



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 hat

[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 professional 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.

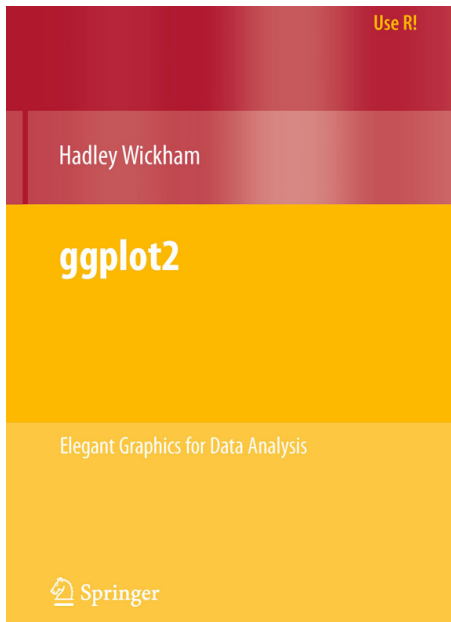
Documentation

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

website: www.had.co.nz

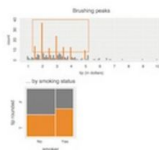


Hadley Wickham (Chief Data Scientist, RStudio)



Visiphilia

The love of plotting data



Monash University | Department of Econometrics and Statistics

[HOME](#)

[ISU STATISTICAL GRAPHICS WORKING GROUP](#)

Posts

- December 26, 2015 [Better cricket plots](#)
- November 6, 2015 [Statistical Sciences, Cornell University](#)
- October 26, 2015 [Center for Statistics and Applications in Forensic Evidence](#)
- October 13, 2015 [Data Science for Managers, 2015, Monash Conference Center, Melbourne](#)
- May 17, 2015 [Graduates in Statistical Graphics Research at ISU 2015](#)

Authors

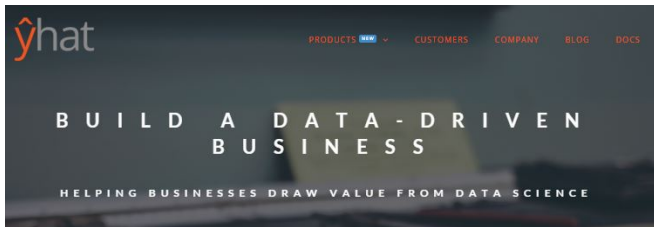
Di Cook

Heike Hofmann

Related Blogs

Hyndsight

Statistics, R, Graphics and Fun
statschat.org.nz



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
```

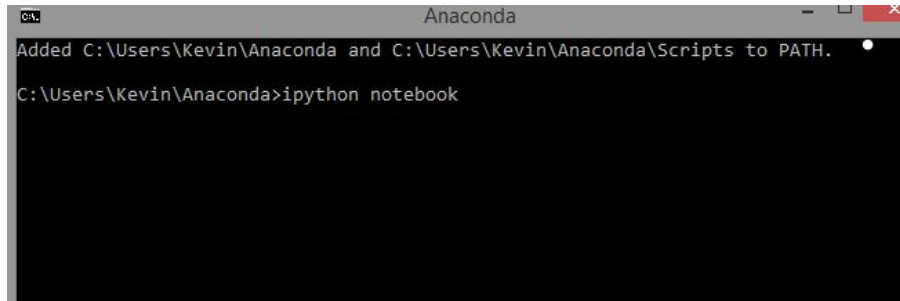
N.B. Not loaded automatically with Anaconda, unlike pandas and numpy

Upgrading Version

```
pip install --upgrade pip ggplot
```

```
C:\Users\Kevin\Anaconda>pip install --upgrade pip ggplot
Collecting pip
  Downloading pip-8.0.2-py2.py3-none-any.whl (1.2MB)
    100% |#####| 1.2MB 166kB/s
Requirement already up-to-date: ggplot in c:\users\kevin\anaconda\lib\site-packages
Collecting scipy (from ggplot)
  Downloading scipy-0.17.0.tar.gz (12.4MB)
    99% |##### | 12.4MB 487kB/s eta 0:00:01
```

2. Start the Jupyter Notebook



A screenshot of a Windows command prompt window titled "Anaconda". The window has a standard Windows title bar with minimize, maximize, and close buttons. The text inside the window shows the command prompt path "C:\Users\Kevin\Anaconda" and the command "ipython notebook" being entered. Above the command, a message states: "Added C:\Users\Kevin\Anaconda and C:\Users\Kevin\Anaconda\Scripts to PATH." The command prompt is currently at the end of the command line, ready for execution.

```
C:\Users\Kevin\Anaconda>ipython notebook
```

3. Getting Set Up on Jupyter Notebook

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

```
In [6]: %matplotlib inline
```



[overview](#) // [get pandas](#) // [documentation](#) // [community](#)

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 language.

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.

```
In [3]: import os  
import pandas as pd
```

```
In [5]: os.getcwd()
```

```
Out[5]: 'C:\\Users\\Kevin\\Dropbox\\Public\\PyData\\NoteBooks'
```

```
In [6]: ansc = pd.read_csv('anscombe.csv')
```



```
In [15]: ansc = pd.read_csv('anscombe.csv', index_col=[0])
```

```
In [16]: ansc.head()
```

```
Out[16]:
```

	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
4	9	9	9	8	8.81	8.77	7.11	8.84
5	11	11	11	8	8.33	9.26	7.81	8.47

```
In [17]: tmp=pd.melt(ansc)
```

```
# 1. Load Libraries
```

```
import os
```

```
import pandas as pd
```

```
# 2. Get Current Working Directory
```

```
os.getcwd()
```

```
# 3. Read in a CSV Data File
```

```
ansc = pd.read_csv('anscombe.csv',index_col=[0])
```

```
# 4. Print the top of that file to inspect it
```

```
ansc.head()
```

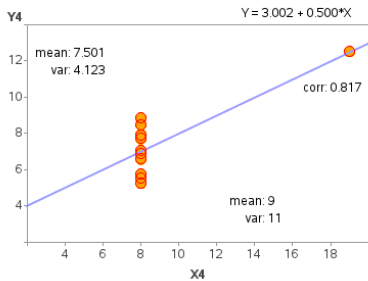
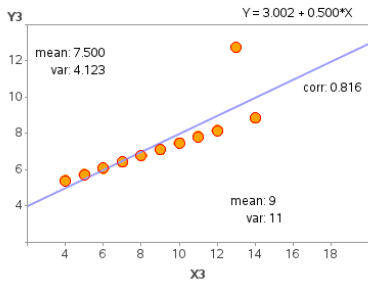
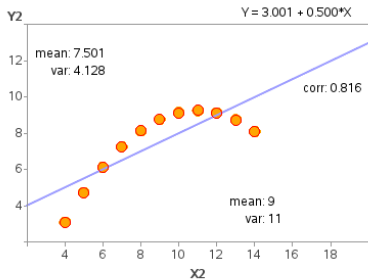
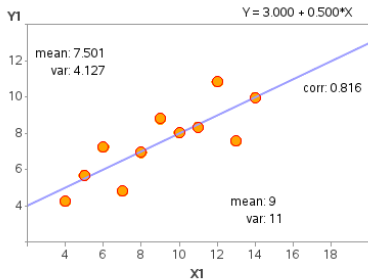
Different Layour (better for out purposes)

```
In [30]: aq = pd.read_csv("anscombe2.csv",header=False)
```

```
In [31]: aq.head()
```

```
Out[31]:
```

	group	x	y
0	1	10	8.04
1	1	8	6.95
2	1	13	7.58
3	1	9	8.81
4	1	11	8.33



(Source: Wikipedia)

Try this out!

```
ggplot(aes(x="x", y="y"), data=aq)  
  + facet_wrap('group', scales='fixed')  
  + geom_point()  
  + stat_smooth(method='lm')
```

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.

content...

```
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.

DOWNLOAD ANACONDA NOW!

Jump to: [Windows](#) | [OS X](#) | [Linux](#)

Get Superpowers with Anaconda

Anaconda is a completely free Python distribution (including for commercial use and redistribution). It includes more than 400 of the most popular [Python packages](#) for science, math, engineering, and data analysis. See [the packages included with Anaconda](#) and the [Anaconda changelog](#).

Basic Premise

- ▶ Making plots is a very repetitive: 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

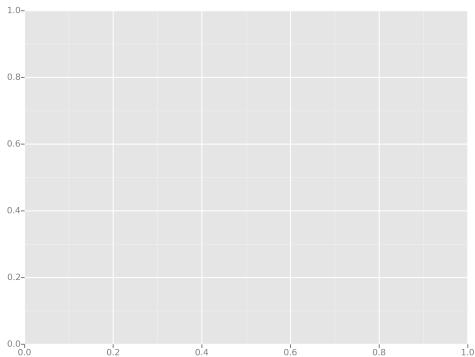
- ▶ **ggplot** lets you combine or add different types of visualization components (or layers) together.
- ▶ The command `ggplot` does not actually create any plot, rather it prepares a “blank canvas” for further plotting
- ▶ We will introduce *geoms* shortly

(Some useful preparation)

```
import pandas as pd  
  
meat2 = pd.melt(meat, id_vars=["date"])
```

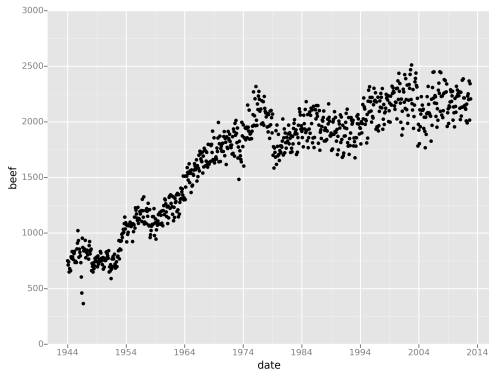
Start with a blank canvas.

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



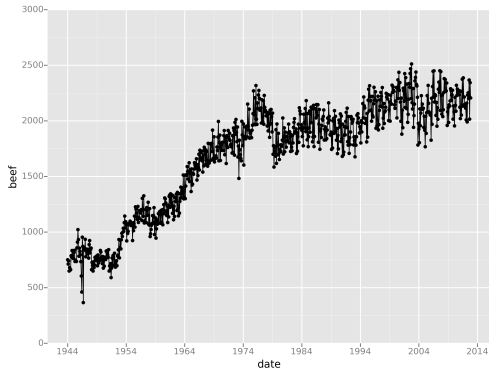
Add some points.

```
p + geom_point()
```



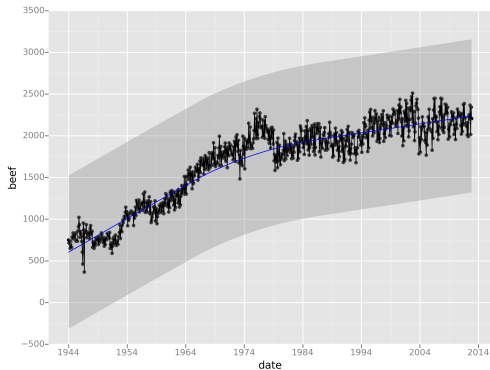
Add a line.

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



Add a trendline.

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



More!

```
p + ggtitle("Plot Title")  
  + xlab(" X Axis Label")  
  + ylab(" Y Axis Label")
```

Rule of Thumb

- ▶ The aes argument is for aesthetics
- ▶ Essentially it identifies which variables are being used, and in what order.
- ▶ The first two variables are the "X" and "Y" variable.
- ▶ Any more variables after that are typically grouping variables.

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()”`.
- ▶ For the most part, the same code works for both.

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

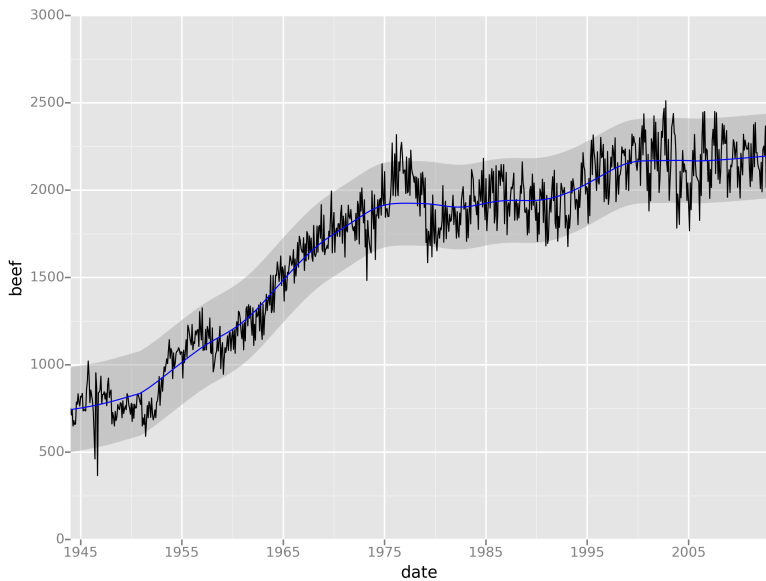
These are the geoms currently available for ggplot in python.

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.

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



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 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

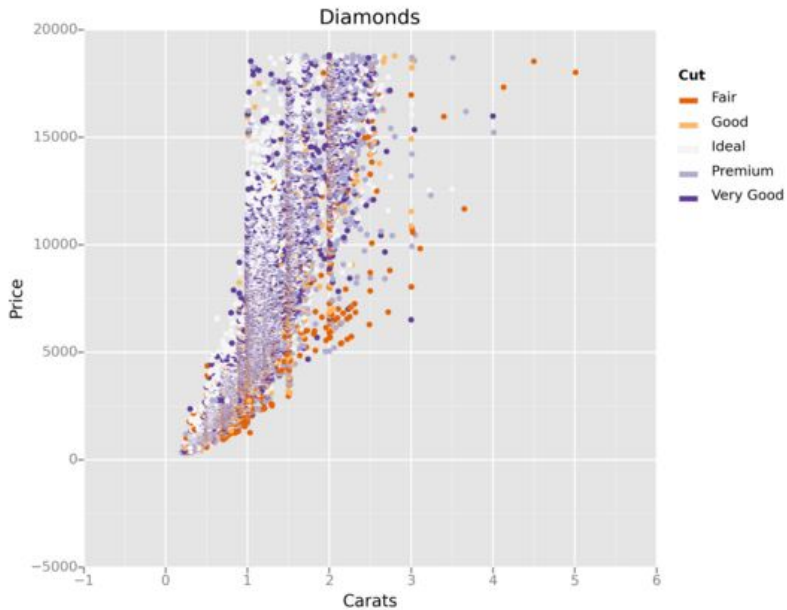
Aesthetics correspond with the “X” and “Y” variables. The first line of the code below indicates which variable is which.

```
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() +\  ....
```



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

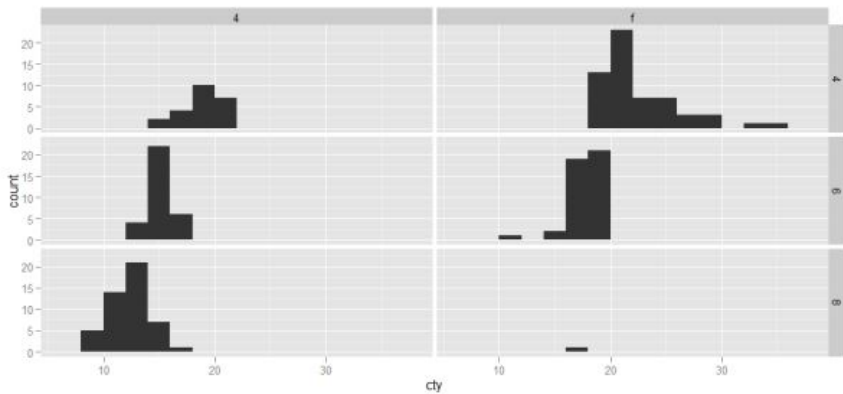


Figure: Grid Faceting

Facet Wrap

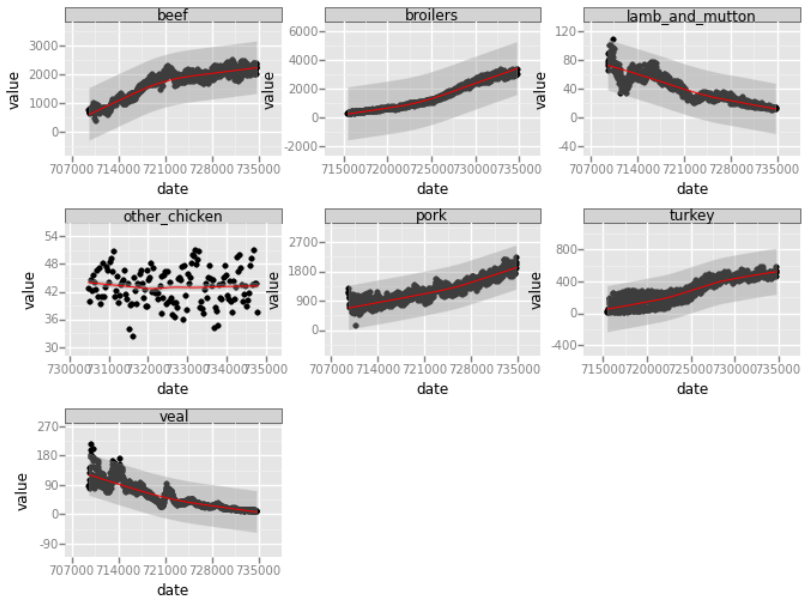
- ▶ An alternative to grid facetting is a wrapped ribbon of plots
- ▶ `facet_wrap` generates a long ribbon of plots, and wraps it into 2d.

Facetting - Examples

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

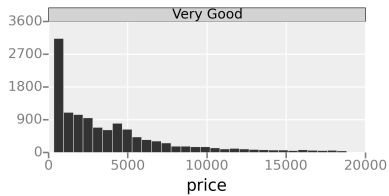
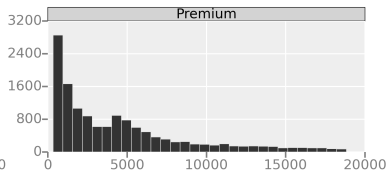
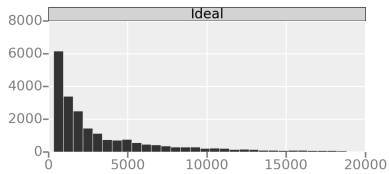
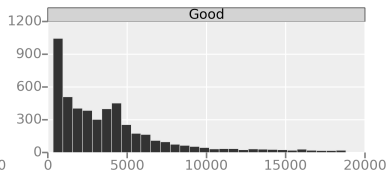
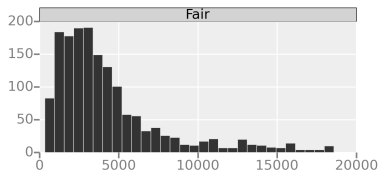
# Scatterplot with Smoother
p + geom_point() + \
  stat_smooth(colour="red") + \
  facet_wrap("variable")
```


Faceting



Faceting

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



Faceting

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

Faceting

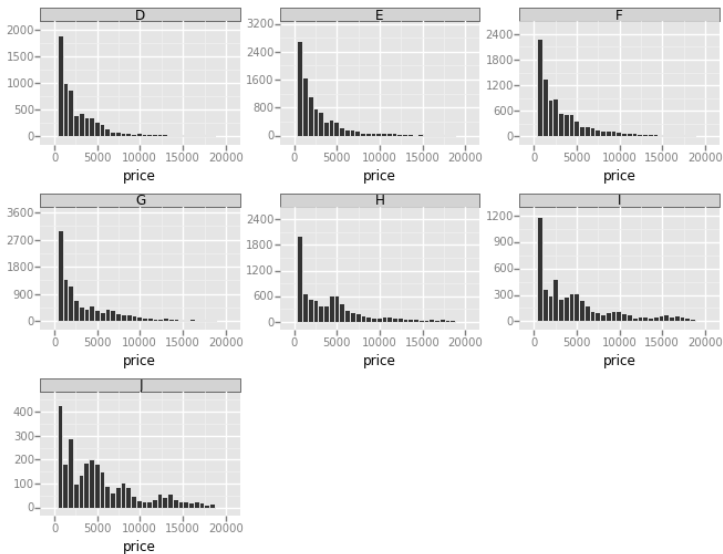
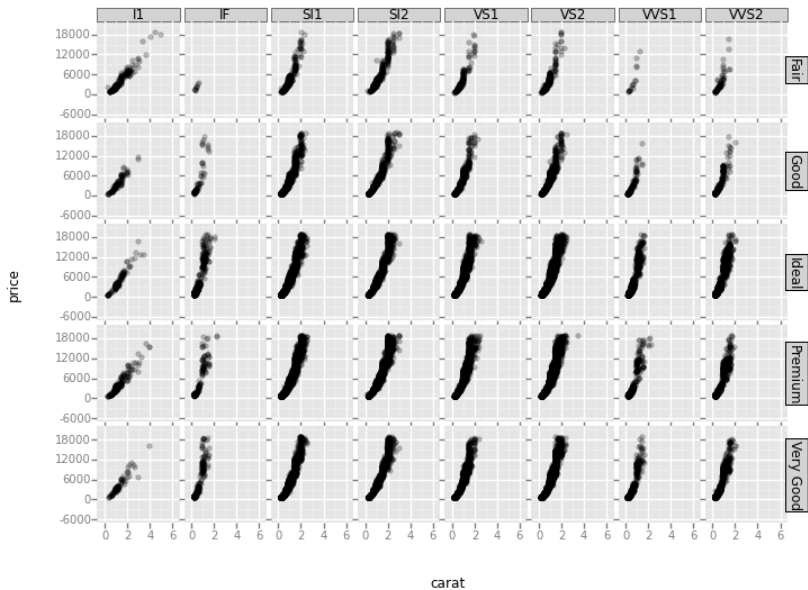


Figure:

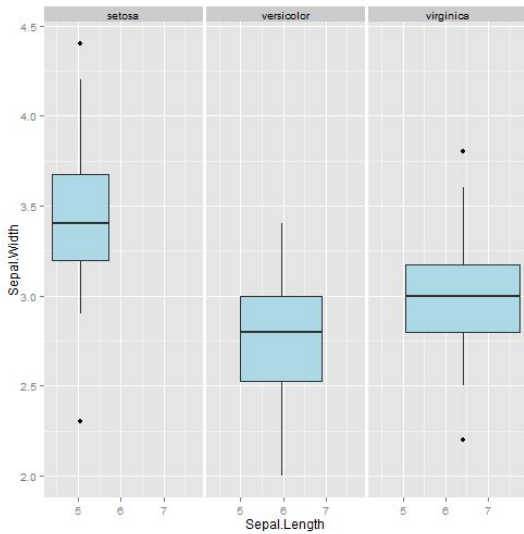
Faceting

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

Faceting



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

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

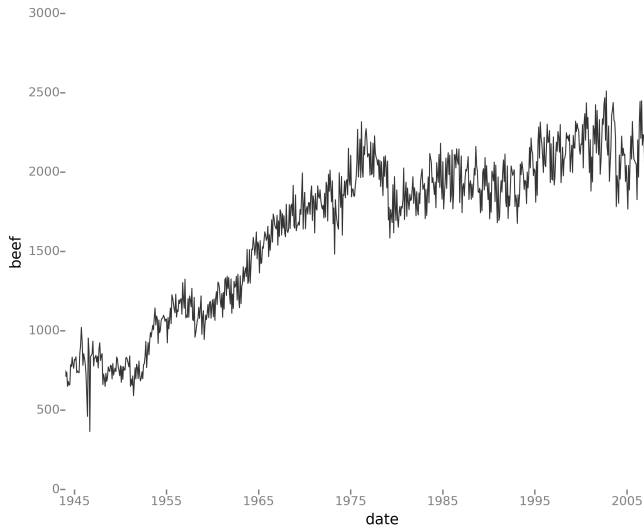
Scales

- ▶ 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 don't like the defaults.

Themes

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

Themes



Themes

Try out the following themes

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