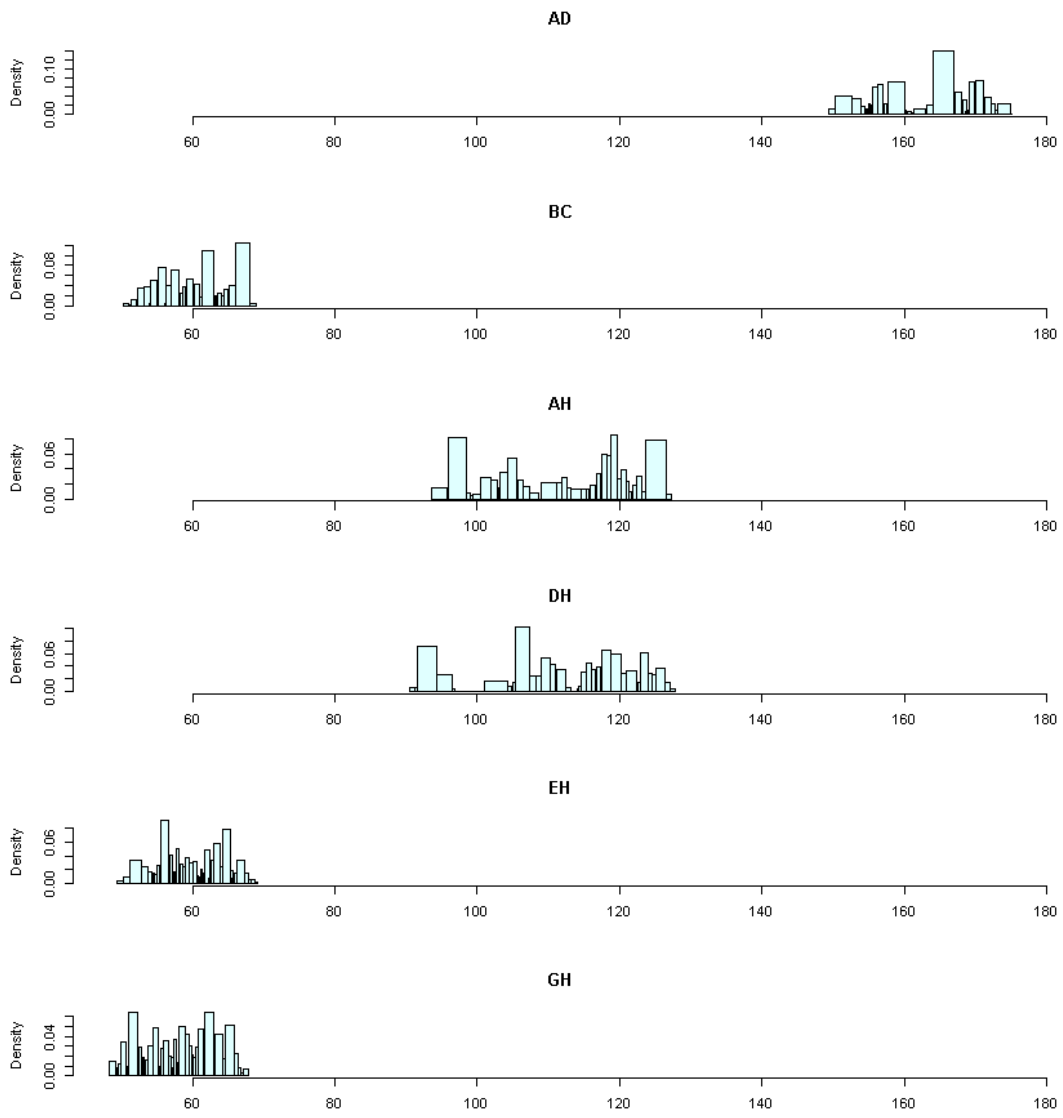


pairs.i(idata.x, col=NA, border.col=y.C)

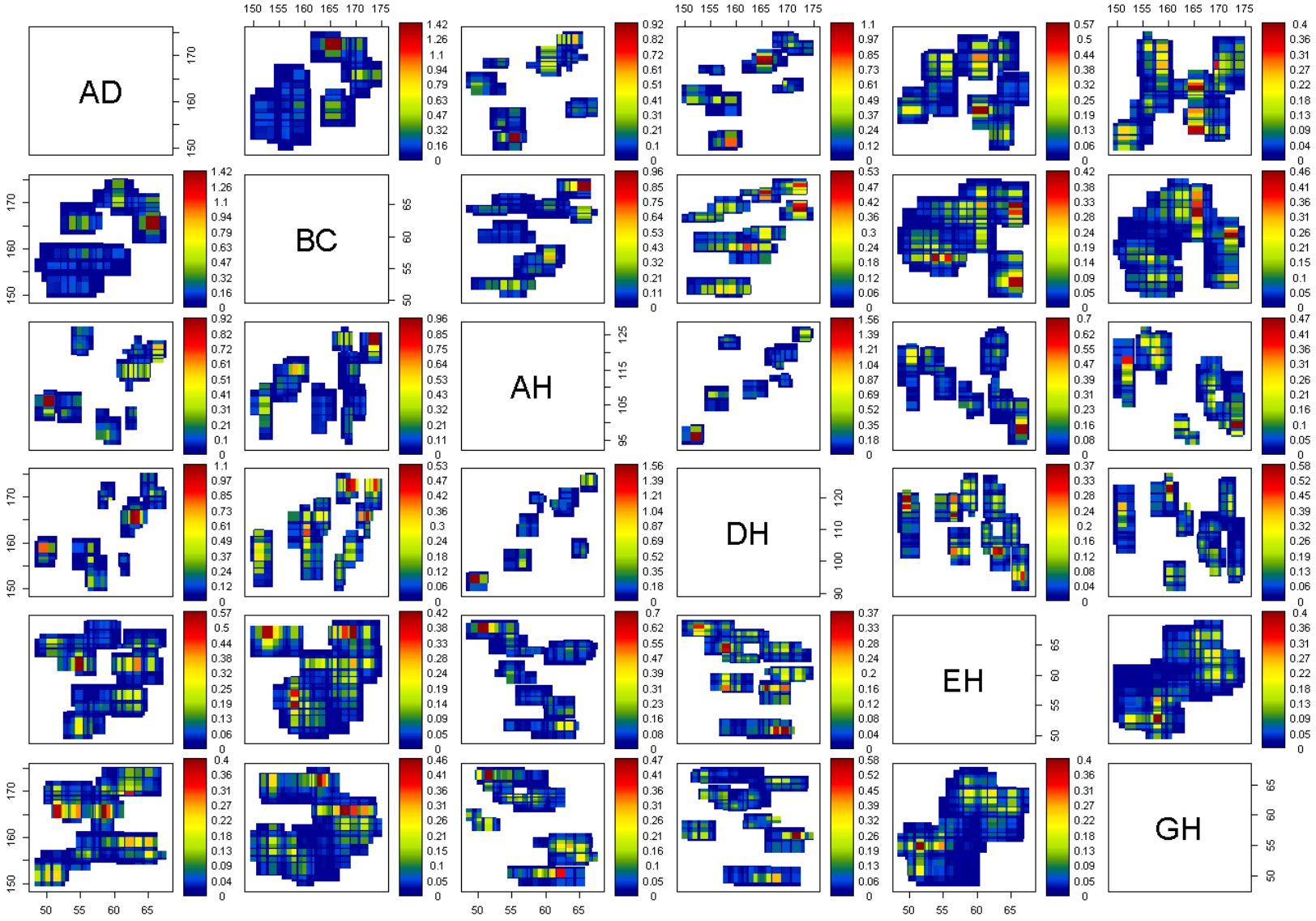
hist.i(idata.x)



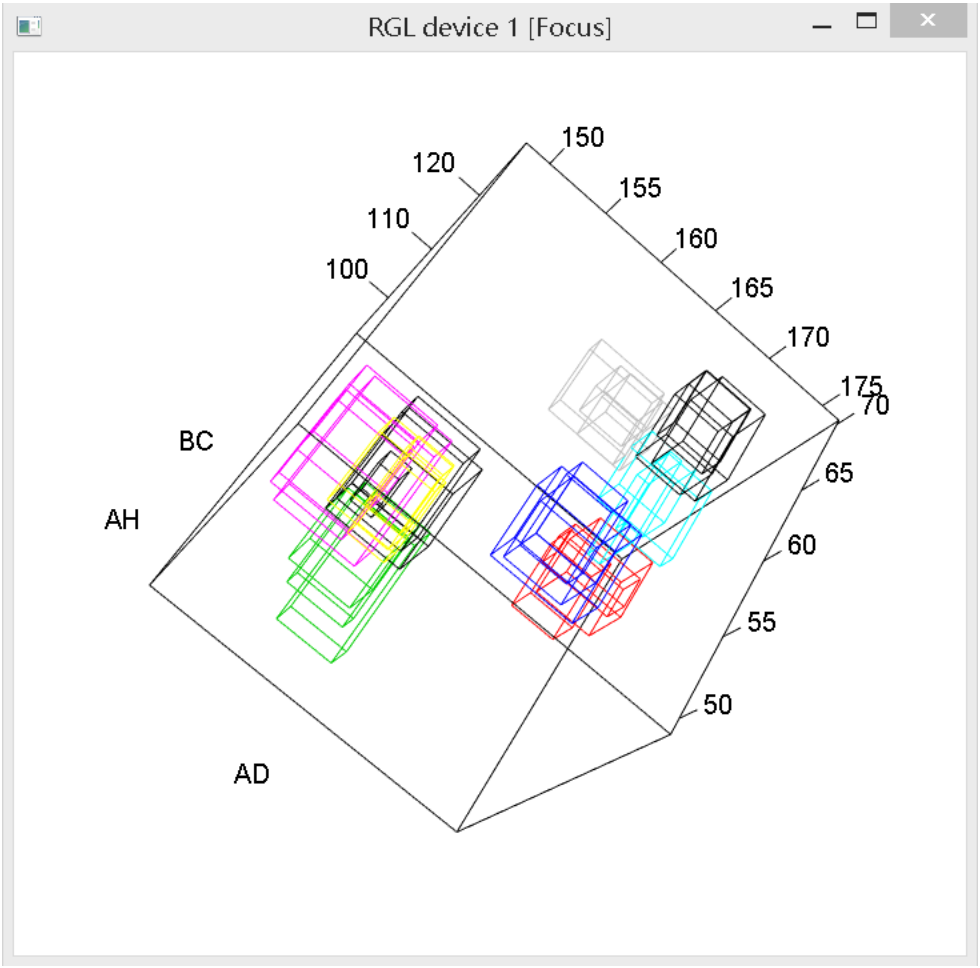
hist.overlap.i(idata.x)



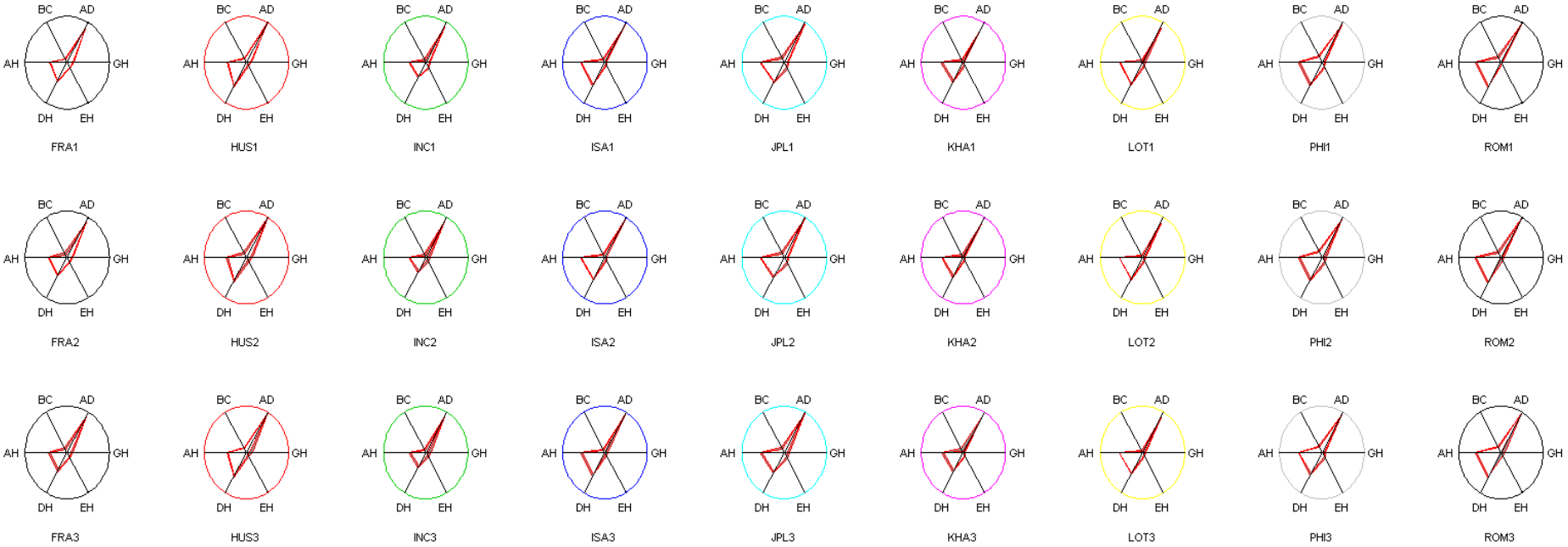
hist.overlap.2d.i(idata.x, col="rb130")



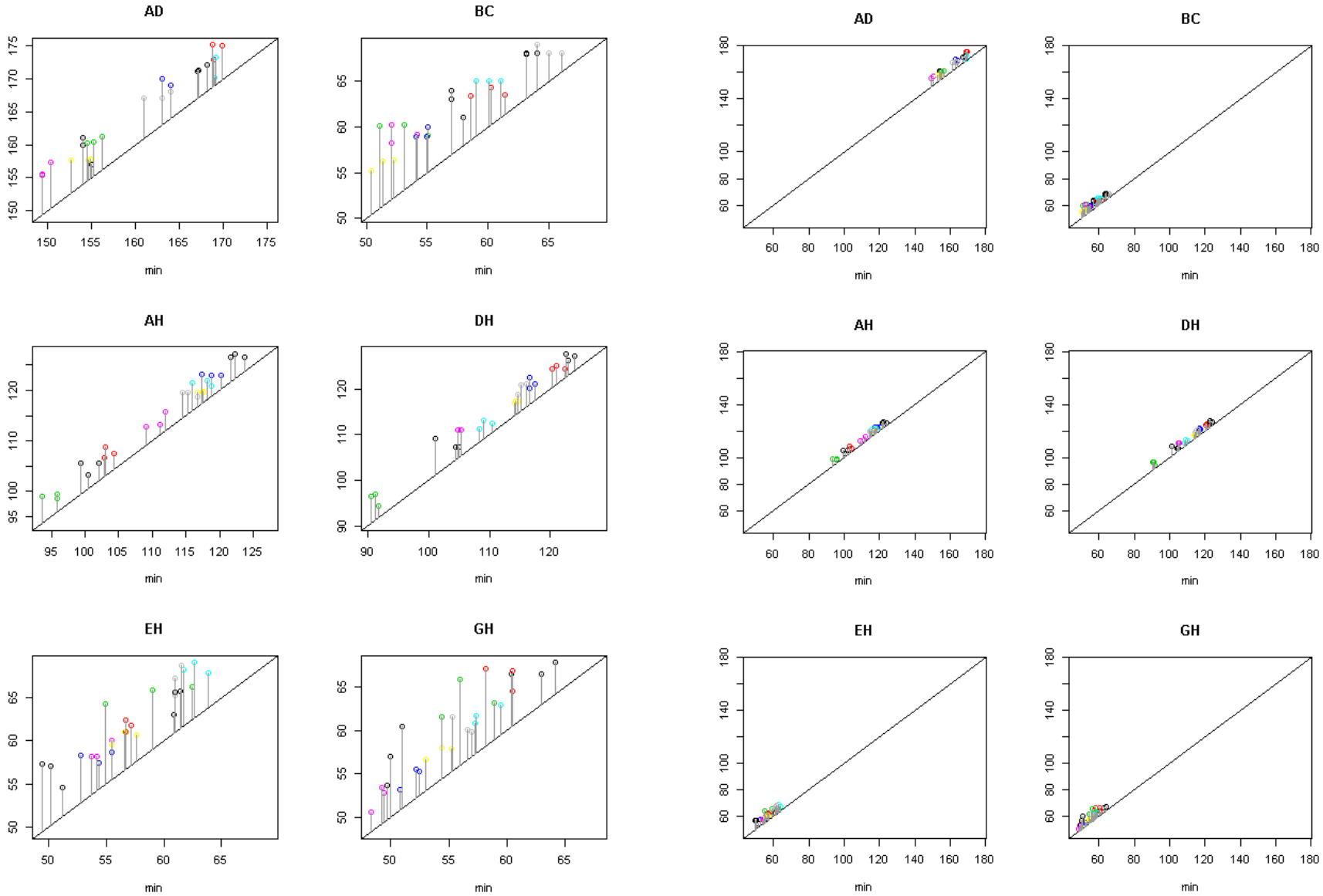
scatterplot3d.i(idata.x[,1:3,], col=y.C, rgl=T)



radar.i(idata.x, nrow=3, byrow=F, circle.col=y.c, scale.id=1)



MMplot(idata.x, plot.type=1)



RCplot(idata.x, plot.type=2, scale=T)

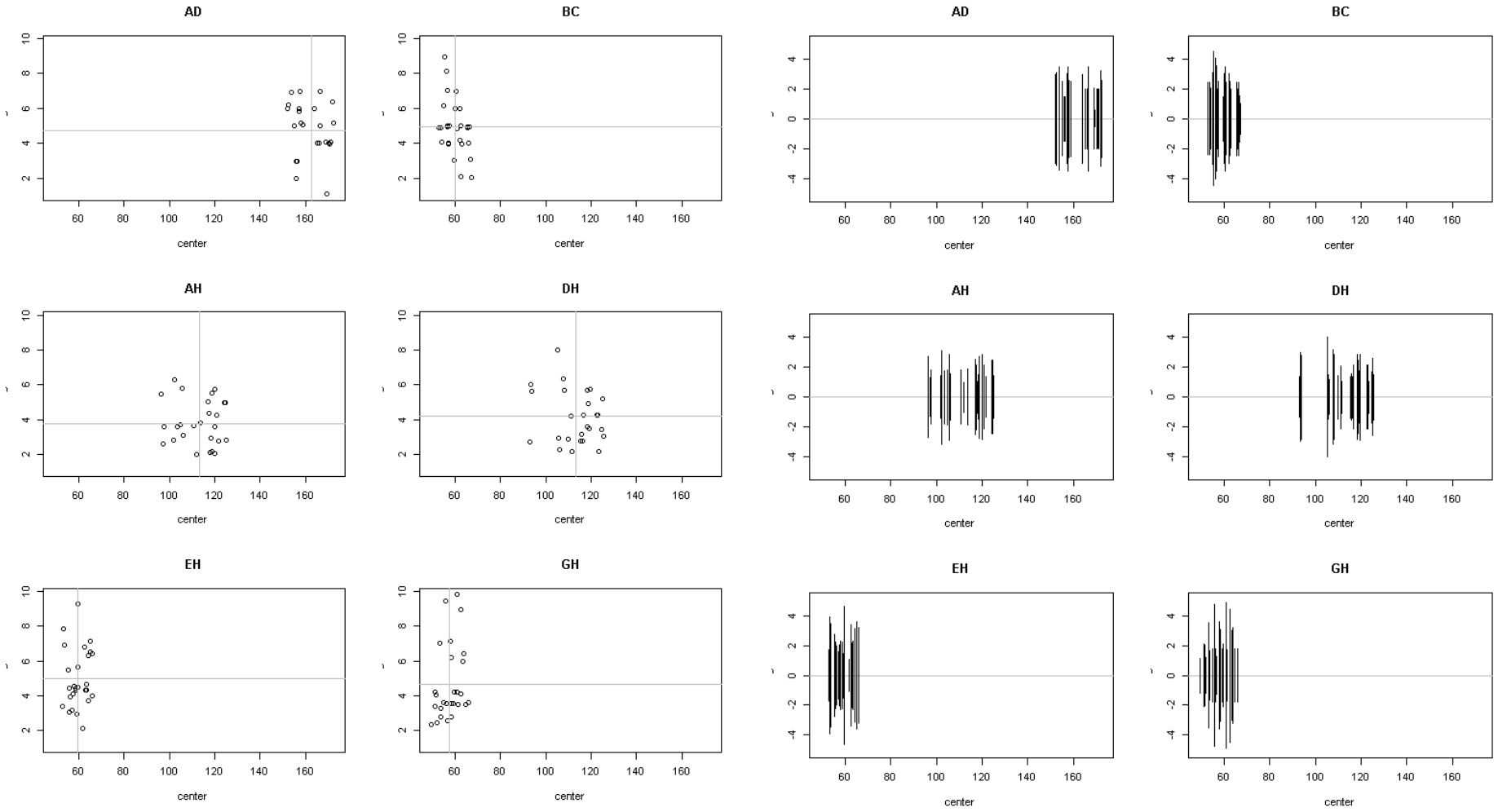
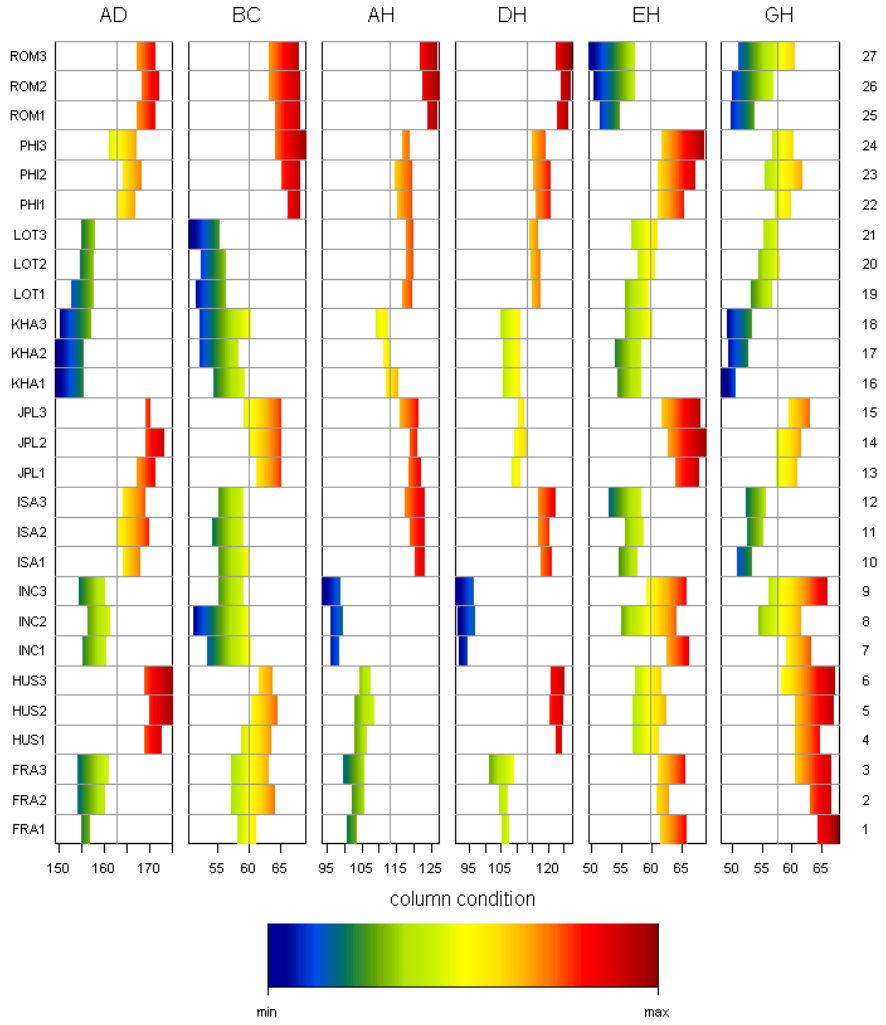


image.i(idata.x)



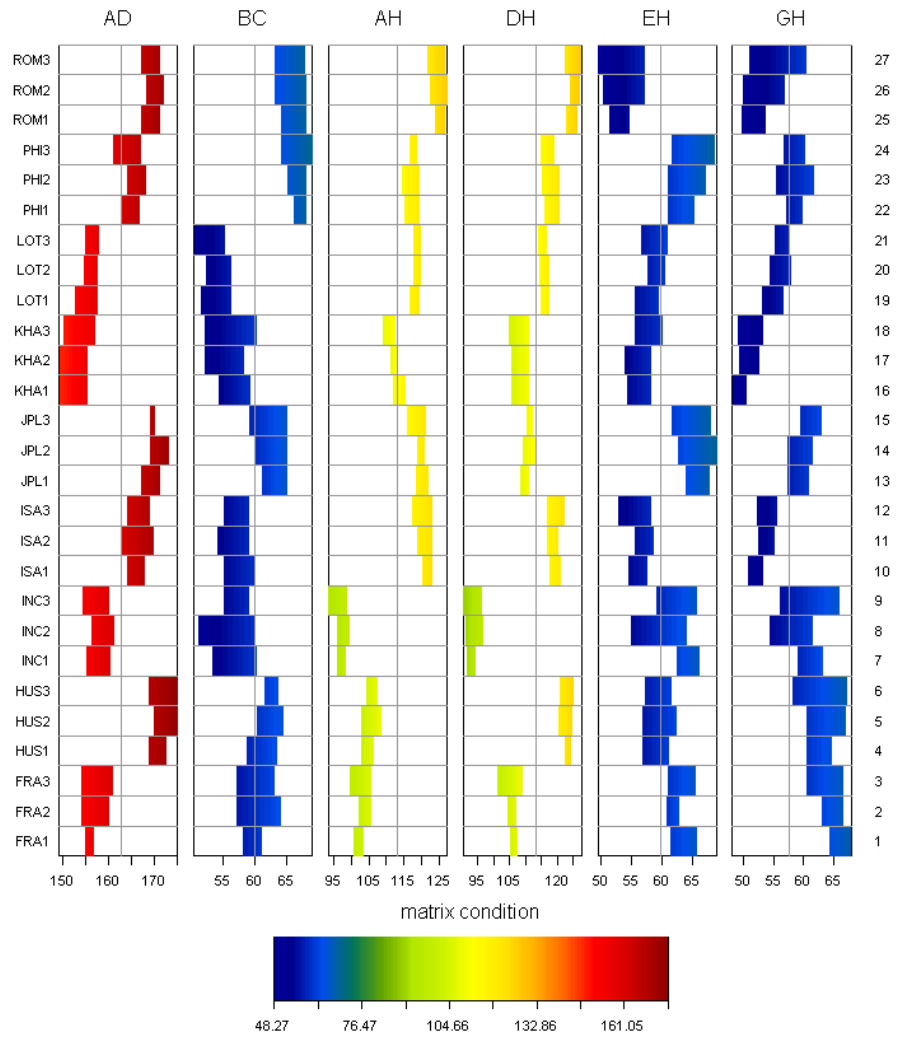
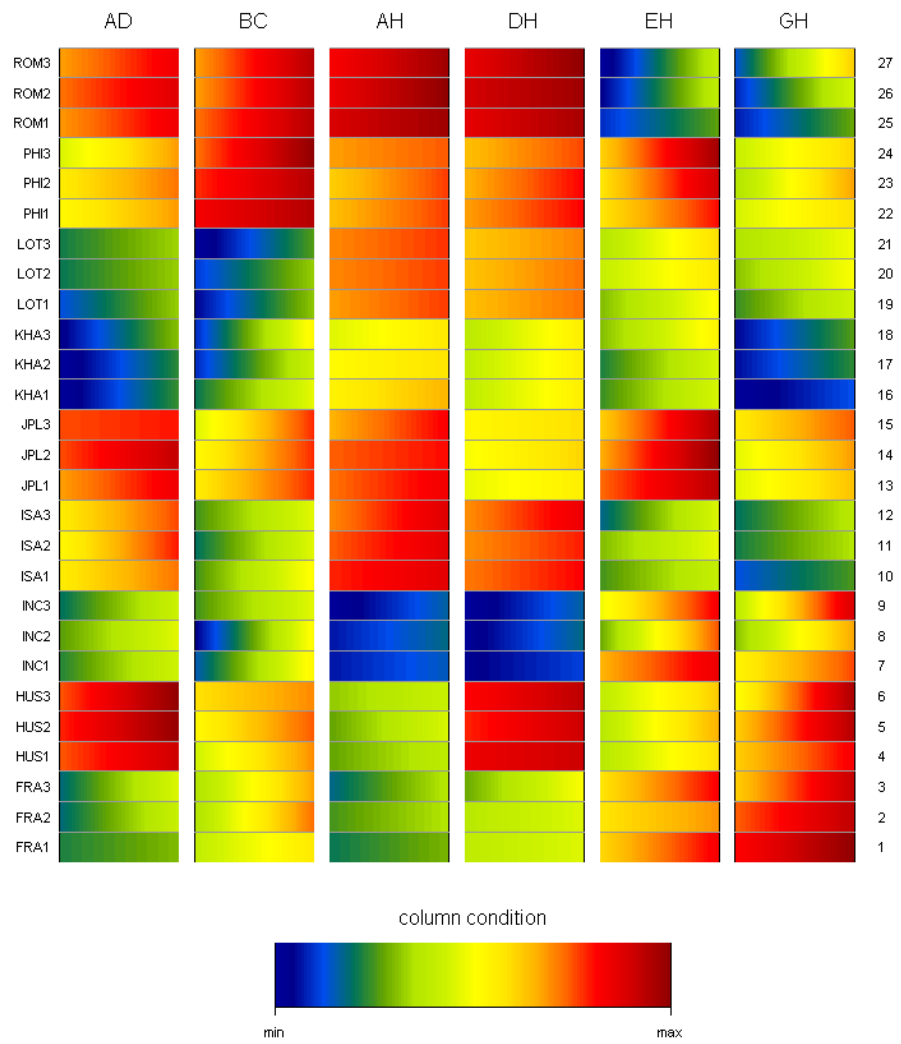
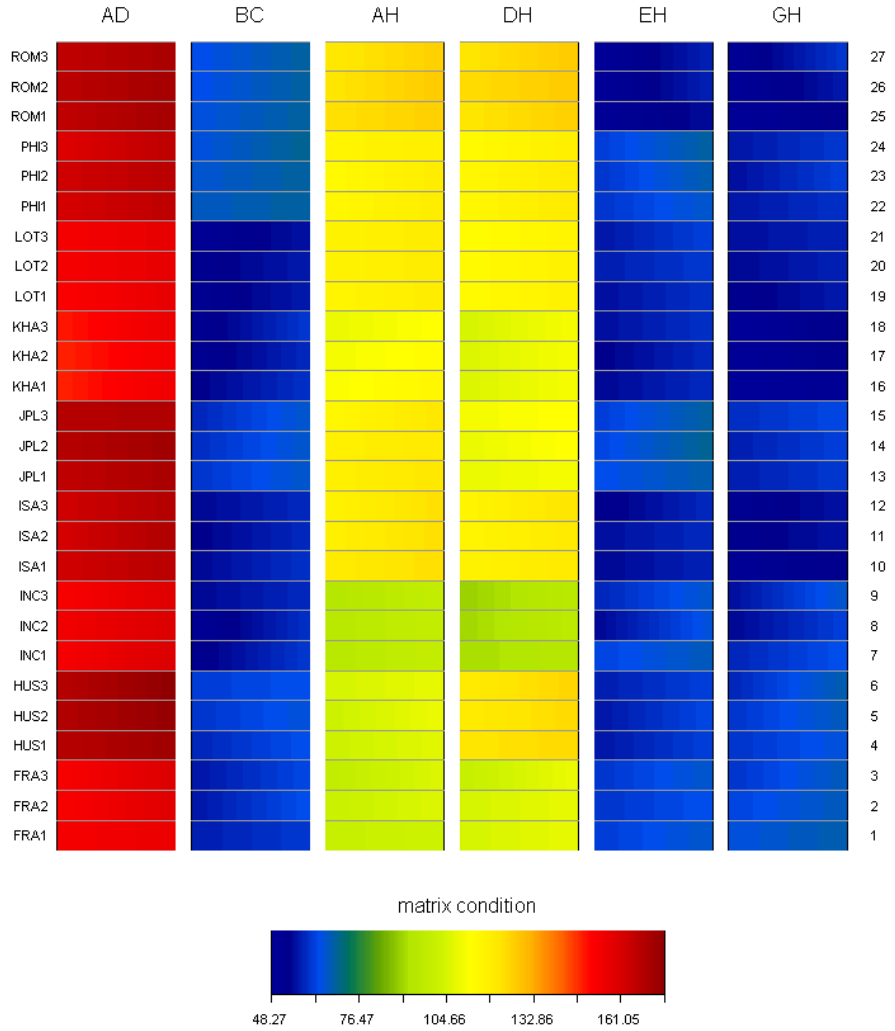
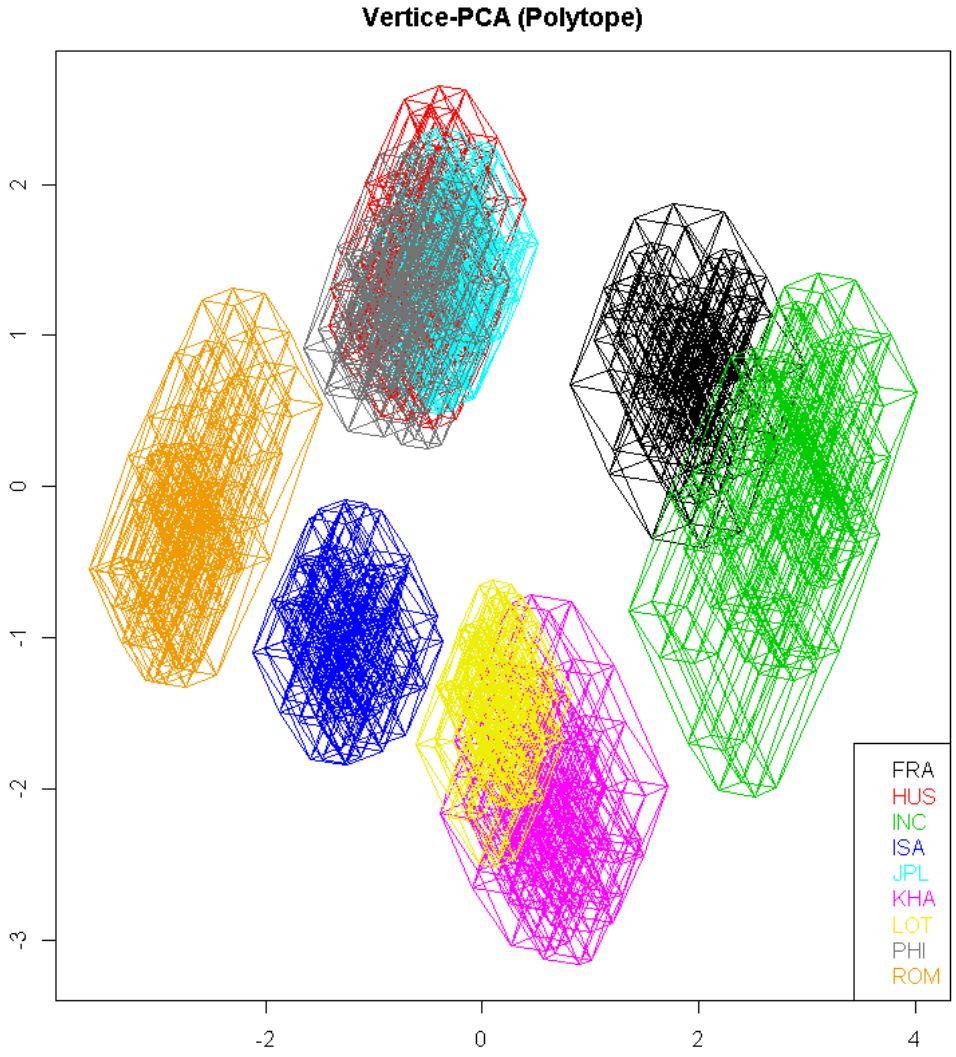


image.i(idata.x, type=2)





PCA for interval-valued Data

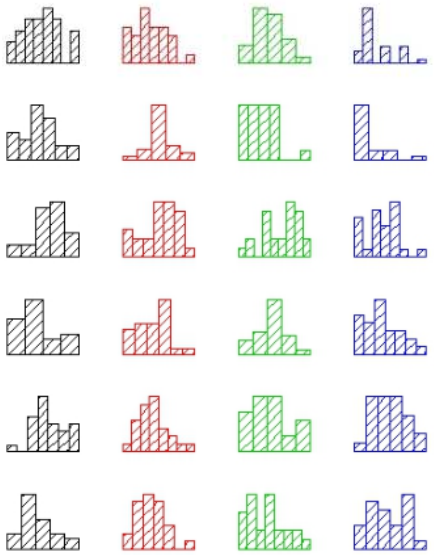


The Histogram-valued data

Let $O = \{o_1, o_2, \cdots, o_n\}$ be given objects.

1. The histogram-valued data: $H = (h_{ij})_{n \times p}$, where

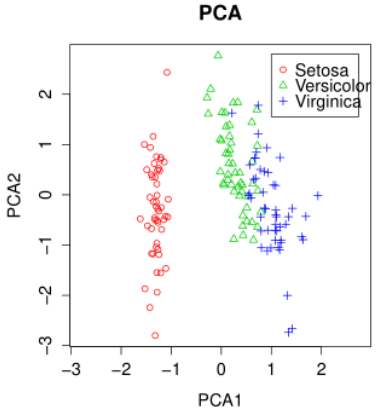
$$h_{ij} = \{I_{ij}, p_{ij}\} = \{x \in I_{ij}^{(b)} = (x_{ij}^{(b)}, x_{ij}^{(b+1)}], p_{ij}^{(b)}, b = 1, \cdots, B_{ij}\}.$$

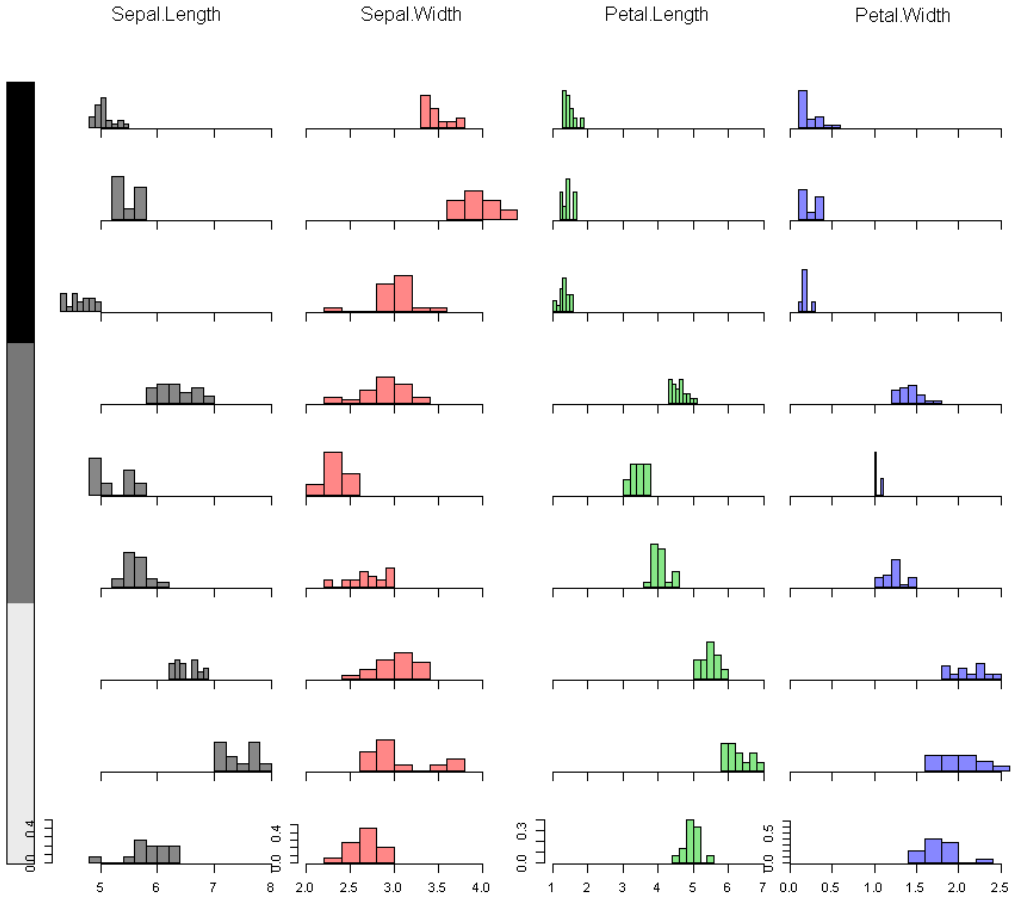


Possible source: the result of an aggregation, the description of a population, or any other grouped collective.

Iris data: generate histograms

The iris data (Fisher, 1936) consists of 50 samples from each of three species of Iris (Setosa, Virginica and Versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.





hist.h(hdata, scale.x = TRUE)

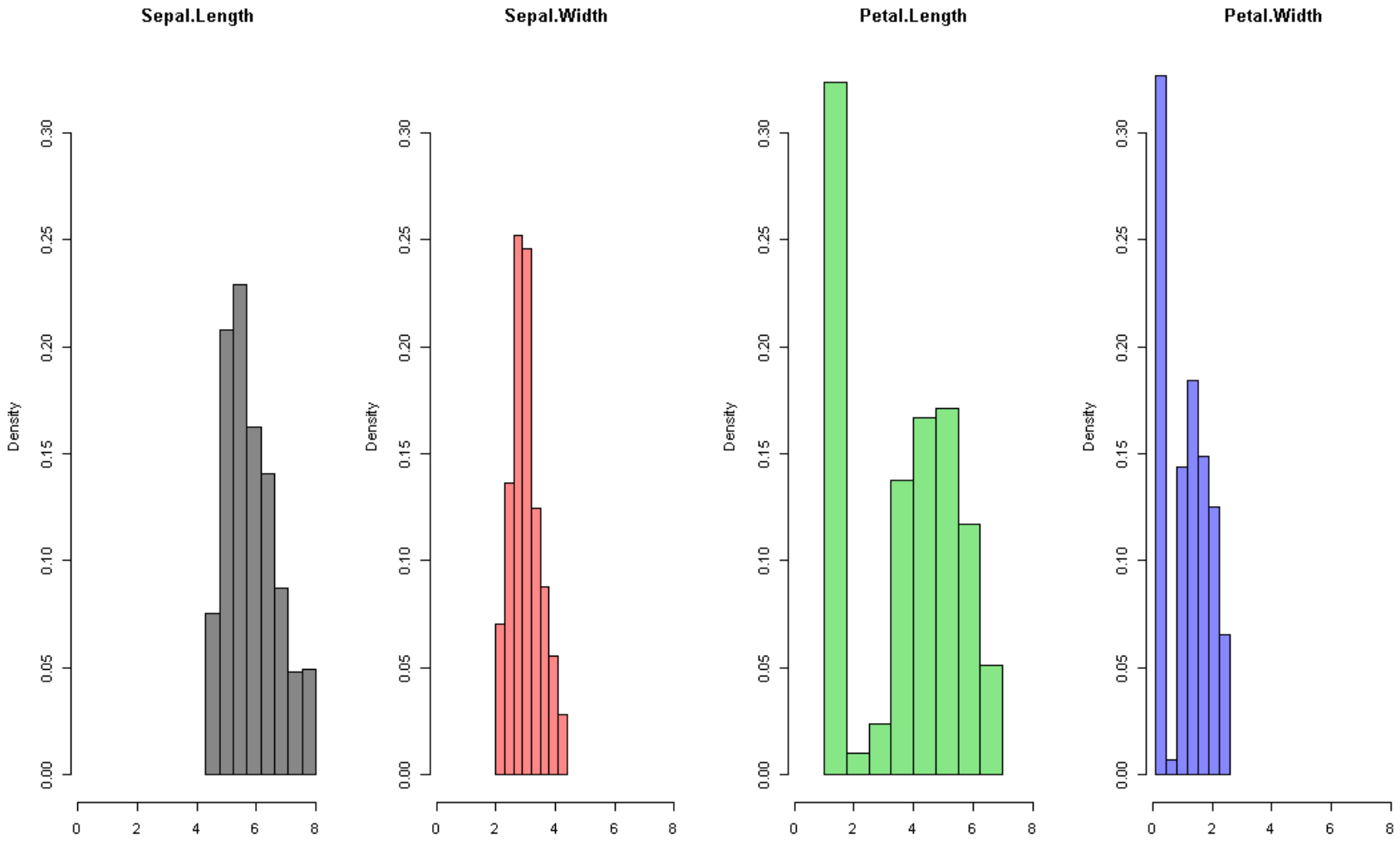
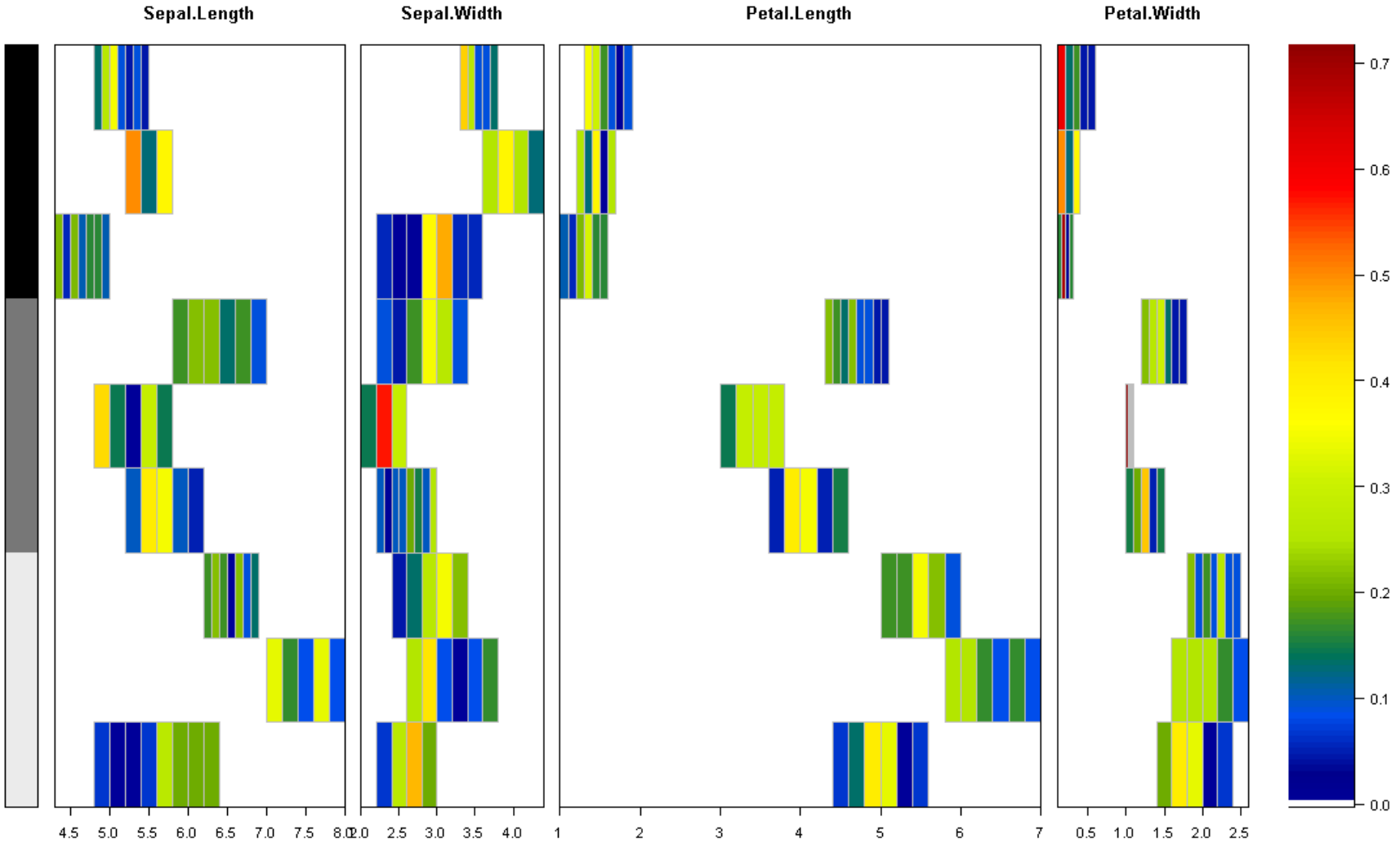
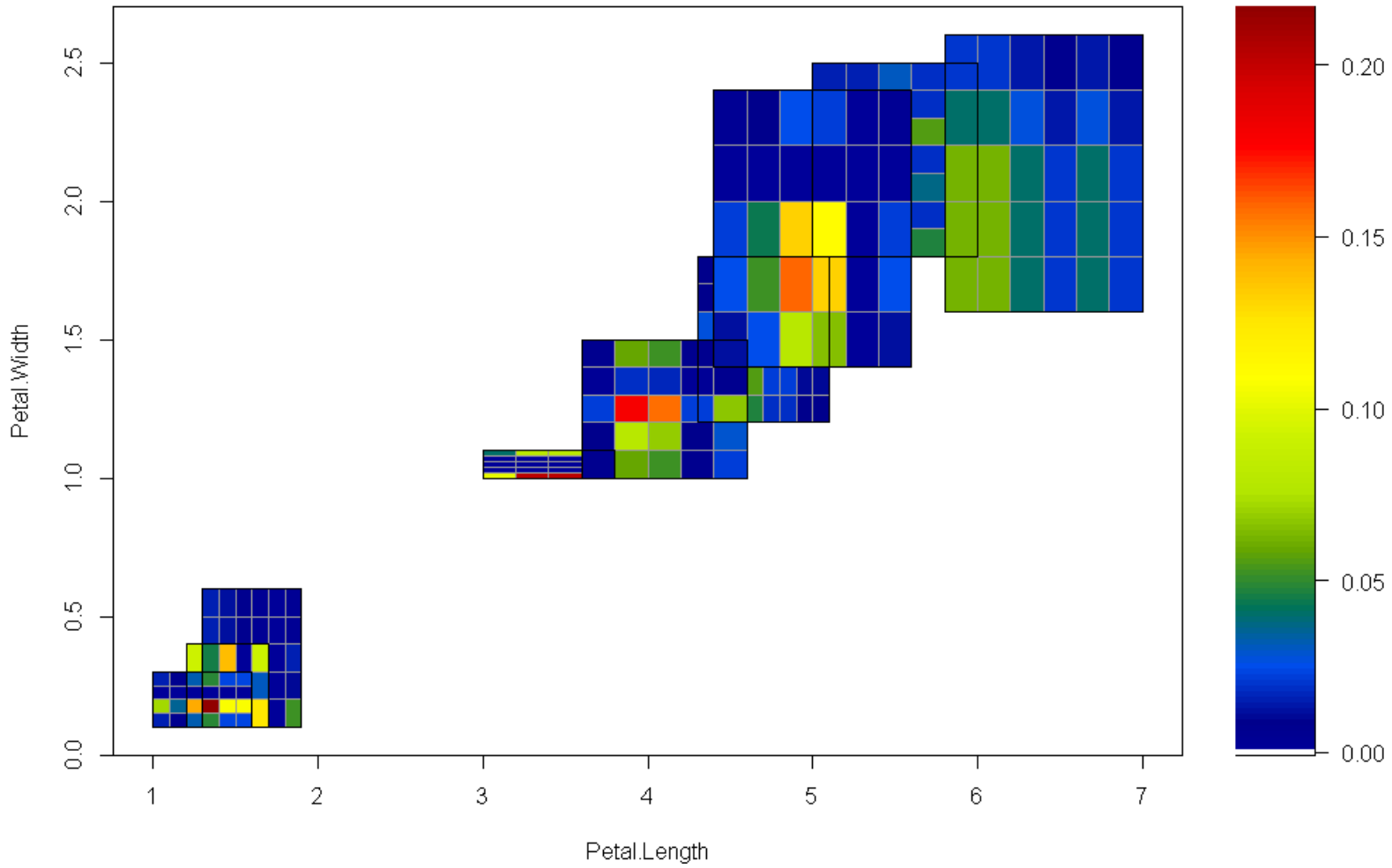


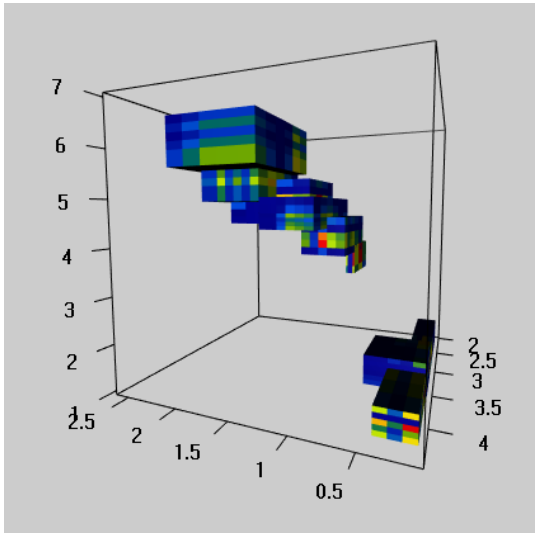
image.h(hdata)

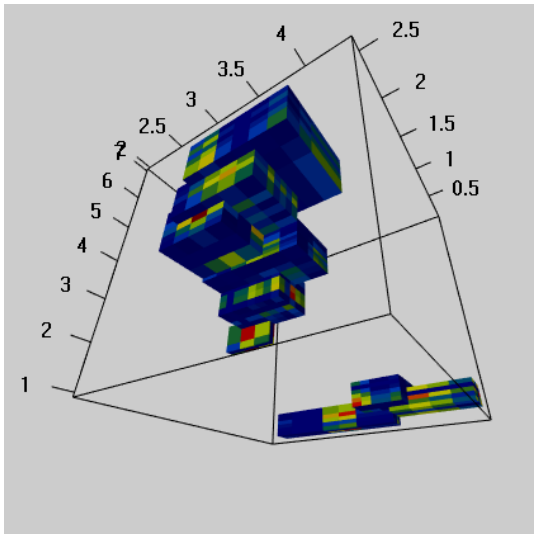


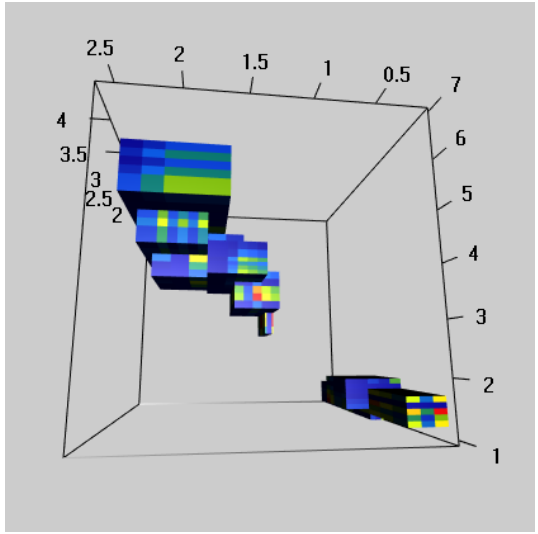
scatterplot.h(hdata.s1)



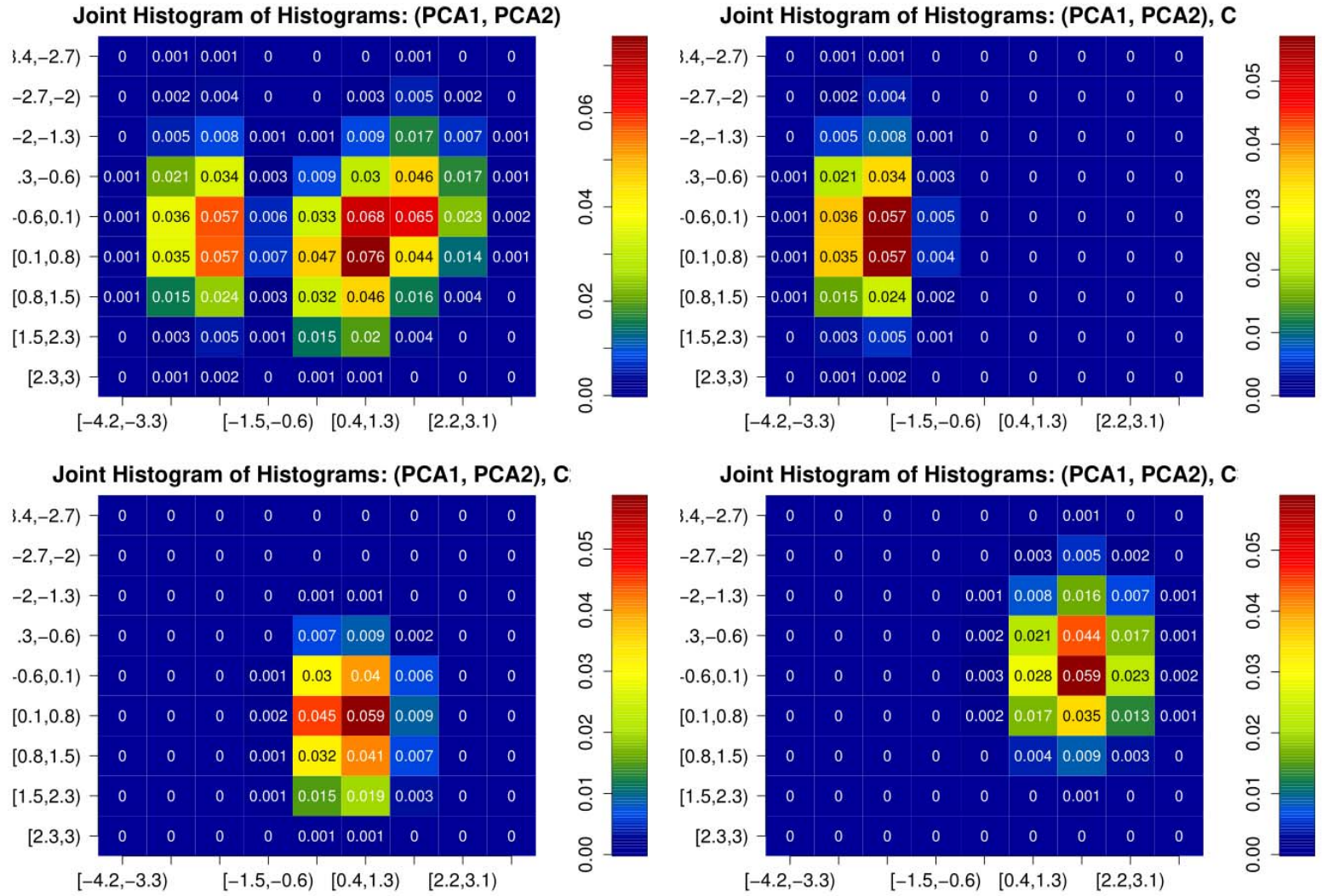
scatterplot3d.h(hdata.s2)







PCA for Histogram-valued Data



Acknowledgment

TKU, NTPU, MOST, Chun-houh Chen (中研院統計所 陳君厚)