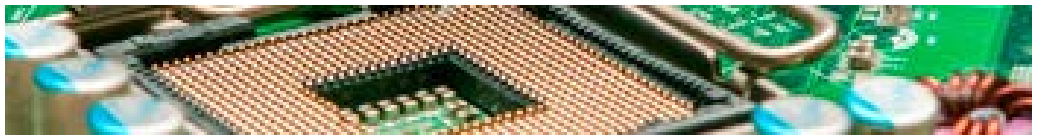
Data Science (CDA)

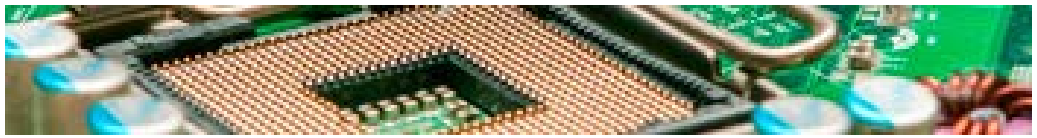
Creating plots with ggplot2

- José Hernández Orallo, DSIC, UPV, jorallo@dsic.upv.es
- Fernando Martínez Plumed, DSIC, UPV, fmartinez@dsic.upv.es



ggplot2: introduction

- ggplot2
  - Ggplot2 is an R library based on a grammar for graphics from Leland Wilkinson.
  - Implemented by Hadley Wickham.
  - It adds up to other graphical systems in R (**base** and **lattice**).
  - It is available on CRAN with the instruction install.packages().
    - install.packages('ggplot2')
    - library('ggplot2')

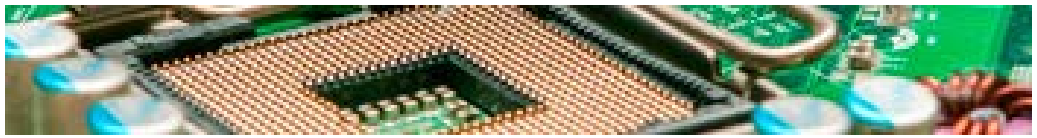


ggplot2: introduction

- Within ggplot2, there are two basic methods to create plots:
  - **qplot()** comes from "quick plotting":
    - Shortcut designed to be familiar if you're used to base plot().
    - It makes a number of assumptions that speed most cases.
    - Allows the mapping of aesthetic variables in a simple way.
    - Allows for facets.
  - **ggplot()**
    - Based on layers:

      Data + geoms + stats + scales + coords + facets +…
    - More advanced features. Much more flexible.

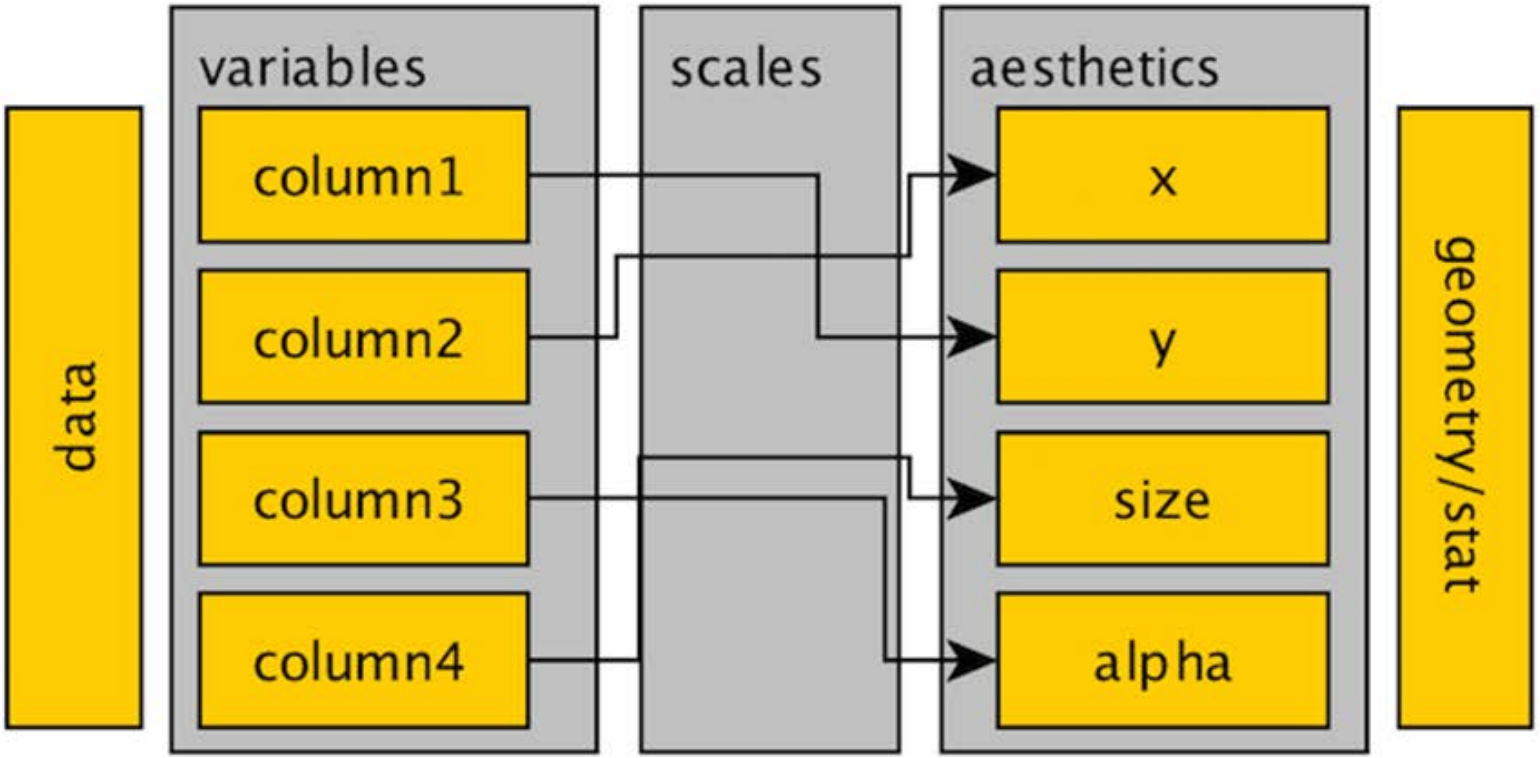
Note: qplot() can also use all the grammar in ggplot(), such as "+".

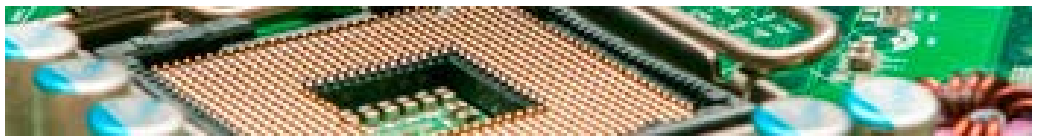


ggplot2: introduction

qplot(**x** = c1, **y** = c2, **data**, **color** = c3, **geom** = "points", ...)

ggplot(**data**, aes(**x** = c1, **y** = c2, **color** = c3)) + **geom_point**()





ggplot2: introduction

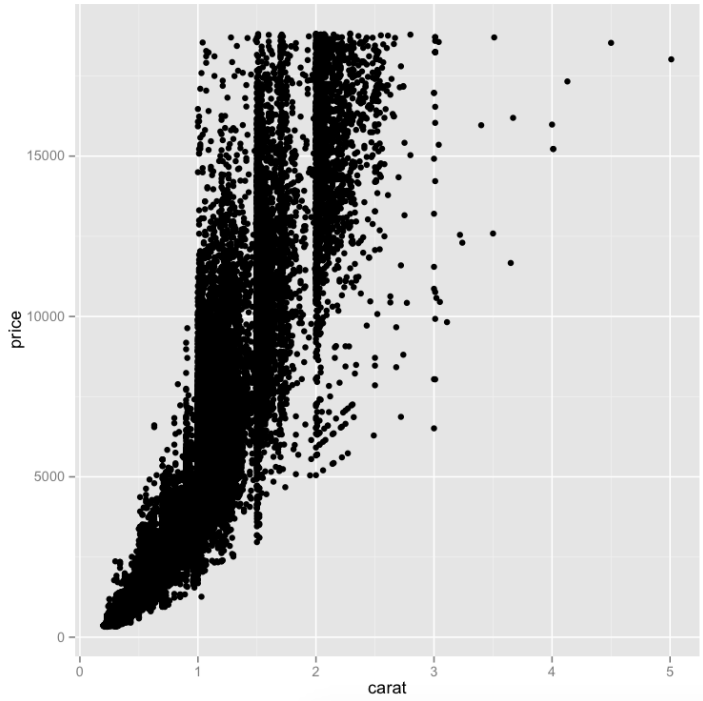
- Basic syntax for qplot() & ggplot():

qplot

*qplot(**x_coord**, **y_coord**, data=**data_frame**)*

qplot(carat, price, data=diamonds)

Translating between qplot and ggplot

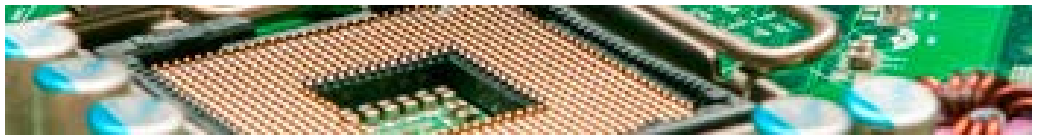


ggplot2

*ggplot(data=**data_frame**, aes(**x_coord**, **y_coord**)) + **geom_XXX**()*

ggplot(diamonds, aes(carat, price)) +
geom_point()

The **diamonds** dataset is in the ggplot package: diamonds (vars: *carat, cut, color, clarity, depth, table, price, x, y, z*)

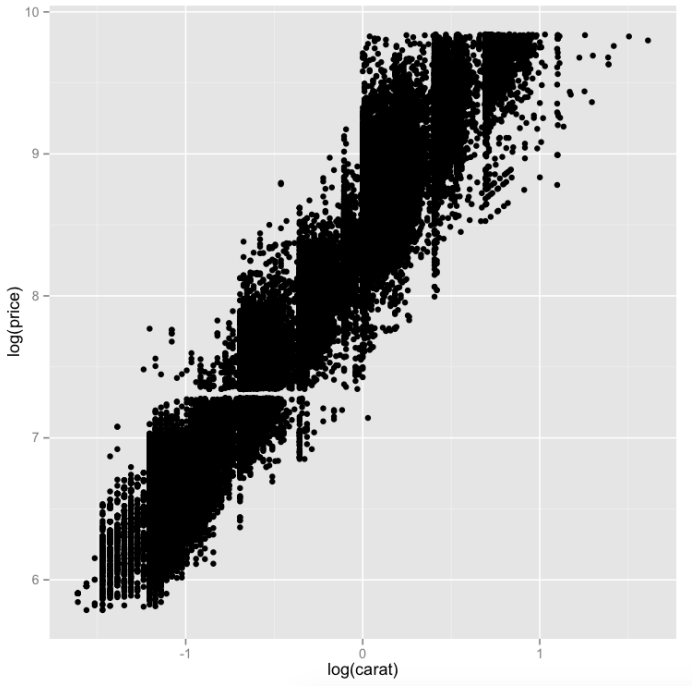


ggplot2: introduction

- The arguments in qplot()/ggplot() can also combine variables:

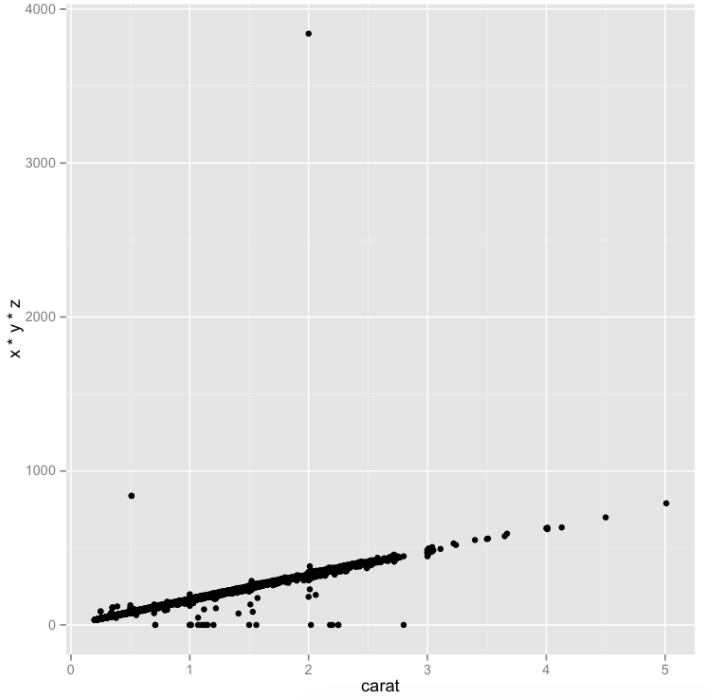
qplot

qplot(**log(carat)**, **log(price)**, data = diamonds)



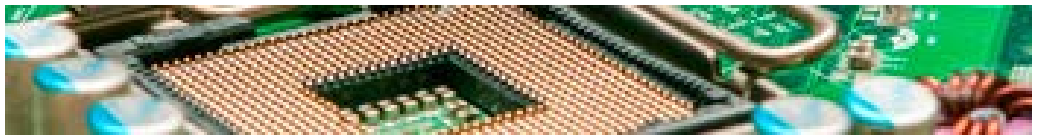
qplot

qplot(carat, **x*y*z**, data=diamonds)



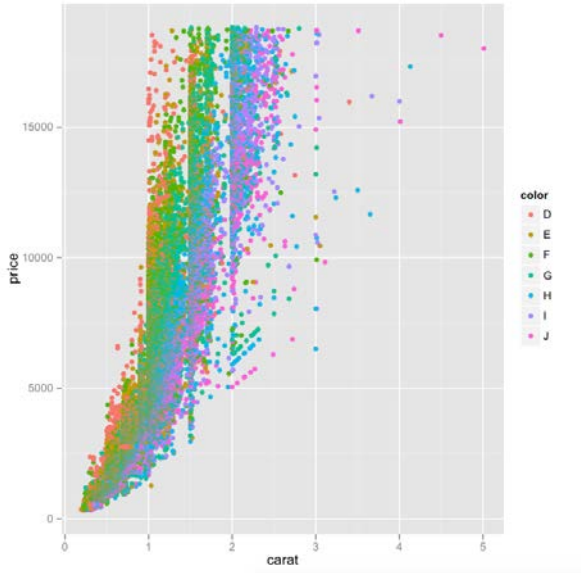
ggplot2

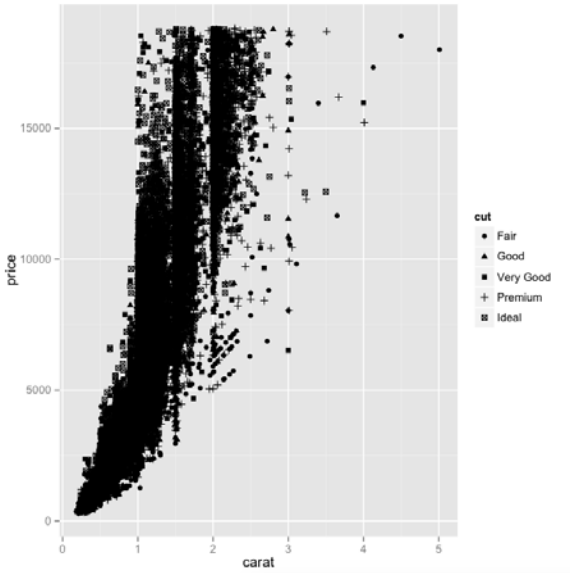
ggplot(diamonds, aes(**log(price)**, **log(carat)**) + geom_point()
ggplot(diamonds, aes(carat, **x*y*z**)) + geom_point()

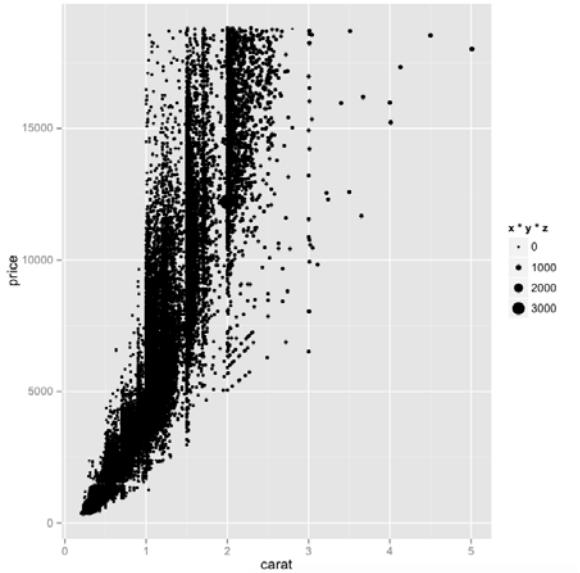


ggplot2: aesthetic attributes.

- qplot()/ ggplot() make it easy to assign colours and shapes to the points.
- The attributes are: ***size*** = *var*, ***colour*** = *var*, ***shape*** = *var*





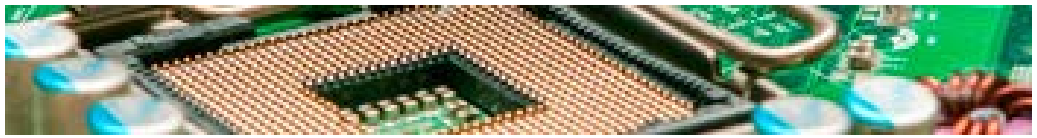


qplot

```
qplot(carat, price, data=diamonds, colour=color)
qplot(carat, price, data=diamonds, shape=cut)
qplot(carat, price, data=diamonds, size=x*y*z)
```

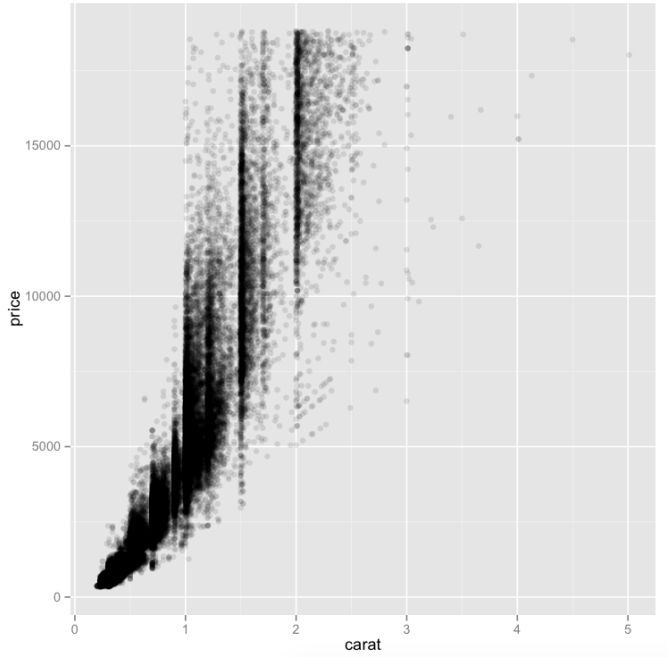
ggplot2

```
ggplot(diamonds, aes(carat, price, colour = color) + geom_point()
ggplot(diamonds, aes(carat, price, shape = cut) + geom_point()
ggplot(diamonds, aes(carat, price, size = x*y*z) + geom_point()
```

ggplot2: aesthetic attributes.

- With many points, semi-transparent points can be helpful.
- The attribute for semi-transparent points is:
  - ***alpha***= value
- Taking values between 0 and 1.

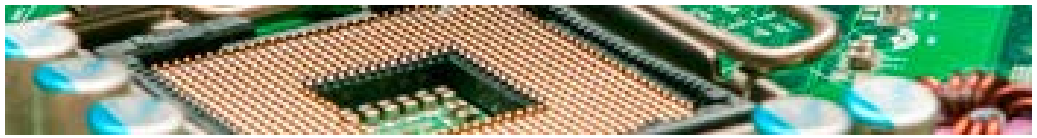


qplot

```
qplot(carat, price, data=diamonds,
alpha=0.1)
```

ggplot2

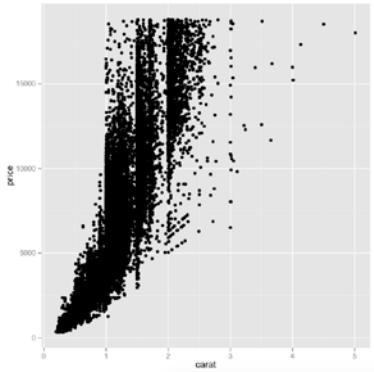
```
ggplot(diamonds, aes(carat, price), alpha=0.1) +
geom_point()
```

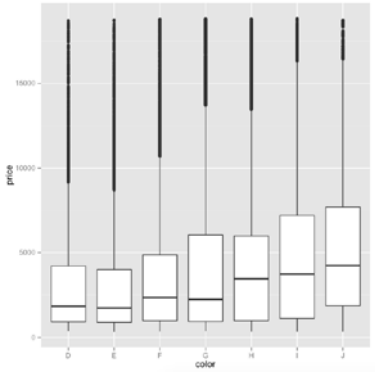
ggplot2: geometries

- Different "geometries" can be specified with:
  - geom
- Leading to the following 1D/2D things:

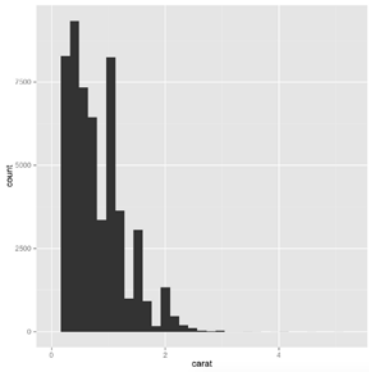
Points



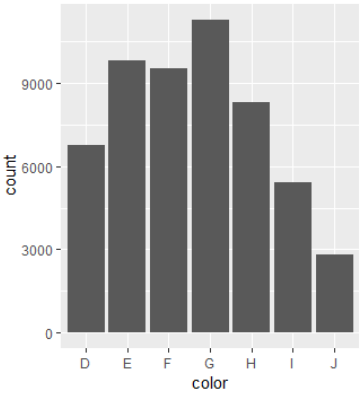
Boxplot



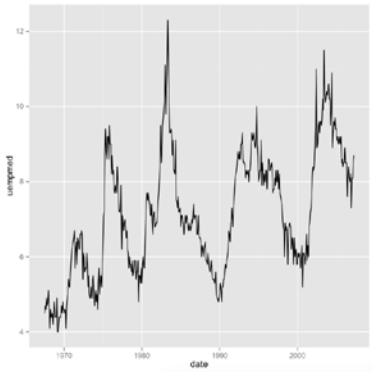
Histogram



Bar



Line

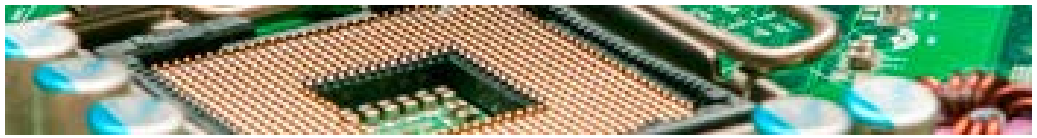


qplot

```
qplot(carat, price, data=diamonds, geom="point")
qplot(color, price, data=diamonds, geom="boxplot")
qplot(carat, data=diamonds, geom="histogram",binwidth=0.1)
qplot(color, data= diamonds, geom="bar")
qplot(date, uempmed, data=economics, geom="line")
```

ggplot2

```
ggplot(diamonds, aes(carat, price)) + geom_point()
ggplot(diamonds, aes(color, price)) + geom_boxplot()
ggplot(diamonds, aes(carat)) + geom_histogram(binwidth=0.1)
ggplot(diamonds, aes(color)) + geom_bar()
ggplot(economics, aes(date, uempmed)) + geom_line()
```

ggplot2: geometries

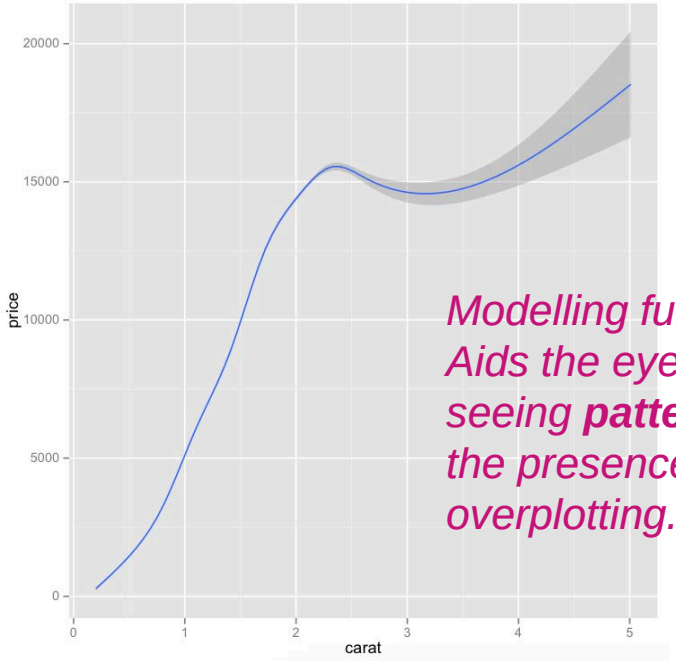
- Smoothers:
  - geom = "smooth".
- Examples:

qplot

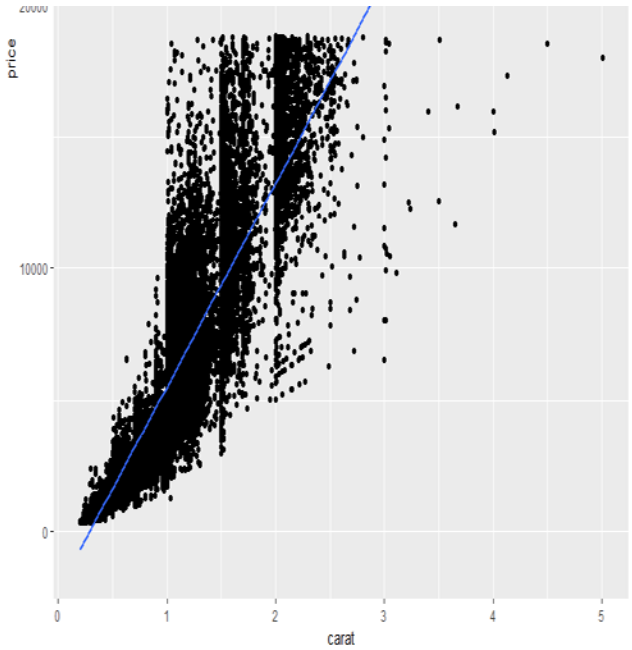
```
qplot(carat, price, data=diamonds, geom="smooth"))

qplot(carat, price, data=diamonds, geom=c("point", "smooth"),
method="lm", se= FALSE)
```

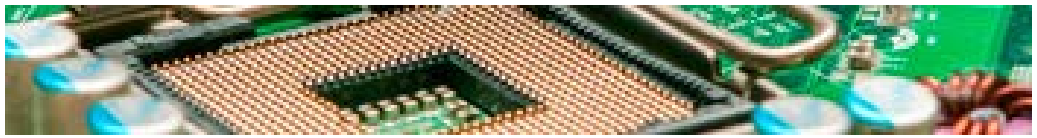
ggplot2

```
ggplot(diamonds, aes(carat, price)) + geom_smooth()

ggplot(diamonds, aes(carat, price)) + geom_point() +
geom_smooth(se=FALSE, method = "lm")
```



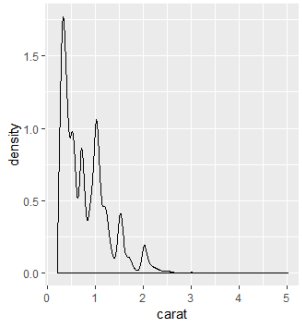
*Modelling functions: Aids the eye in seeing **patterns** in the presence of overplotting.*

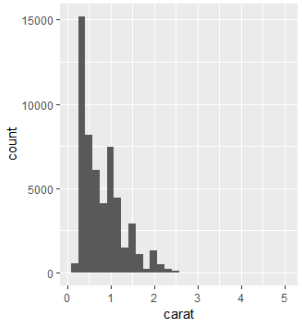


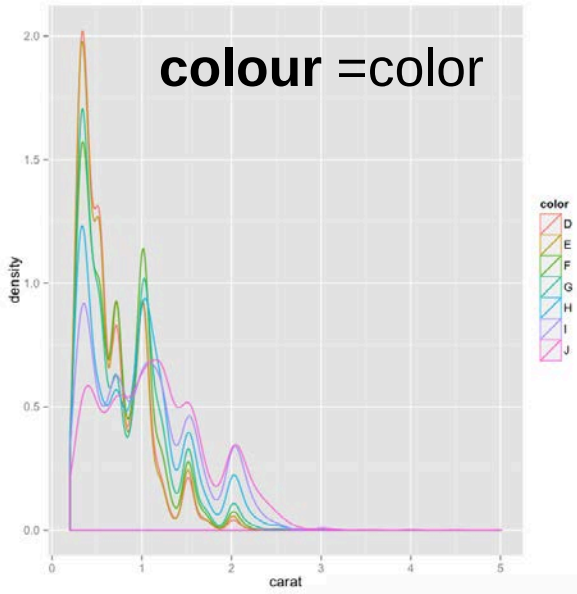


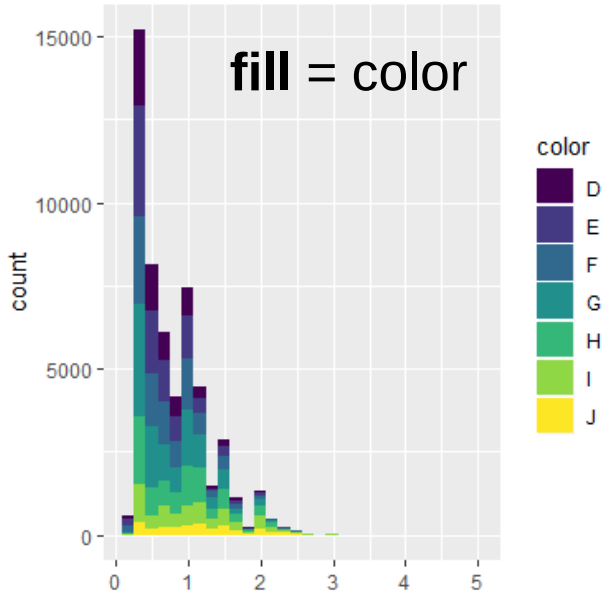
ggplot2: geometries

- Histograms and density functions:
  - To compare distributions for different groups, we can add aesthetics:







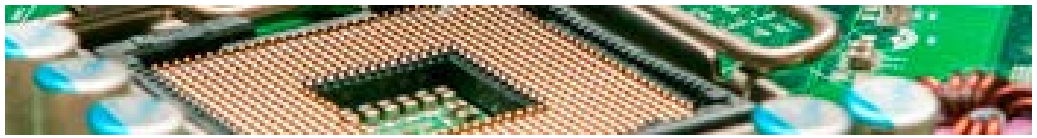


qplot

```
qplot(carat, data=diamonds, geom="density", colour=color)

qplot(carat, data=diamonds, geom="histogram", fill=color)
```

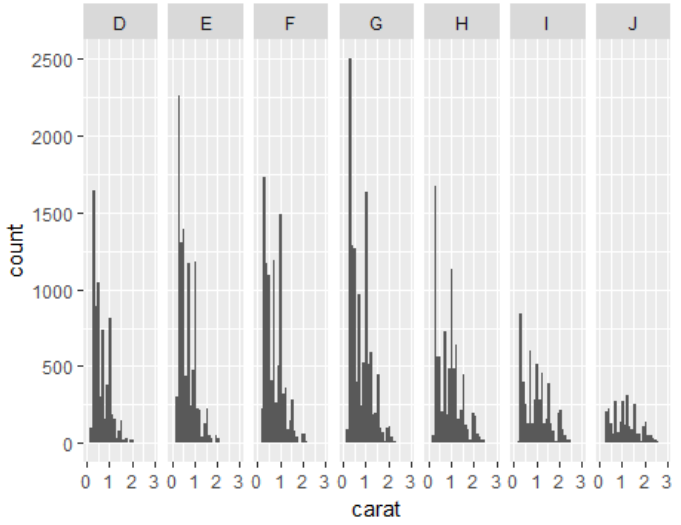
ggplot2

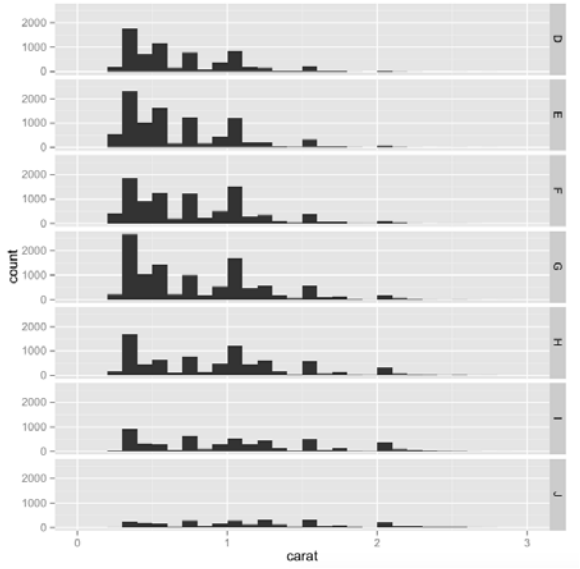
```
ggplot(diamonds, aes(carat, colour=color)) + geom_density()

ggplot(diamonds, aes(carat, fill=color)) + geom_histogram()
```

ggplot2: facets

- Facets: split up your data by one or more variables and plot the subsets of data together.
  - *facet* = variable1~variable2 : grid
  - *facet* = variable1~. : a column.
  - *facet* = .~variable2 : a row





qplot

```
qplot(carat, data=diamonds, facets=.~color, geom="histogram", binwidth=0.1, xlim=c(0,3))
```

ggplot2

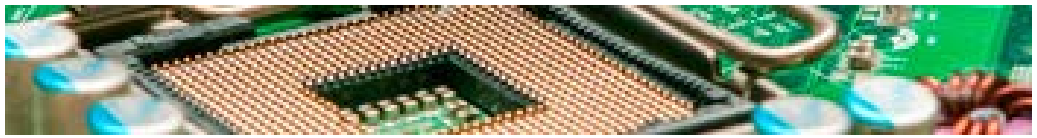
```
ggplot(diamonds, aes(carat)) + geom_histogram(bindwidth=0.1) + facet_grid(.~color)+xlim(0,3)
```

qplot

```
qplot(carat, data=diamonds, facets=color~., geom="histogram", binwidth=0.1, xlim=c(0,3))
```

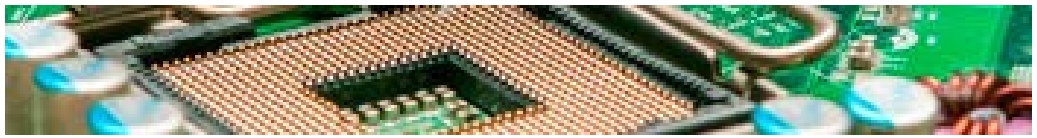
ggplot2

```
ggplot(diamonds, aes(carat)) + geom_histogram(bindwidth=0.1) + facet_grid(color~.)+xlim(0,3)
```

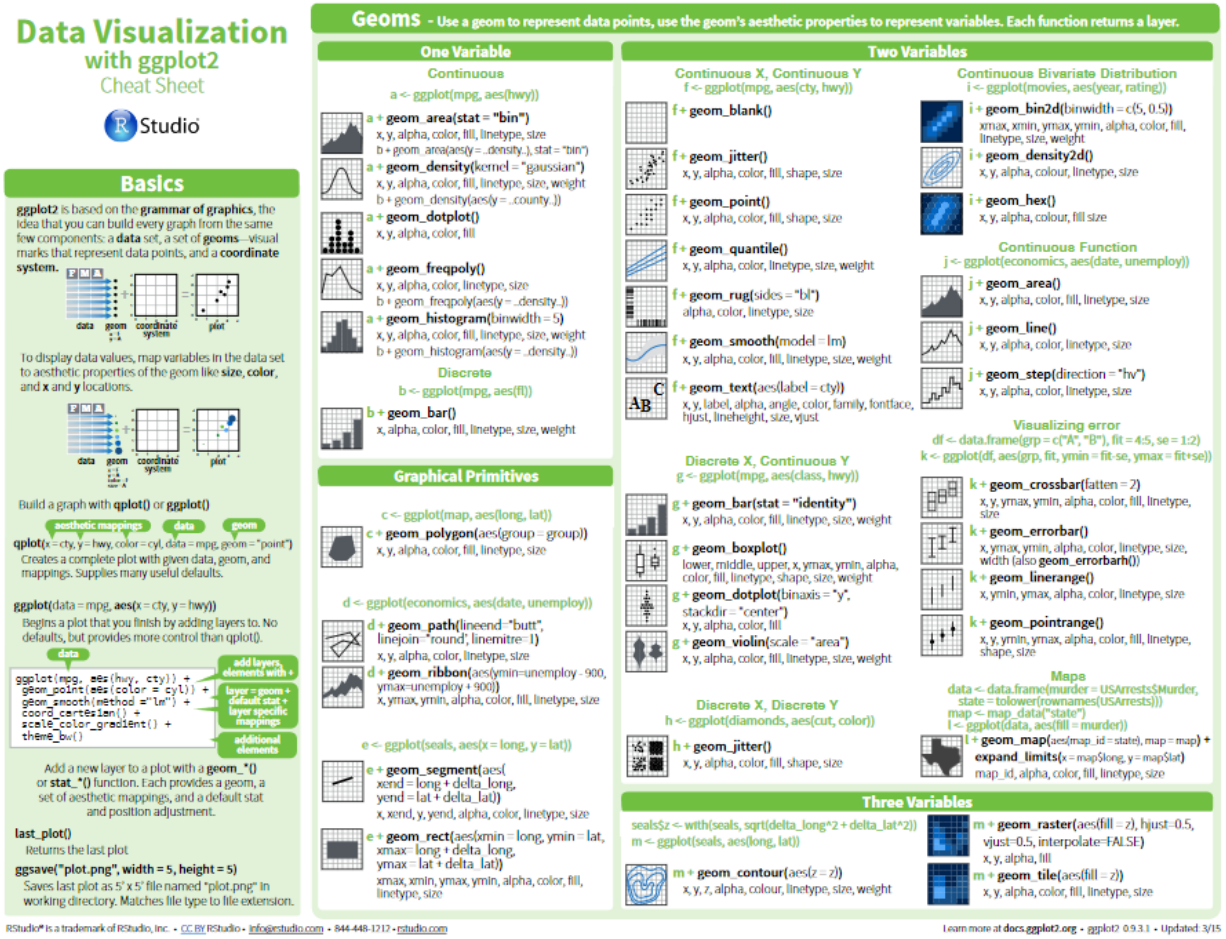



ggplot2: other options

- Other attributes in qplot/ggplot (many just like plot):
  - **xlim** : limits the values of the x-axis (e.g., xlim=c(0,3)).
  - **ylim** : limits the values of the y-axis (e.g., ylim=c(0,3)).
  - **main**: title of the plot (e.g., main="mytitle").
  - **xlab** : label of x.
  - **ylab** : label of y.



ggplot2: Cheat Sheet



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 few components: a data set, a set of geoms—visual marks that represent data points, and a coordinate system.

data geom coordinate system plot

To display data values, map variables in the data set to aesthetic properties of the geom like size, color, and x and y locations.

data geom coordinate system plot

Build a graph with qplot() or ggplot()

aesthetic mappings data geom

```
qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")
```

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

```
ggplot(data = mpg, aes(x = cty, y = hwy))
```

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

data

```
ggplot(mpg, aes(hwy, cty)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method ="lm") +
  coord_cartesian() +
  scale_color_gradient() +
  theme_bw()
```

add layers, elements with +

layer = geom + default stat + layer specific mappings

additional elements

Add a new layer to a plot with a geom_*() or stat_*() function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

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

One Variable

Continuous

```
a <- ggplot(mpg, aes(hwy))
```

a + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + geom_dotplot()
x, y, alpha, color, fill

a + geom_freqpoly()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

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

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

Graphical Primitives

```
c <- ggplot(map, aes(long, lat))
```

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

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

d + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, linetype, size

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

```
e <- ggplot(seals, aes(x = long, y = lat))
```

e + geom_segment(aes(xend = long + delta_long, yend = lat + delta_lat))
x, xend, y, yend, alpha, color, linetype, size

e + geom_rect(aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

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

f + geom_blank()

f + geom_jitter()
x, y, alpha, color, fill, shape, size

f + geom_point()
x, y, alpha, color, fill, shape, size

f + geom_quantile()
x, y, alpha, color, linetype, size, weight

f + geom_rug(sides = "bl")
alpha, color, linetype, size

f + geom_smooth(model = lm)
x, y, alpha, color, fill, linetype, size, weight

f + geom_text(aes(label = cty))
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

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

g + geom_bar(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

g + geom_boxplot()
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + geom_dotplot(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill

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

Discrete X, Discrete Y

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

h + geom_jitter()
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

```
i <- ggplot(movies, aes(year, rating))
```

i + geom_bin2d(binwidth = c(5, 0.5))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + geom_density2d()
x, y, alpha, colour, linetype, size

i + geom_hex()
x, y, alpha, colour, fill size

Continuous Function

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

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

j + geom_line()
x, y, alpha, color, linetype, size

j + geom_step(direction = "hv")
x, y, alpha, color, linetype, size

Visualizing error

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

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

k + geom_errorbar()
x, ymax, ymin, alpha, color, linetype, size, width (also geom_errorbarh())

k + geom_linerange()
x, ymin, ymax, alpha, color, linetype, size

k + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

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

l + 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))
m <- ggplot(seals, aes(long, lat))
```

m + geom_contour(aes(z = z))
x, y, z, alpha, colour, linetype, size, weight

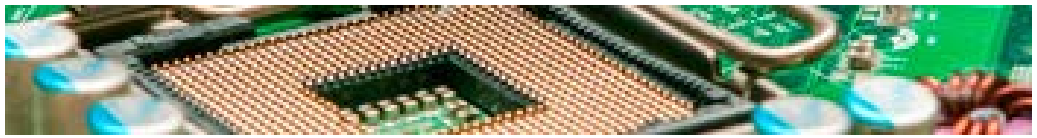
m + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

m + geom_tile(aes(fill = z))
x, y, alpha, color, fill, linetype, size

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

Learn more at docs.ggplot2.org • ggplot2 0.9.3.1 • Updated: 3/15

https://github.com/rstudio/cheatsheets/raw/master/data-visualization.pdf



ggplot2: references

- More references:
  - H. Wickham, **ggplot2: Elegant Graphics for Data Analysis**, Ed. Springer, ISBN: 978-0-387-98140-6, 2011.
  - C. Chen, W. Härdle, A. Unwin, **Handbook of Data Visualization**, Ed. Springer, ISBN: 978-3-540-33036-3, 2008.
  - L. Wilkinson, **The Grammar of Graphics**, Ed. Springer, 2nd edition, ISBN: 978-1-4419-2033-1, 2005.
  - Web site: https://ggplot2.tidyverse.org/
  - Extensions: https://exts.ggplot2.tidyverse.org/gallery/