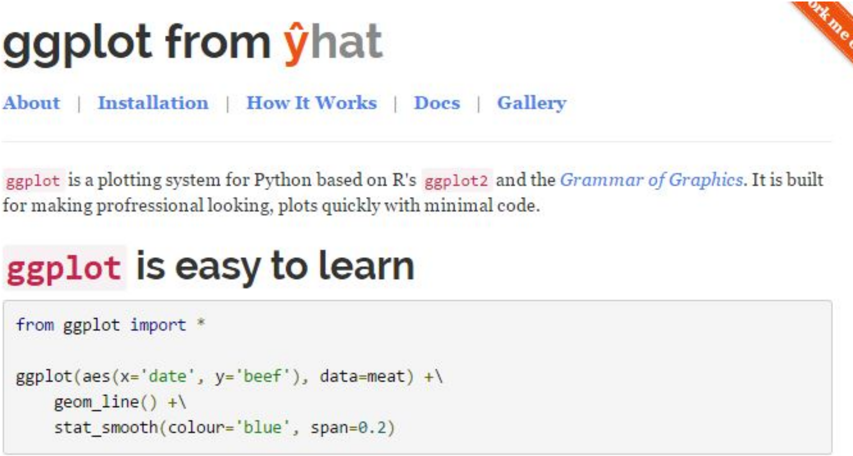
About ggplot

ggplot is a graphics package for Python that aims to approximate `R`'s ggplot2 package in both usage and aesthetics.

Authors: Greg Lamp and Austin Ogilvie

Website: `ggplot.yhathq.com`



ggplot from ŷhat

About | Installation | How It Works | Docs | Gallery

`ggplot` is a plotting system for Python based on R's `ggplot2` and the *Grammar of Graphics*. It is built for making profressional looking, plots quickly with minimal code.

`ggplot` is easy to learn

```
from ggplot import *

ggplot(aes(x='date', y='beef'), data=meat) +\
    geom_line() +\
    stat_smooth(colour='blue', span=0.2)
```

Important :

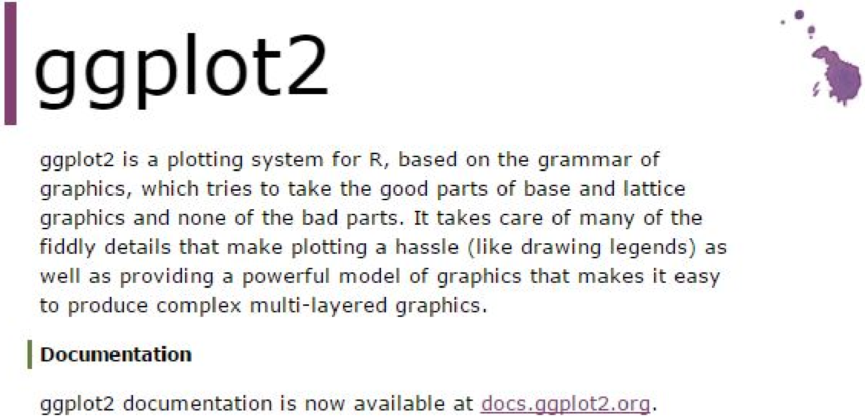
For Python, the name is simply “**ggplot**”.

What are Yhat saying?

1. ggplot is easy to learn [*1*]
2. ggplot is fun
3. ggplot is powerful [*2*]

[1] Lots of learning resources, mainly intended for the R environment, that can applied to Python also.

[2] Less code required to compute high-level publication quality plot



ggplot2

ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

Documentation

ggplot2 documentation is now available at docs.ggplot2.org.

website: www.had.co.nz

Hadley Wickham (Chief Data Scientist, RStudio)

Use R!

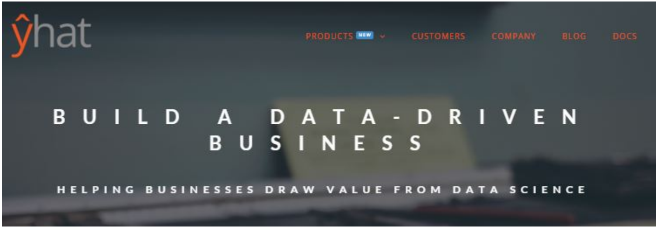
Hadley Wickham

ggplot2

Elegant Graphics for Data Analysis

Springer

ggplot2: Elegant Graphics for Data Analysis



BUILD A DATA-DRIVEN BUSINESS

HELPING BUSINESSES DRAW VALUE FROM DATA SCIENCE

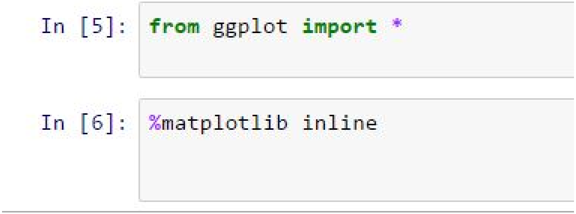
Yhat (*pronounced y-hat*) is a data science technology company that provides tools and systems that allow enterprises to turn data insights into data-driven products.

1. Installing ggplot

```
pip install ggplot
```

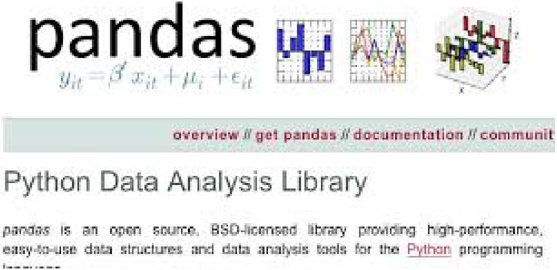
N.B. Not loaded automatically with Anaconda, unlike pandas and numpy

2. Getting Set Up on Jupyter Notebook



```
In [5]: from ggplot import *

In [6]: %matplotlib inline
```


pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
overview // get pandas // documentation // communit

Python Data Analysis Library

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming languag

Important: ggplot accepts data in the form of a `pandas Dataframe`, so you need to configure all data accordingly first.

Data

- ggplot has a symbiotic relationship with pandas.
- If you're planning on using ggplot, it's best to keep your data in DataFrames.
- Think of a DataFrame as a tabular data object.

ggplot - Inbuilt Data Sets

The ggplot package contains these inbuilt pandas `DataFrame`.

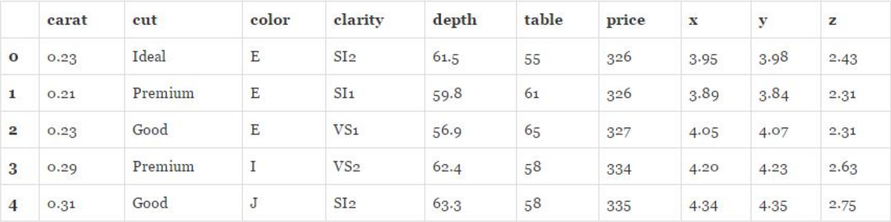
- diamonds
- meat (also: derived data set called meat2)
- movies
- mtcars
- pageviews

Remark: ggplot code will work on any pandas `DataFrame`.

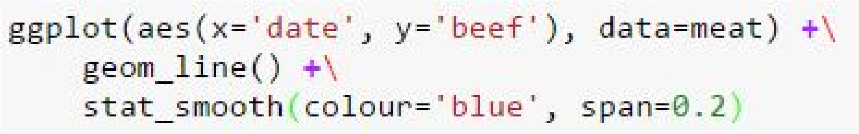
```
import pandas as pd

meat2 = pd.melt(meat, id_vars=[’date’])
```

```
from ggplot import *
diamonds.head()
```



	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
2	0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
3	0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
4	0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75



```
ggplot(aes(x='date', y='beef'), data=meat) +\
    geom_line() +\
    stat_smooth(colour='blue', span=0.2)
```

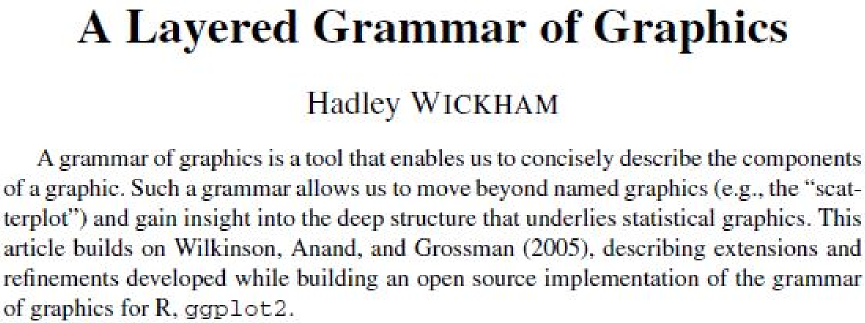
For R Users

- In Python, dataframe columns (i.e. variables) are specified with quotation marks
- Watch out for this operator

```
. . . . +\ . . . . .
```

- The main command is `ggplot()`.
- The name comes from ”**grammar of graphics**”, a book by Leland Wilkinson
- A very “high-level” approach to data visualization.

> *A grammar of graphics is a tool that enables us to concisely describe the components of a graphic. Such a grammar allows us to move beyond named graphics (e.g., the scatterplot) and gain insight into the deep structure that underlies statistical graphics.*



A Layered Grammar of Graphics

Hadley WICKHAM

A grammar of graphics is a tool that enables us to concisely describe the components of a graphic. Such a grammar allows us to move beyond named graphics (e.g., the "scatterplot") and gain insight into the deep structure that underlies statistical graphics. This article builds on Wilkinson, Anand, and Grossman (2005), describing extensions and refinements developed while building an open source implementation of the grammar of graphics for R, `ggplot2`.

Hadley Wickham.
A layered grammar of graphics.
Journal of Computational and Graphical Statistics,
vol. 19, no. 1, pp. 328, 2010.

Basic Premise

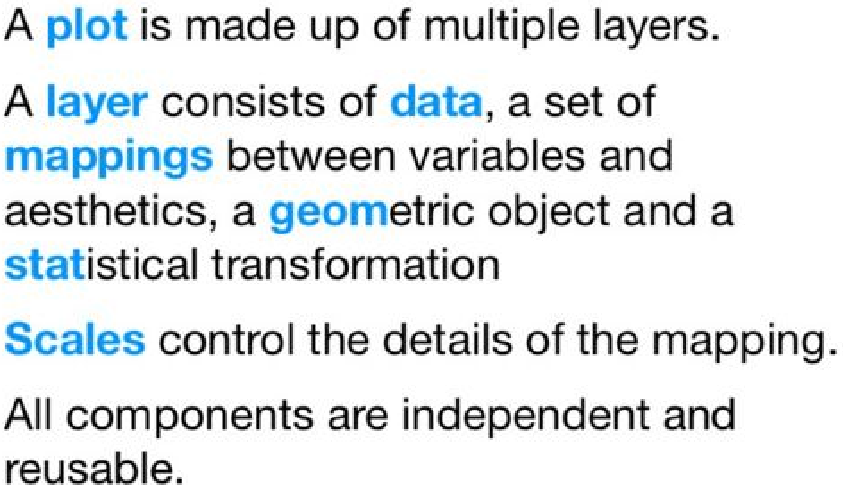
- Making plots is a very repetive: draw this line, add these colored points, then add these, etc.
- Instead of re-using the same code over and over, `ggplot` implements them using a high-level but very expressive API.
- The result is less time spent creating your charts, and more time interpreting what they mean.

(From ggplot documentation)

Basic Premise

- `ggplot` is not a good fit for people trying to make highly customized data visualizations.
- *(Compare this to high level "Bokeh" plots)*
- While you can make some very intricate, great looking plots, `ggplot` sacrifices highly customization in favour of general doing "what you'd expect".

(From ggplot documentation)



A **plot** is made up of multiple layers.

A **layer** consists of **data**, a set of **mappings** between variables and aesthetics, a **geom**etric object and a **stat**istical transformation

Scales control the details of the mapping.

All components are independent and reusable.

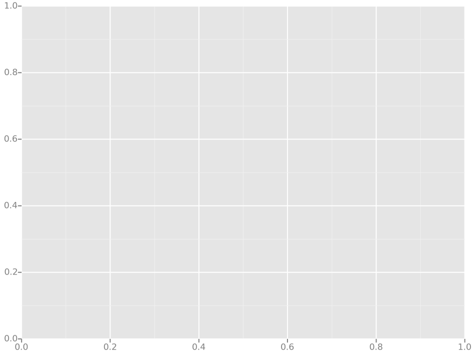
Layers

Layers

- **ggplot** lets you combine or add different types of visualization components (or layers) together.
- The command `ggplot` does not actually create any plot, rather it prepares a “blank canvas” for further plotting
- We will introduce *geoms* shortly

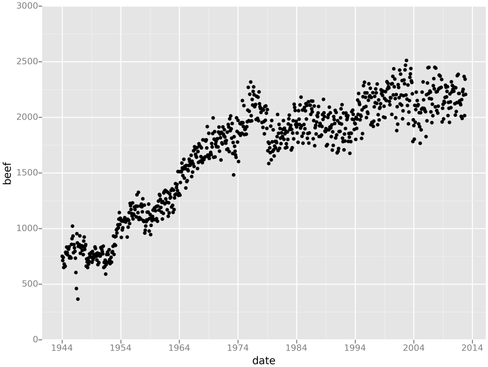
Start with a blank canvas.

```
p = ggplot(aes(x="date", y="beef"), data=meat)
p
```



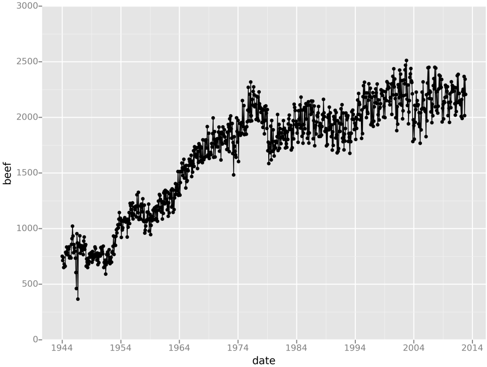
Add some points.

```
p + geom_point()
```



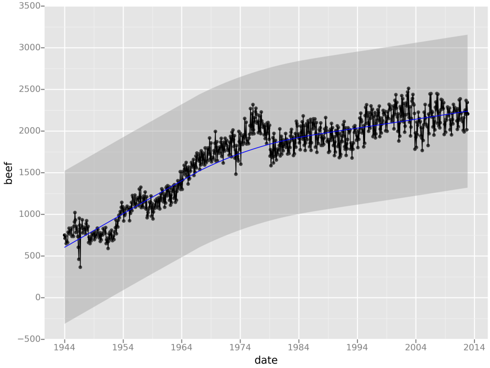
Add a line.

```
p + geom_point() + geom_line()
```



Add a trendline.

```
p + geom_point() + geom_line() +
stat_smooth(color="blue")
```



qplot

qplot

- `qplot` is the basic plotting function in the ggplot package, designed for quick inspections of the data.
- The functionality is not as expansive as with using “ggplot”.

geoms

Geometric objects (geoms) are the visual representations of (subsets of) observations.

- **Univariate** - single numeric variable
- **Bivariate** - two numeric variable
- **Multivariate** - Multiple variables

geoms for ggplot

These are the geoms currently available for ggplot in python.

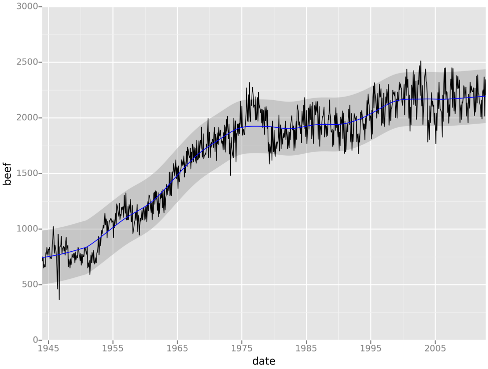
geom_abline	geom_histogram	geom_pointrange
geom_area	geom_hline	geom_rect
geom_bar	geom_jitter	geom_smooth
geom_blank	geom_line	geom_step
geom_boxplot	geom_linerange	geom_text
geom_density	geom_path	geom_tile
geom_dotplot	geom_point	geom_vline

stats

- *Stats* apply statistical transformations that are used to summarise the data, and allows a huge range of possibilities.
- `Stat_smooth` is a nice stat to illustrate the principles, which fits a line and a shaded band to indicate some specified level of uncertainty, as shown in the following example which fits a linear regression line.

stats

```
ggplot(aes(x=’date’, y=’beef’), data=meat) +
geom_line() +\
stat_smooth(colour=’blue’, span=0.2)
```


3000
2500
2000
1500
1000
500
0
beef
1945
1955
1965
1975
1985
1995
2005
date

stats for ggplot

These are the stats currently available for ggplot in python.

stat_abline	stat_hline
stat_bar	stat_identity
stat_bin	stat_smooth
stat_bin2d	stat_summary
stat_density	stat_vline
stat_function	

Aesthetics

Aesthetics

- Aesthetics describe how your data will relate to your plots.
- Some common aesthetics are: **x**, **y**, and **color**.
- Aesthetics are specific to the type of plot (or layer) you're adding to your visual.
- For example, a scatterplot (`geom_point`) and a line (`geom_line`) will share x and y, but only a line chart has a linetype aesthetic.

Aesthetics

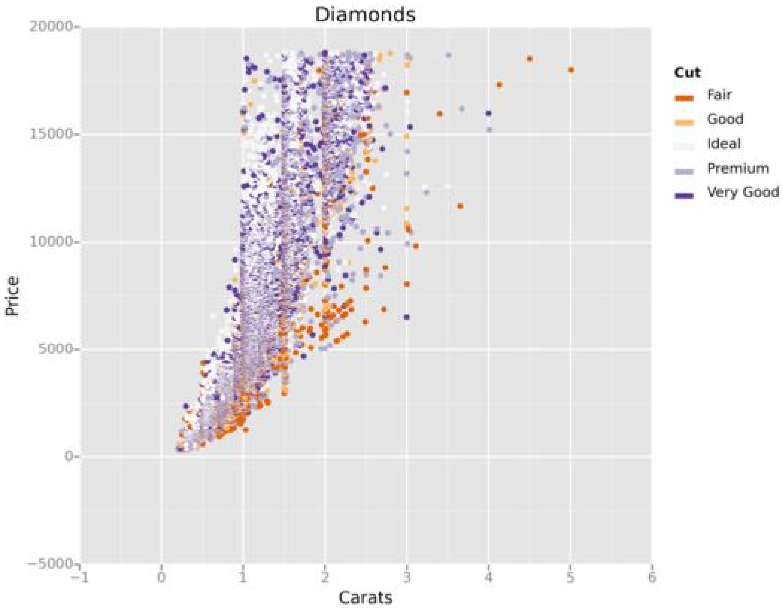
Aesthetics correspond with the “X” and “Y” variables. The first line of the code below indicates which variable is which.

```
ggplot(aes(x="date",y="beef"), meat) +\
geom_line() +\
stat_smooth(colour="blue", span=0.2)
```

Aesthetics

The first aesthetic variable is for `color`. This can be used to depict subcategories in the data.

```
ggplot(diamonds, aes(x="carat",
  y="price", color="cut")) +\
  geom_point() +\ ....
```


Diamonds
Price
20000
15000
10000
5000
0
−5000
−1
0
1
2
3
4
5
6
Carats
Cut
Fair
Good
Ideal
Premium
Very Good

Faceting

The faceting approach supported by ggplot partitions a plot into a matrix of panels. Each panel shows a different subset of the data. There are two faceting approaches:

- `facet_wrap("cell")` - univariate: create a 1-d strip of panels, based on one factor, and wrap the strip into a 2-d matrix
- `facet_grid("row","col")` - (usually) bivariate: create a 2-d matrix of panels, based on two factors

Faceting

Suppose `cyl` and `drv` are two categorical variables in a data frame

```
qplot(.....) + facet_grid("cyl","drv")
```

Faceting

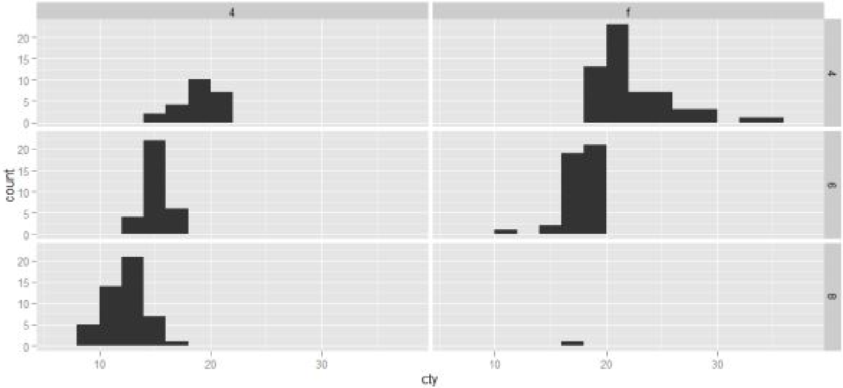


Figure: Grid Faceting

Faceting

Facet Wrap

- An alternative to grid facetting is a wrapped ribbon of plots
- `facet_wrap` generates a long ribbon of plots, and wraps it into 2d.

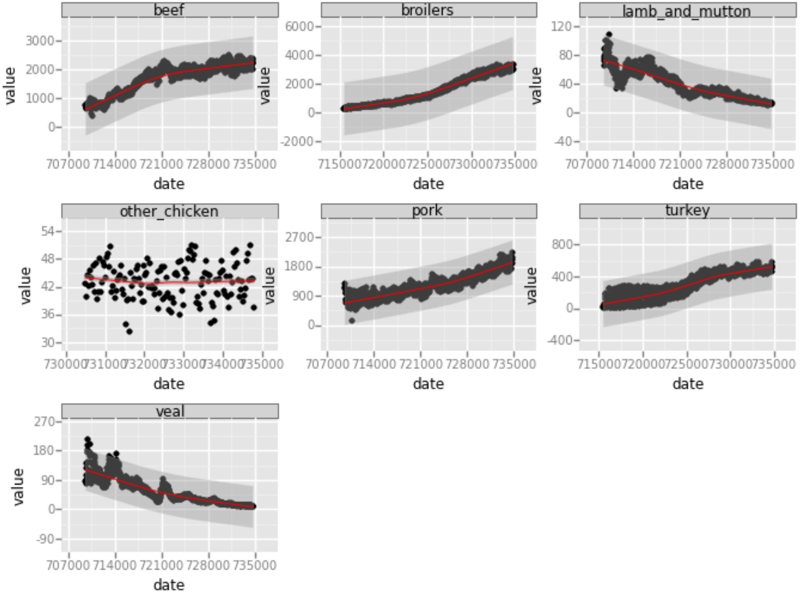
Faceting

Facetting - Examples

```
p = ggplot(aes(x="date", y="value"), data=meat2)

# Scatterplot with Smoother
p + geom_point() + \
stat_smooth(colour="red") + \
facet_wrap("variable")
```

Faceting


beef
broilers
lamb_and_mutton
other_chicken
pork
turkey
veal
value
date
3000
2000
1000
0
707000 714000 721000 728000 735000
6000
4000
2000
0
-2000
715000720000725000730000735000
120
80
40
0
-40
54
48
42
36
30
730000731000732000733000734000735000
2700
1800
900
0
800
400
0
-400
270
180
90
0
-90

Fi

Faceting

```
p = ggplot(aes(x="price"), data=diamonds)
p + geom_histogram() + facet_wrap("cut")
```

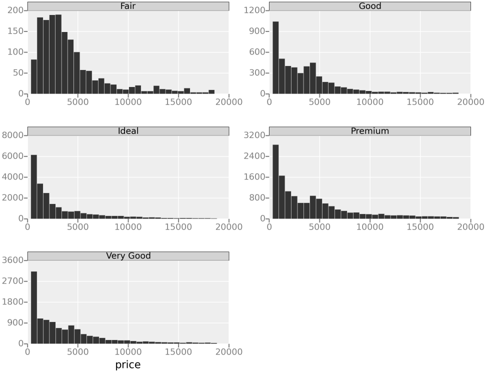

Fair
200
150
100
50
0
0
5000
10000
15000
20000
Good
1200
900
600
300
0
0
5000
10000
15000
20000
Ideal
8000
6000
4000
2000
0
0
5000
10000
15000
20000
Premium
3200
2400
1600
800
0
0
5000
10000
15000
20000
Very Good
3600
2700
1800
900
0
0
5000
10000
15000
20000
price

Figure:

Faceting

```
p = ggplot(diamonds, aes(x='price'))
p + geom_density() + \
facet_grid("cut", "clarity")
```

Faceting

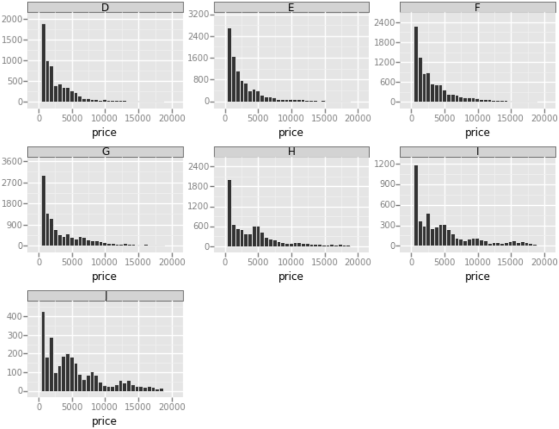
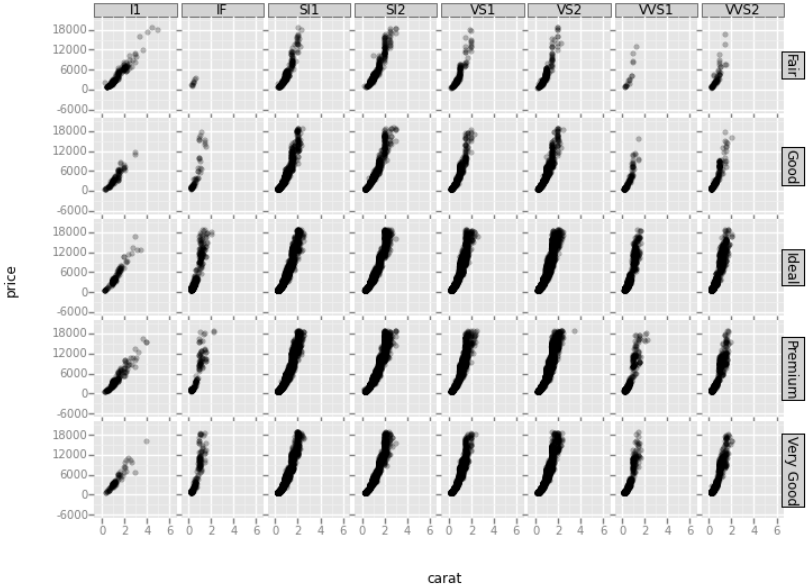


Figure:

Faceting

```
p = ggplot(diamonds, aes(x='carat', y='price'))
p + geom_point(alpha=0.25) + \
facet_grid("cut", "clarity")
```

Faceting


I1
IF
SI1
SI2
VS1
VS2
VVS1
VVS2
Fair
Good
Ideal
Premium
Very Good
price
18000
12000
6000
0
-6000
carat
0
2
4
6

```
p  = ggplot(iris, aes("Sepal.Length", "Sepal.Width"))
p + geom_boxplot(fill = c("lightblue")) + facet_wrap("Spe
```

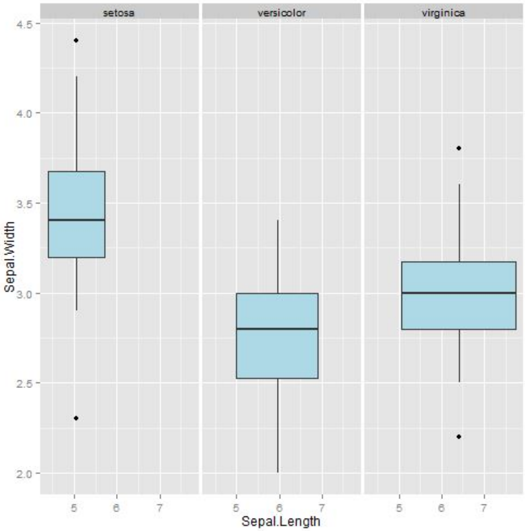

setosa
versicolor
virginica
Sepal.Width
4.5
4.0
3.5
3.0
2.5
2.0
5
6
7
Sepal.Length

Figure:

Scales and Themes

- ggplot2 provides a large number of scale functions to control aspects of a graphic including axes and legends
- theme functions allow us to control the overall style of the graphic

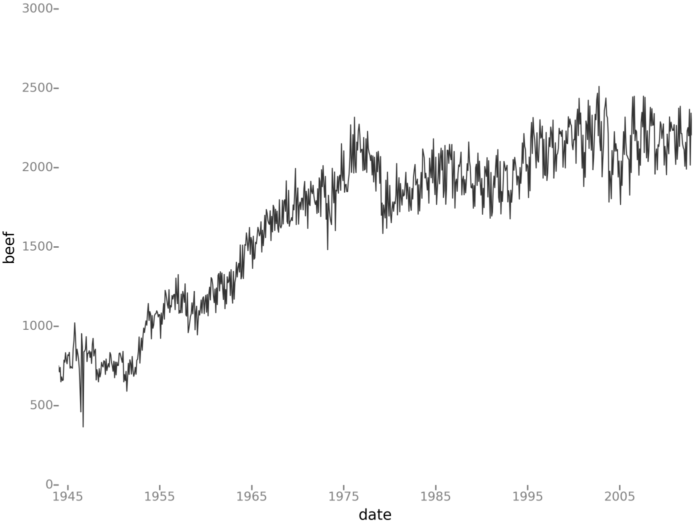
Scales

- A scale determines how an attribute of the data is mapped into an aesthetic property of a `geom` (e.g., the geom's position along the x axis, or a geom's fill color in a color space).
- The colours and shapes used in the chart can be manually adjusted if you dont like the defaults.

Themes

```
ggplot(aes(x="date", y="beef"),meat) +\
geom_line() +\
theme_bw()
```

Themes



3000
2500
2000
1500
1000
500
0
beef
1945
1955
1965
1975
1985
1995
2005
date

Themes

Try out the following themes

- `theme_538`
- `theme_bw`
- `theme_gray`
- `theme_matplotlib`
- `theme_seaborn`
- `theme_xkcd`