REVOLUTION ANALYTICS

Intro to Plotting with ggplot2

Revolution Analytics

Overview

In this session we cover plotting with the very popular add-on package `ggplot2`

Objectives:

- Introduce basic applications of the `ggplot2` package
- Introduce other plotting resources

Outline

Introduction to ggplot2

The package `ggplot2` is a flexible, colorful, and dynamic graphics package.

In this session we introduce `ggplot2`, covering some of the fundamentals and then create two relatively complex examples:

- Dot plots with different facets (`cars`)
- Line Graphs with different groups (Google Finance)

In a separate session we will create maps of airport locations with `ggplot2`.

Introduction to ggplot2

The package `ggplot`:

- Provides an elegant way of developing high quality graphics for reports and publication
- Written by Hadley Wickham
- Based on “Grammar of Graphics” by Wilkinson, 2005

You can find many useful examples and source codes at Hadley’s website

Benefits of ggplot2

- Flexible
  - Composed of independent “components” that can be put together in a variety of ways
- Iterative
  - Works with “layers”. Starts with the raw data then add statistical transformations, annotations, etc.

Basic components of a statistical plot

`data` The data that you are looking to visualize

`mapping` The aesthetic *mapping* that explain a set of mappings that describe the relationship between the variables and visual attributes (lines, dots, bars, etc)

`geom` The *geometric* objects (geoms) i.e. lines, points, polygons, etc.

`stat` The *statistical* transformations (stats) i.e. counts/bins for a histogram, log, etc.

Other components of a ggplot

`scale` The *scales* describe the relationship between data values and values in the aesthetic space via colors, size, shape. Scales draw a legend or axes making it possible to interpret the chart

`coord` The *coordinate system* (coord) controls how spatially-referenced data is mapped – normally a Cartesian coordinate system but also can be polar coordinates, map projections, etc.

`facet` The *faceting* specifications subset the data, if necessary, for lattice (also called trellis plots)

Plot definition

The formal definition of a `ggplot` looks like this:

```
ggplot(data, mapping) +
  layer(
    stat = "",
    geom = "",
    position = "",
    geom_parms = list(),
    stat_params = list(),
  )
```

But don't be overwhelmed - there is a set of easy shortcuts

Layers

Usually use a shortcut to refer to a layer instead of writing out the full specification.

- `+ geom_smooth()` instead of `+ layer(geom="smooth")`
- `+ stat_summary()`
- … etc.: `http://docs.ggplot2.org/current/`

Every `geom` has a default `stat`, every `stat` a default `geom` (but can override)

A simplified and full control function

The package `ggplot2` has two plotting functions:

`qplot()` A quick way of producing a plot that is of a single class of plots (called geoms). It can be a bar chart, line, error bar, density, etc.

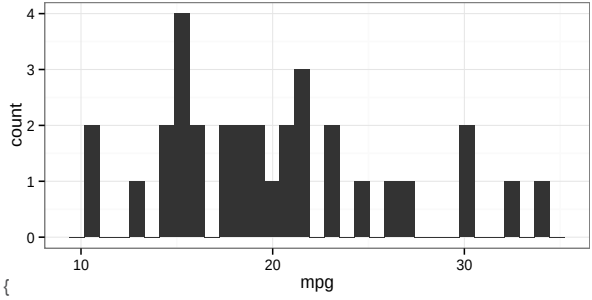
`ggplot()` Provides full functionality to produce hybrid plots (e.g. maps with population plotted in color)

Outline

qplot histogram

Here are a couple examples of `qplot()` based on the `mtcars` dataset.

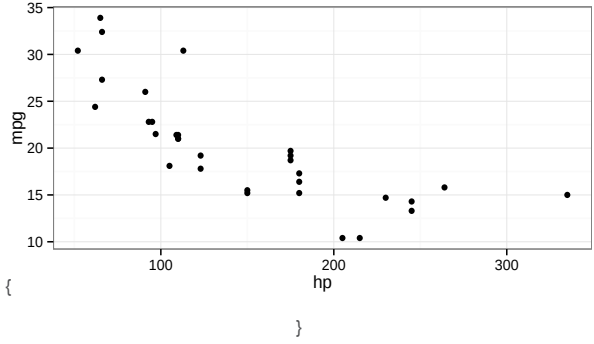
```
library("ggplot2")
theme_set(theme_bw())
qplot(mpg, data = mtcars)
```




```
{
}
```

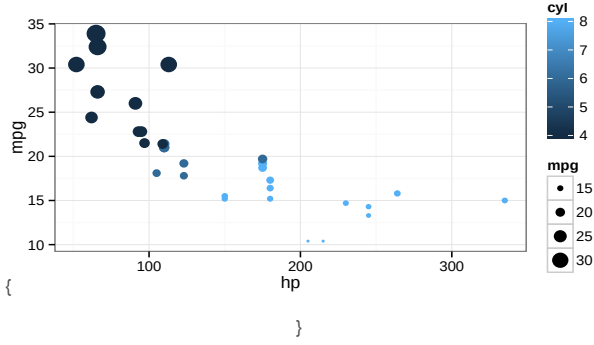
qplot scatter plot

```
qplot(x = hp, y = mpg, data = mtcars)
```



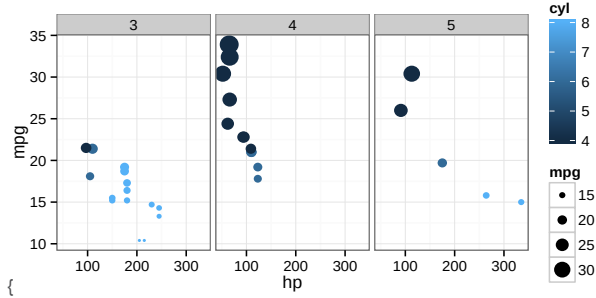
qplot scatter plot with color

```
qplot(x = hp, y = mpg, data = mtcars, color = cyl, size = mpg)
```



qplot scatter plot with color and size

```
qplot(x = hp, y = mpg, data = mtcars, color = cyl, size = mpg, facets = . ~
  gear)
```

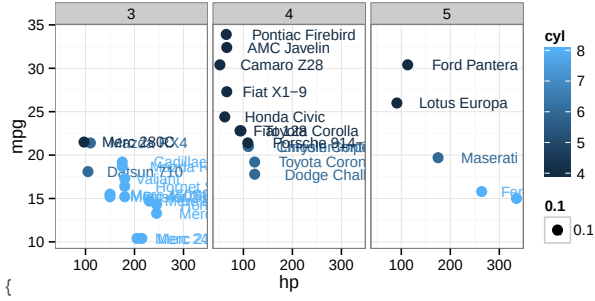


{

}

qplot with facets

```
qplot(x = hp, y = mpg, data = mtcars, color = cyl, facets = . ~ gear,
  label = rownames(mtcars), geom = c("text", "point"), size = 0.1,
  hjust = -0.25)
```



Outline

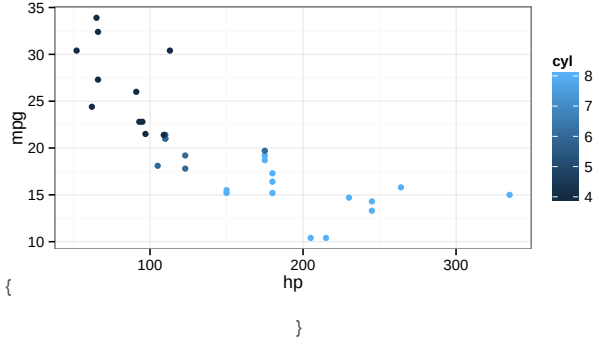
Printing `ggplot()` objects

Note that `p`, in the example below, is a `ggplot` object. You need to print the object if you want to view your graph. If you are running a function or a loop, you will need to use `print(p)`

```
p <- ggplot(data = mtcars, aes(hp, mpg, label = rownames(mtcars))) +
  geom_point(aes(colour = cyl))
```

Printing `ggplot()` objects

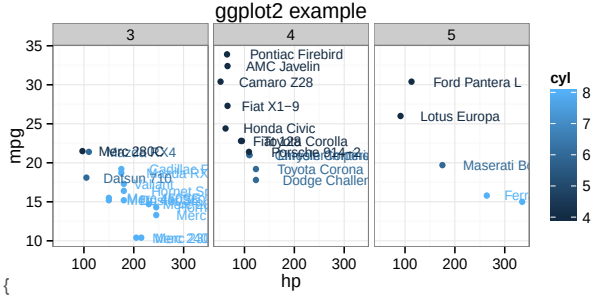
```
print(p)
```




```
{
}
```

Adding labels and titles

```
p <- p + facet_grid(. ~ gear) + geom_text(aes(colour = cyl), size = 3,
  hjust = -0.25) + ggtitle("ggplot2 example")
print(p)
```



Downloading financial data

Now let's use an example of Microsoft, Google, and Apple stock data. First we import the data and clean it up.

```
msft.url <- "http://www.google.com/finance/historical?q=NASDAQ:MSFT&output=csv"
goog.url <- "http://www.google.com/finance/historical?q=NASDAQ:GOOG&output=csv"
aapl.url <- "http://www.google.com/finance/historical?q=NASDAQ:AAPL&output=csv"

msft.data <- read.table(msft.url, header = TRUE, sep = ",")
msft.data$name <- "MSFT"
goog.data <- read.table(goog.url, header = TRUE, sep = ",")
goog.data$name <- "GOOG"
aapl.data <- read.table(aapl.url, header = TRUE, sep = ",")
aapl.data$name <- "AAPL"
stock.data <- rbind(msft.data, goog.data, aapl.data)
```

Read the data from local copy

We have previously stored a copy of this data on a local drive. Use `read.csv()` to import the data:

```
stock.data <- read.csv("../data/stock_data.csv")
head(stock.data)
```

```
##         Date  Open  High   Low Close   Volume name
## 1 2012-03-13 32.24 32.69 32.15 32.67 48347339 MSFT
## 2 2012-03-12 31.97 32.20 31.82 32.04 34076755 MSFT
## 3 2012-03-09 32.10 32.16 31.92 31.99 34628398 MSFT
## 4 2012-03-08 32.04 32.21 31.90 32.01 36752011 MSFT
## 5 2012-03-07 31.67 31.92 31.53 31.84 34340619 MSFT
## 6 2012-03-06 31.54 31.98 31.49 31.56 51938950 MSFT
```

```
# stock.data$Date <- as.Date(stock.data$Date , '%d-%b-%y')
stock.data$Date <- as.Date(stock.data$Date)
```

Plotting financial data

Plot the time series:

```
ggplot(stock.data, aes(x = Date, y = Close, group = name)) + geom_line(aes(color = name))
```

{

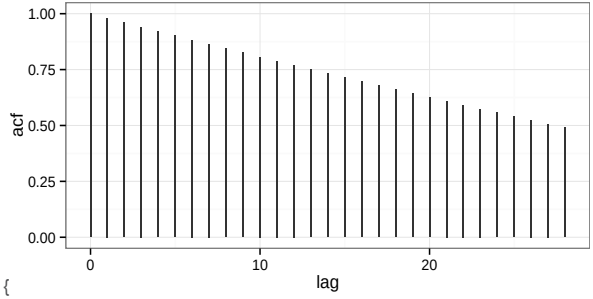
}

Plotting a quick correlogram

- Checks for the leading and lagging relationship between two or more time series objects
- Note that this is not the same as the “corrgram” function

```
bacf = acf(x = stock.data$Date, plot = F)
bacfdf <- with(bacf, data.frame(lag, acf))
ggplot(data = bacfdf, mapping = aes(x = lag, y = acf)) + geom_bar(stat = "identity",
  position = "identity", width = 0.1)
```

Plot a Correlogram



{

}

Create an index and normalise

MSFT is way down there: let's normalize by the day 1 value.

```
stock.data.norm <- stock.data[, c("Date", "Close", "Volume", "name")]
stock.data.norm <- do.call("rbind", by(stock.data.norm, stock.data$name,
  function(STOCK) {
    STOCK$Close <- STOCK$Close/STOCK$Close[nrow(STOCK)]
    return(STOCK)
  }))
```

Plot the normalised data

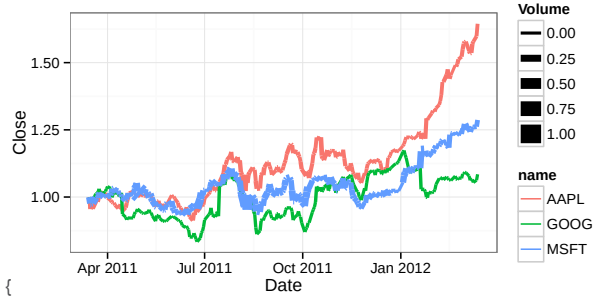
```
ggplot(stock.data.norm, aes(x = Date, y = Close, group = name)) +
  geom_line(aes(color = name))
```

{

}

Add trading volume data

```
stock.data.norm$Volume <- stock.data.norm$Volume/max(stock.data.norm$Volume)
ggplot(stock.data.norm, aes(x = Date, y = Close, group = name)) +
  geom_line(aes(color = name, size = Volume))
```



{

}

Exercise: Play with ggplot2

Your turn:

- Think of an interesting, informative graphic that you would like to generate based on the performance dataset.
- Start with some a basic `qplot()` or two, then move on to working with `ggplot()`.

You have 15 minutes.

Outline

Other plotting resources

A great intro to ggplot2

This is just the very beginning. R's graphics capabilities go way beyond what we've shown so far.

Following are a few examples from from
https://www.facebook.com/pages/R-Graph-Gallery/169231589826661 &
http://gallery.r-enthusiasts.com/

Module review questions

- What are advantages of `base` graphics compared to `ggplot2`?
- How can a user set an R graphics window to have multiple panes?

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