REVOLUTION ANALYTICS

Intro to Plotting with Lattice Graphics

Revolution Analytics

Outline

Introduction

The `lattice` package is:

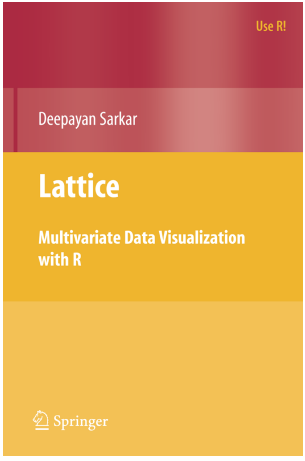
- an API built on Trellis graphics which were a part of S and S+
- Independent of R base graphics
  - Not a pen-and-paper model
  - Uses `grid` graphics under the hood

Benefits of using 'lattice:

- Excellent for visualizing multivariate data
- Simple to code
- Provides functions for many common statistical graphics
- Provides plots to visualise conditioning on a variable

For more information, see the lattice website.

Introduction



Univariate lattice functions

Univariate functions provide easy plotting of a *single* variable

Plot function	Description
`barchart`	Bar plot
`bwplot`	Comparative box-and-whisker plots
`densityplot`	Kernel density plot
`dotplot`	Cleveland dot plot
`histogram`	Histogram
`qqmath`	Theoretical quantile plot

Bivariate lattice functions

Bivariate functions provide easy plotting of the relationship among *two* variables (or more, with advanced features)

Plot function	Description
`xyplot`	Scatter plot

Multivariate lattice functions

Trivariate and hypervariate functions provide easy plotting of the interaction of *more than two* variables

Plot function	Description
`splom`	Scatter-plot matrix
`cloud`	Three-dimensional scatter plot
`wireframe`	Three-dimensional surface plots

lattice formulae

The lattice functions make use of formulae to specify statistical models

```
help(formula)
```

Formula Examples

Formula	Meaning
`y ~ x`	Plot y as a function of `x`
`y ~ x | z`	denotes conditioning
`y ~ x | z + s + t`	`z`, `s`, and `t` are conditioning variables
`y ~ x | z * s * t`	Conditioning variables may be seperated by + or *
`log(y) ~ x * z | s`	Transforms involving more than one column
`log(y) ~ x | factor(s)`	Transforms can inlude other functions
`~ x`	Valid formula

Outline

We need data

Let's load some data from the airlines visualization contest.

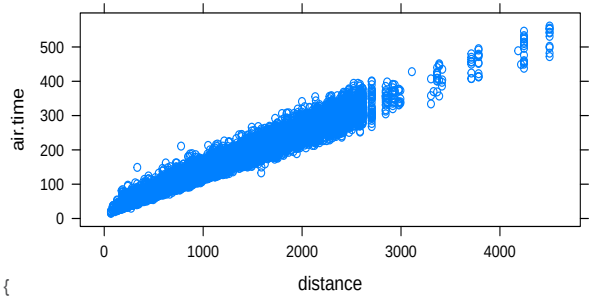
```
dataPath <- "../data"
load(file.path(dataPath, "airline.RData"))
str(airline)
```

```
## 'data.frame':    31617 obs. of  19 variables:
##  $ carrier         : Factor w/ 5 levels "American","Delta",..: 3 2 3 3 1 4 3 1 4 3 ...
##  $ year              : num  2008 2008 2008 2008 2008 ...
##  $ month          : Factor w/ 12 levels "January","February",..: 6 1 2 6 2 10 11 6 8 4 ...
##  $ day               : num  16 18 17 2 3 20 3 9 8 4 ...
##  $ weekday         : Factor w/ 7 levels "Monday","Tuesday",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ departure.time    : num  802 1328 1602 1643 959 ...
...
```

Creating an xyplot

Create a simple xyplot.

```
xyplot(air.time ~ distance, data = airline)
```



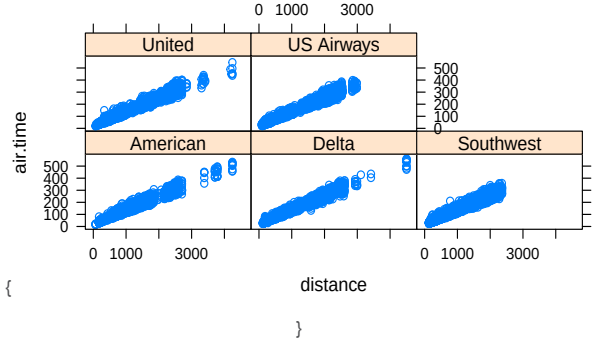
{

}

Add a Conditioning Variable

Easily add multi-panel conditioning

```
xyplot(air.time ~ distance | carrier, data = airline)
```

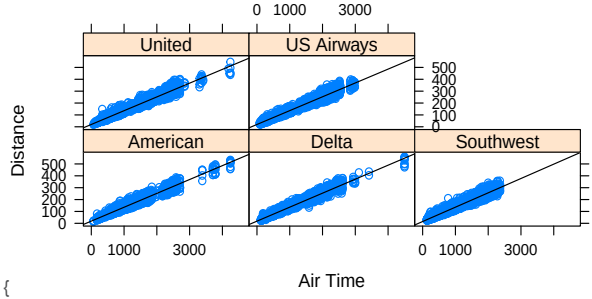


Combining Multiple Elements

Unlike base graphics, `lattice` takes all extra graphical elements in the initial plot function in the form of a “panel” function.

```
# Define a panel function to plot xyplot and lmline
panelFun <- function(x, y) {
  panel.xyplot(x, y)
  panel.lmline(x, y)
}
xyplot(air.time ~ distance | carrier, data = airline, panel = panelFun,
  xlab = "Air Time", ylab = "Distance")
```

Combining Multiple Elements



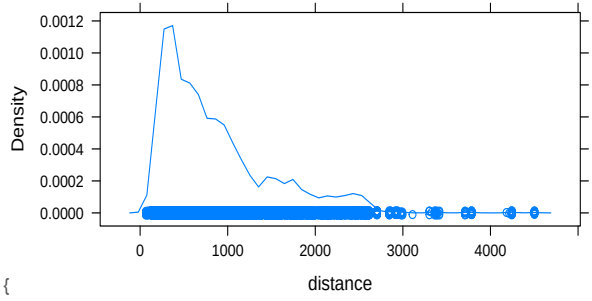
Incorrect formula specification

```
densityplot(distance, data = airline)  # doesn't work
densityplot(airline$distance, data = airline)  # throws an error
```

Why not?

Correct formula specification

```
# Have to use formula for single variable
densityplot(~distance, data = airline)  # proper syntax
```



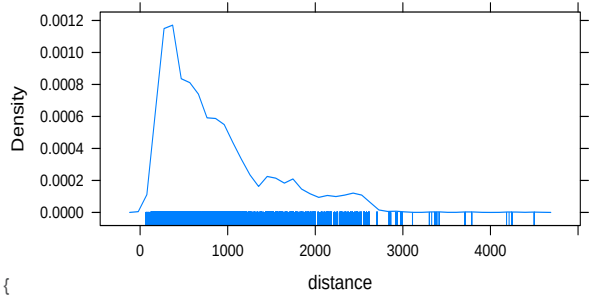
{

}

Density plot with rug in margin

Creating a density plot with a rug and rectangle kernel function

```
densityplot(~distance, data = airline, plot.point = "rug")
```

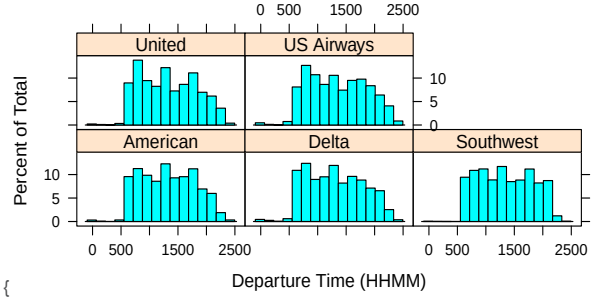


{

}

Histogram by carrier

```
# Histogram of departure time for each carrier
histogram(~departure.time | carrier, data = airline, xlab = "Departure Time (HHMM)")
```



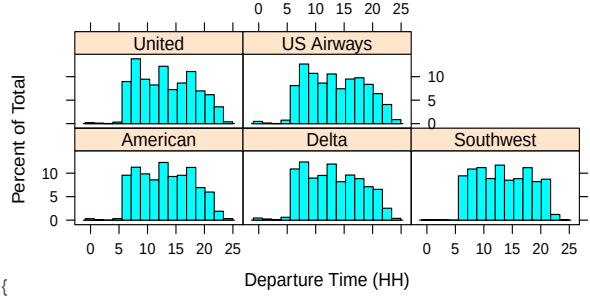
{

}

On-the-Fly Transformations

Histogram of departure time for each carrier with dynamic transformation

```
histogram(~departure.time/100 | carrier, data = airline, xlab = "Departure Time (HH)")
```

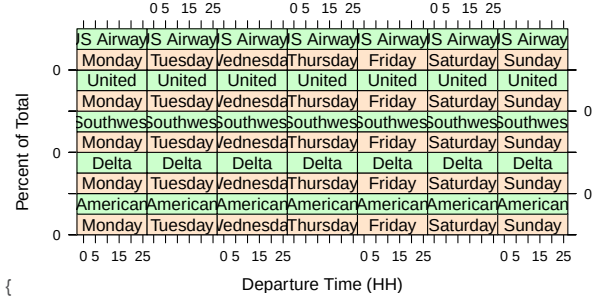


{

}

Complex Paneling

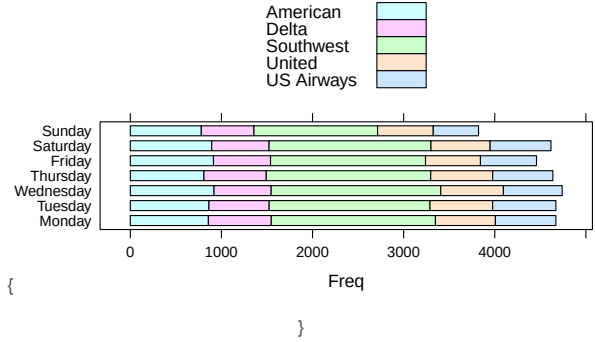
```
# Histogram of departure time for each carrier and day of the week
histogram(~departure.time/100 | weekday + carrier, data = airline,
  xlab = "Departure Time (HH)")
```



Outline

Simple barchart

```
# Barchart using tabulated values we created previously
table <- table(airline$weekday, airline$carrier)
barchart(table, auto.key = TRUE)
```

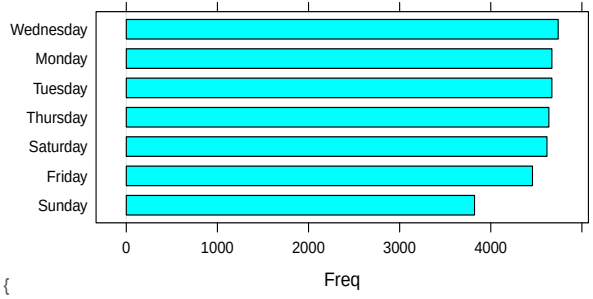



```
{

}
```

Simple sorted barchart

```
airline$weekday = factor(airline$weekday, levels(airline$weekday)[c(4,
  2, 6, 7, 5, 1, 3)])
table <- table(airline$weekday)
barchart(sort(table), auto.key = TRUE)
```

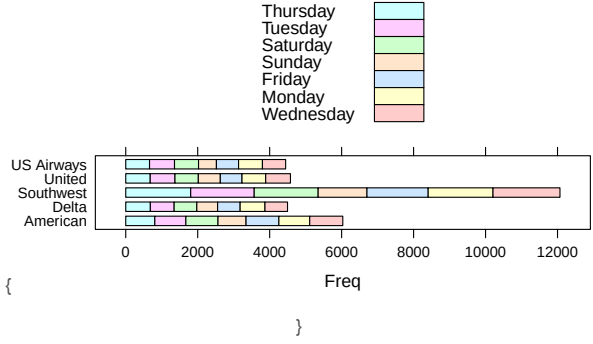



```
{

}
```

Bar chart

```
# Barchart using new tablulated values
barchart(table(airline$carrier, airline$weekday), auto.key = TRUE)
```



Exercise

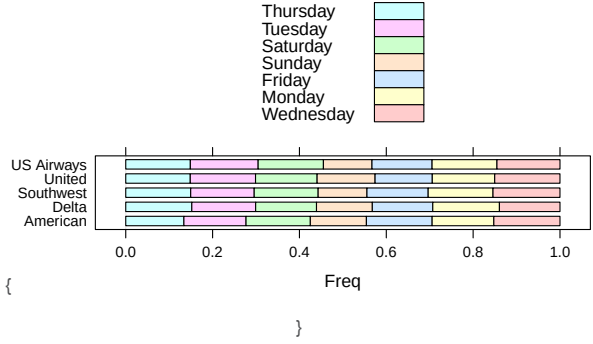
Your turn:

- Recreate the previous plot with Airways ordered by ascending total Freq.

Hint: use functions `order()` and `rowSums()`.

Barchart with proportions

```
# Barchart using new tablulated values and showing proportions
tabdata <- with(airline, prop.table(table(carrier, weekday), margin = 1))
barchart(tabdata, auto.key = TRUE)
```




```
{

}
```

Bar chart with binned groups

Back to some more data shaping. Our goal is to shape our data to visualize the amount of time each airline carrier spends in the air.

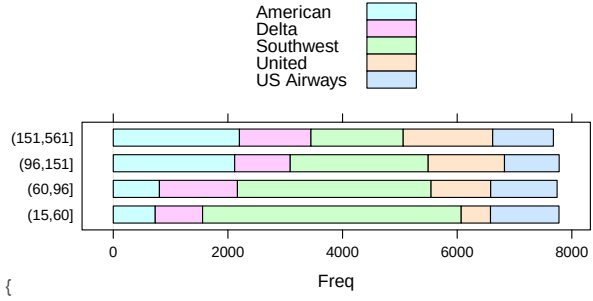
```
# Create a new cross table with binned air time using quantile(),
# cut() and table()
cbreaks <- quantile(airline$air.time, na.rm = TRUE)
airline$time.cut <- cut(airline$air.time, breaks = cbreaks)
(tabdata <- with(airline, table(time.cut, carrier)))
```

```
           carrier
time.cut    American Delta Southwest United US Airways
  (15,60]        732   826      4512    511       1195
  (60,96]        805  1361      3378   1040       1159
  (96,151]      2118   969      2407   1331        954
  (151,561]     2201  1248      1609   1561       1059
```

Bar chart with binned groups

Plotting barchart using the table we just created

```
barchart(tabdata, auto.key = TRUE)
```

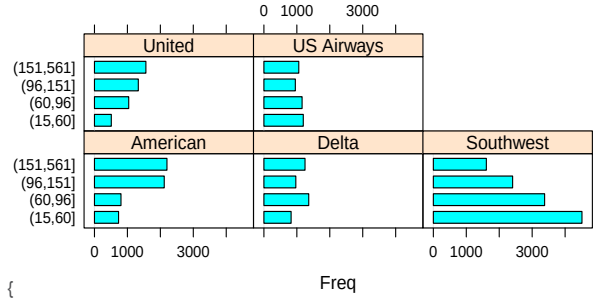



```
{
}
```

Bar chart with separated groups

Plotting barchart using the table we just created, this time with groups separated

```
barchart(tabdata, groups = FALSE)
```



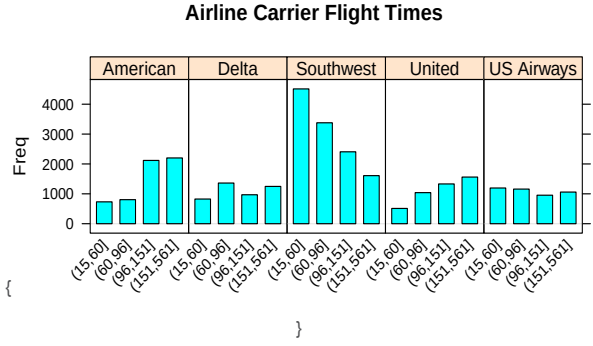

```
{
}
```

Changing layout

Change the direction of the bars, and adjust the axis labels via the scales argument.

```
# Plot previous barchart but stack them with layout and add title
barchart(tabdata, auto.key = TRUE, groups = FALSE, layout = c(5, 1),
  horizontal = FALSE, main = "Airline Carrier Flight Times", scales = list(x = list(rot = 45)))
```

Changing layout



{

}

Exercise

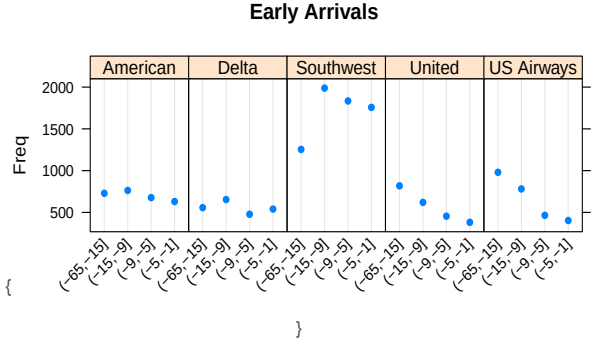
Your turn:

- Shape and plot the data showing early arrival barcharts for each airline carrier.
- You will need to subset the data for arrival delays less than 0.

Outline

Creating a dot plot

```
# Dotplot showing early arrivals
dotplot(tabdata, layout = c(5, 1), auto.key = TRUE, groups = FALSE,
  horizontal = FALSE, scales = list(x = list(rot = 45)), main = "Early Arrivals")
```

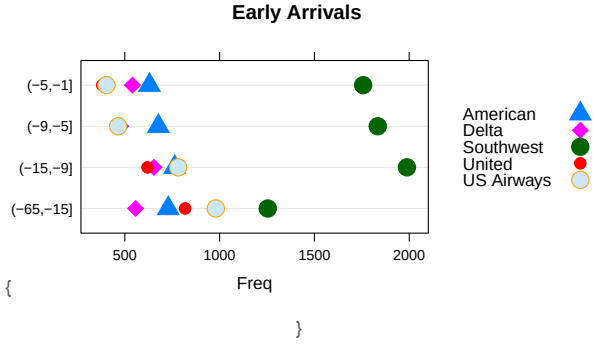


Creating a dot plot

We can modify the appearance of any trellis graphic by changing its parameters using `trellis.par.set()`.

```
# Dotplot showing early arrivals
pars <- trellis.par.get("superpose.symbol")
pars$cex <- 2
pars$pch <- 17:23
trellis.par.set(superpose.symbol = pars)
dotplot(tabdata, auto.key = list(space = "right"), main = "Early Arrivals")
```

Creating a dot plot

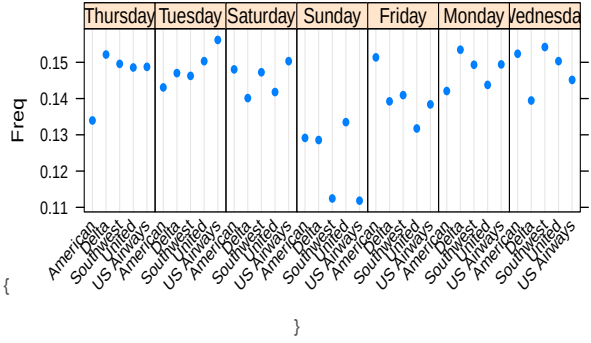


{

}

Creating a dot plot with proportion table

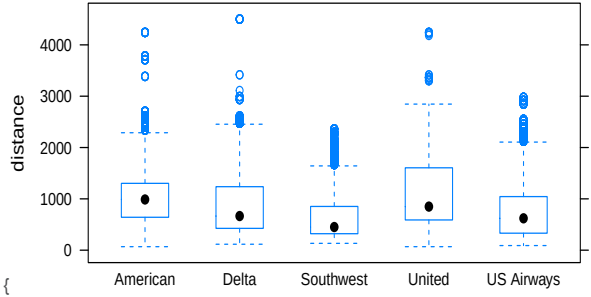
```
tabdata <- with(airline, table(carrier, weekday))
tabdata <- prop.table(tabdata, margin = 1)
dotplot(tabdata, layout = c(7, 1), horizontal = FALSE, scales = list(x = list(rot = 45)),
  auto.key = TRUE, groups = FALSE)
```



Outline

Box and whisker plot

```
# Box and whiskers plot showing each airline carrier's distance
bwplot(distance ~ carrier, data = airline)
```

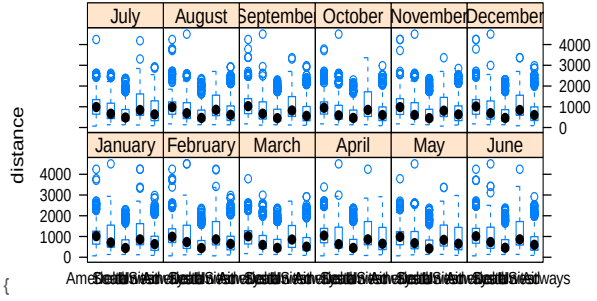



```
{
}
```

Box and whisker plot (conditioned)

Creating a box and whisker plot showing each airline carrier's distance by each month

```
bwplot(distance ~ carrier | month, data = airline)
```

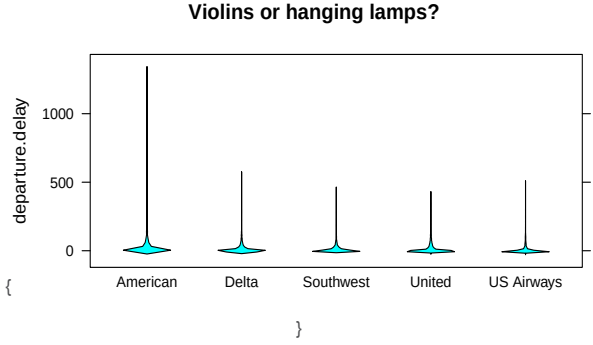


{

}

Violin plot using `bwplot()`

```
# Violin plot using panel.violin
bwplot(departure.delay ~ carrier, data = airline, panel = panel.violin,
  main = "Violins or hanging lamps?")
```



{

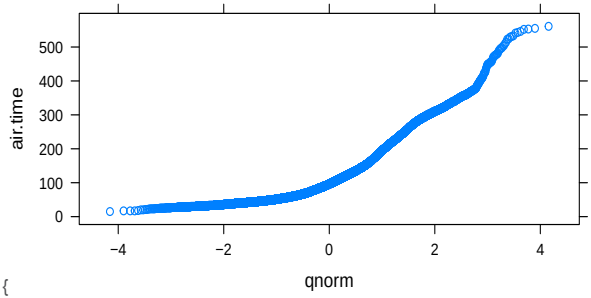
}

Creating a qqplot

A quantile plot enables visual inspection of the distribution of data, compared to a theoretical distribution

- Creating a `qqplot` with `air.time`

```
qqmath(~air.time, data = airline)
```

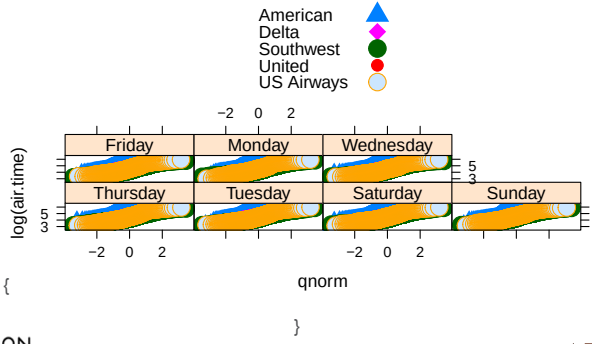


{ }

Using a transform with qqplot

QQ plot with log transform of air.time, airline carrier groups, and weekday panels

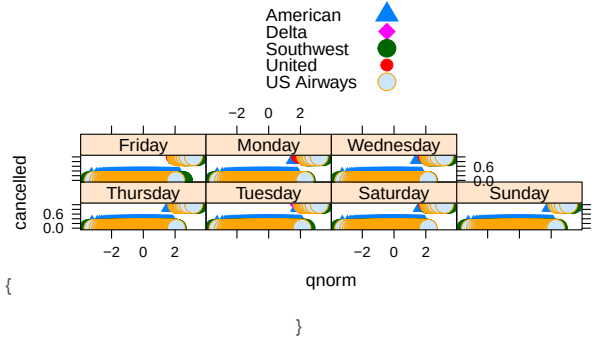
```
qqmath(~log(air.time) | weekday, groups = carrier, data = airline,
  auto.key = TRUE)
```



Plot functions

QQ plot with binary cancellation.code - sigmoidal logistic

```
qqmath(~cancelled | weekday, groups = carrier, data = airline, auto.key = TRUE)
```



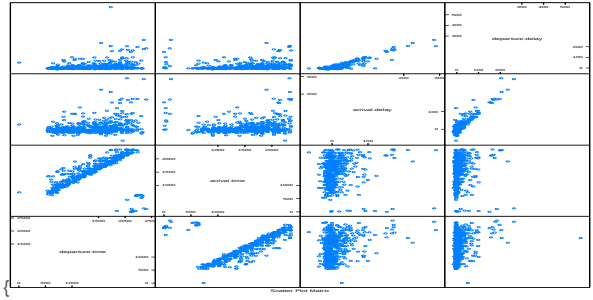
{

}

Outline

Scatterplot matrix

```
# Scatterplot matrix using sample data
set.seed(42)
airSample <- airline[sample.int(nrow(airline), 500), ]
airSample <- airSample[c("departure.time", "arrival.time", "arrival.delay",
  "departure.delay")]
splom(airSample)
```

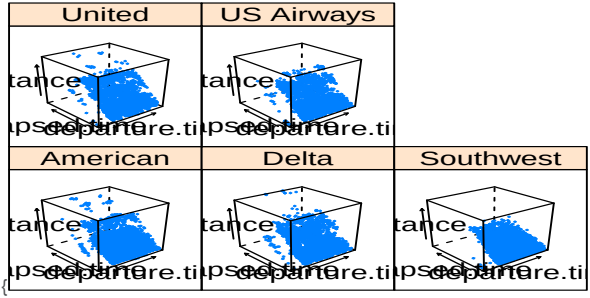



```
}
```

Cloud plot

Cloud plot (3D Scatter) using formula `z ~ x * y`

```
cloud(distance ~ departure.time * elapsed.time | carrier, data = airline)
```



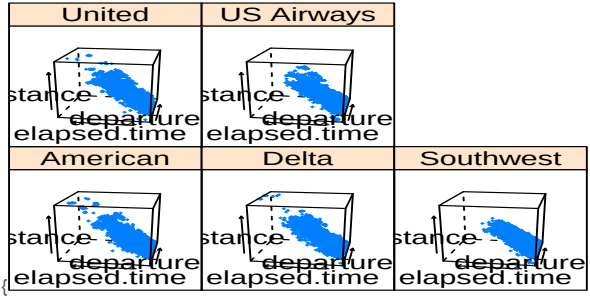

```
}
```

Cloud plot

Creating a cloud plot with rotation and zoom out. Note that the default rotation is `screen = list(z = 40, x = -60)`

For more information, see `?(panel.cloud)`

```
cloud(distance ~ departure.time * elapsed.time | carrier, data = airline,
  screen = list(z = 80, x = -70), zoom = 0.7)
```



Summary

Lattice provides a lot of additional utility for quickly generating some pretty sophisticated graphics.

- Scatter plots (with conditioning variables)
- Bar and dot charts
- Box-and-Whisker Plots
- Histograms and density plots
- Scatter plot matrices and cloud plots

Questions?

REVOLUTION ANALYTICS

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com
1.855.GET.REVO
Twitter: @RevolutionR