Data Visualisation: Telling Visual Stories From Data

Bevan Koopman

03/04/2017

Anscombe's quartet

```
anscombe_m <- read.csv("anscombe_m.csv")
summary(anscombe_m)
```

```
##  Dataset       x            y
##  A:11    Min.   : 4   Min.   : 3.100
##  B:11    1st Qu.: 7   1st Qu.: 6.117
##  C:11    Median : 8   Median : 7.520
##  D:11    Mean   : 9   Mean   : 7.501
##          3rd Qu.:11   3rd Qu.: 8.748
##          Max.   :19   Max.   :12.740
```

Checks the mean, variance for each dataset

```
aggregate(y ~ Dataset, anscombe_m, mean)
```

```
##   Dataset        y
## 1       A 7.500909
## 2       B 7.500909
## 3       C 7.500000
## 4       D 7.500909
```

```
aggregate(x ~ Dataset, anscombe_m, mean)
```

```
##   Dataset x
## 1       A 9
## 2       B 9
## 3       C 9
## 4       D 9
```

```
aggregate(y ~ Dataset, anscombe_m, var)
```

```
##   Dataset        y
## 1       A 4.127269
## 2       B 4.127629
## 3       C 4.122620
## 4       D 4.123249
```

```
aggregate(x ~ Dataset, anscombe_m, var)
```

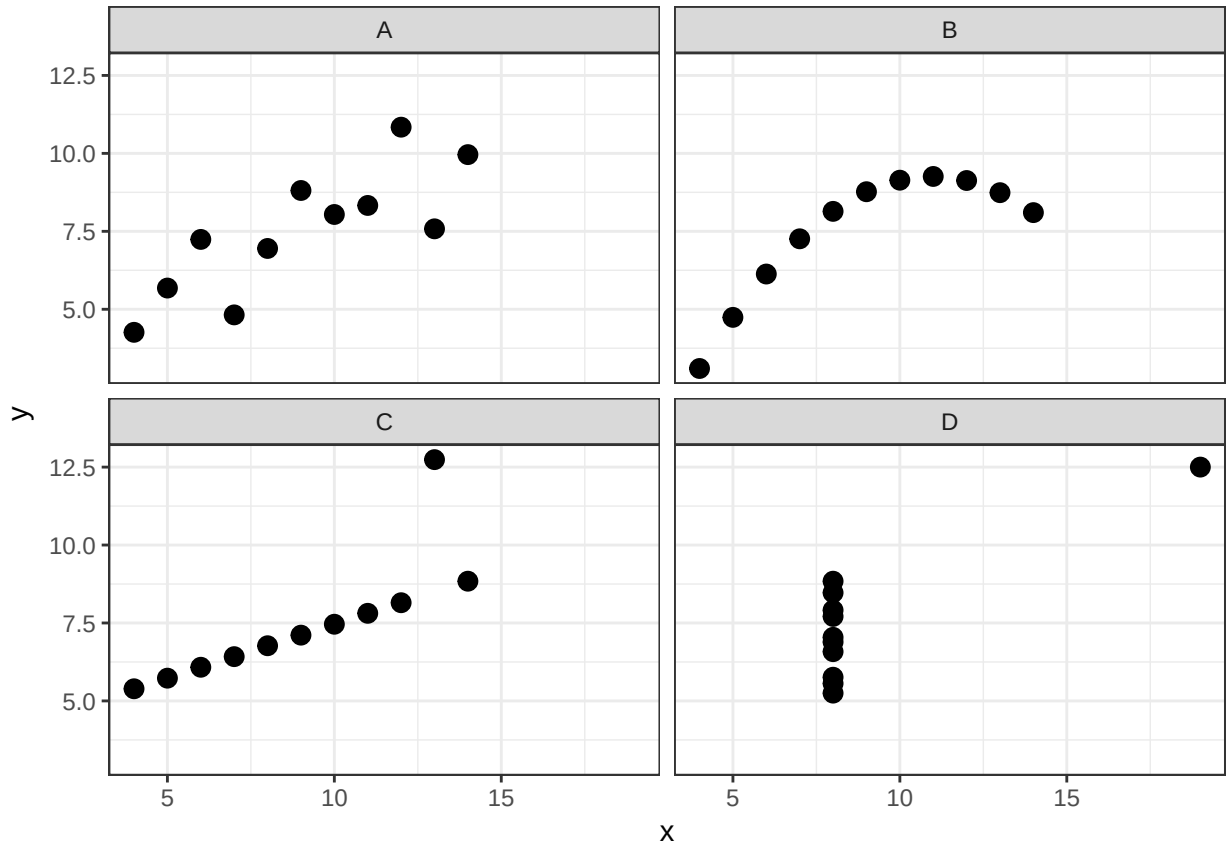
```
##   Dataset  x
## 1       A 11
## 2       B 11
## 3       C 11
## 4       D 11
```

```
for(d in c("A", "B", "C", "D")) {
  print(cor(anscombe_m[anscombe_m$Dataset==d,]$x, anscombe_m[anscombe_m$Dataset==d,]$y))
}
```

```
## [1] 0.8164205
## [1] 0.8162365
## [1] 0.8162867
## [1] 0.8165214
```

Now plot the four different data sets

```
ggplot(anscombe_m, aes(x=x, y=y)) + geom_point(size=3)  + facet_wrap(~Dataset)
```



Grammar of graphical elements

1. The Data layer

Have a look at the diamonds dataset.

```
summary(diamonds)
```

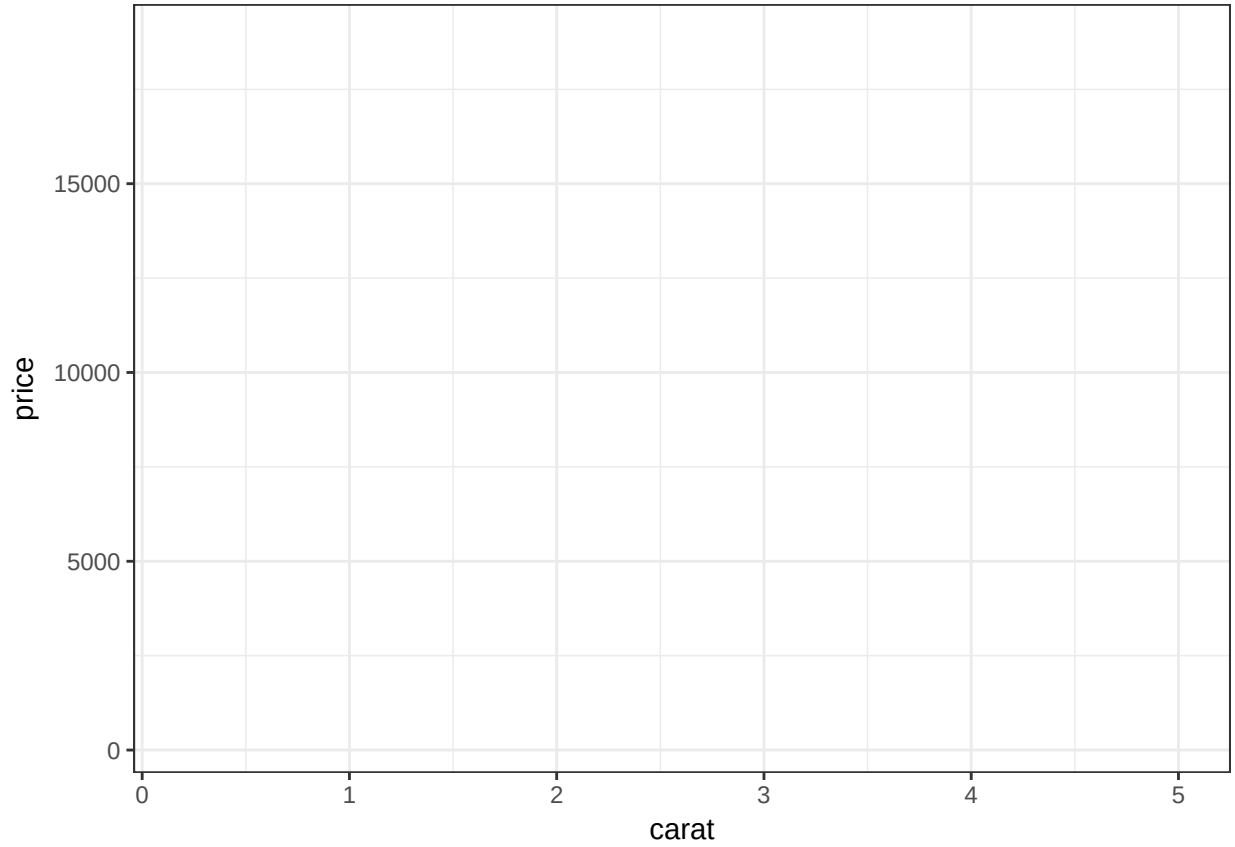
```
##      carat               cut        color        clarity
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258
##  Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194
##  Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066
##  Max.   :5.0100                     I: 5422   VVS1    : 3655
##                                     J: 2808   (Other): 2531
##      depth           table           price             x
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
```

```
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
##
##        y                z
##  Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.530
##  Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :58.900   Max.   :31.800
##
```

2. The `Aesthetics` layer

Now we map parts of the data to some graphical elements (x, y, color, size, etc)

```
ggplot(diamonds, aes(x=carat, y=price))
```

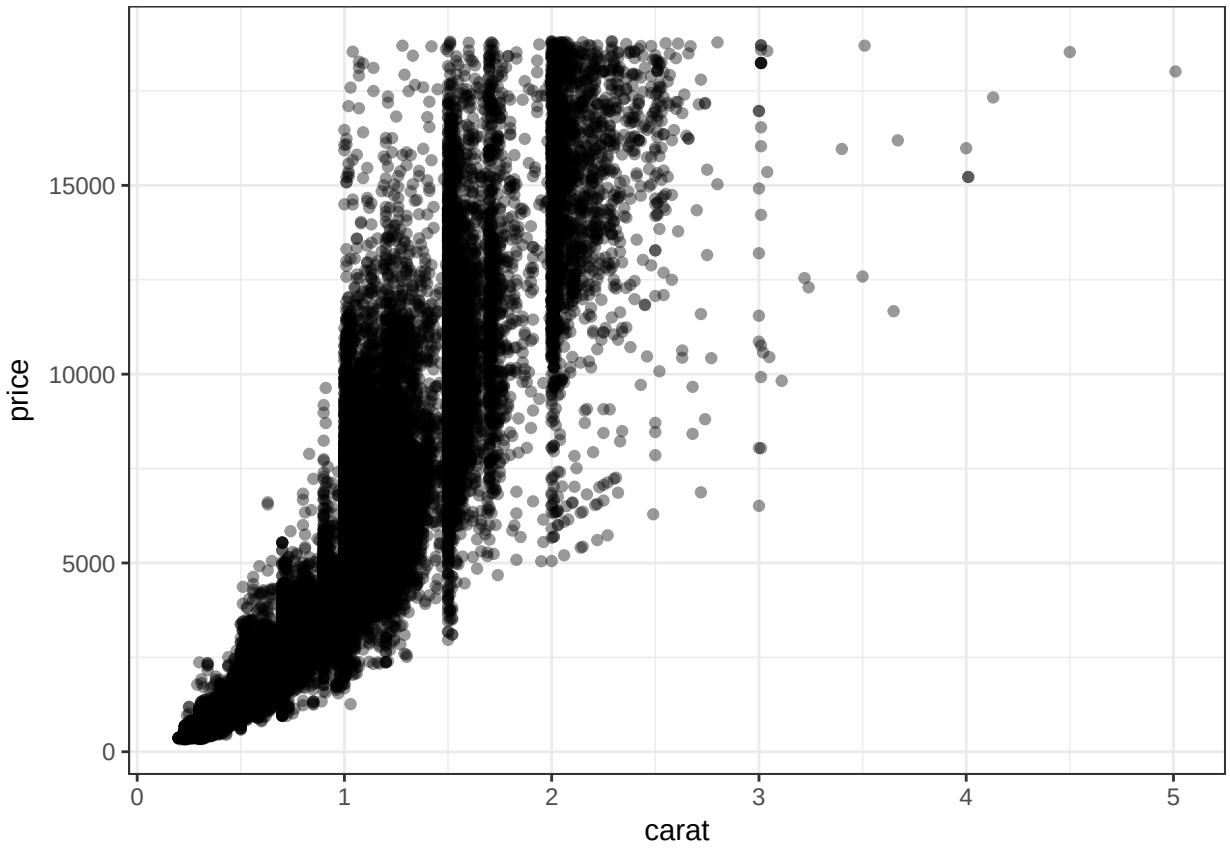


Note, that there is no notion at the moment of the types of plot - that is the next layer!

3. The `Geometries` layer

Now we define a geometry (this case, a scatter geometry):

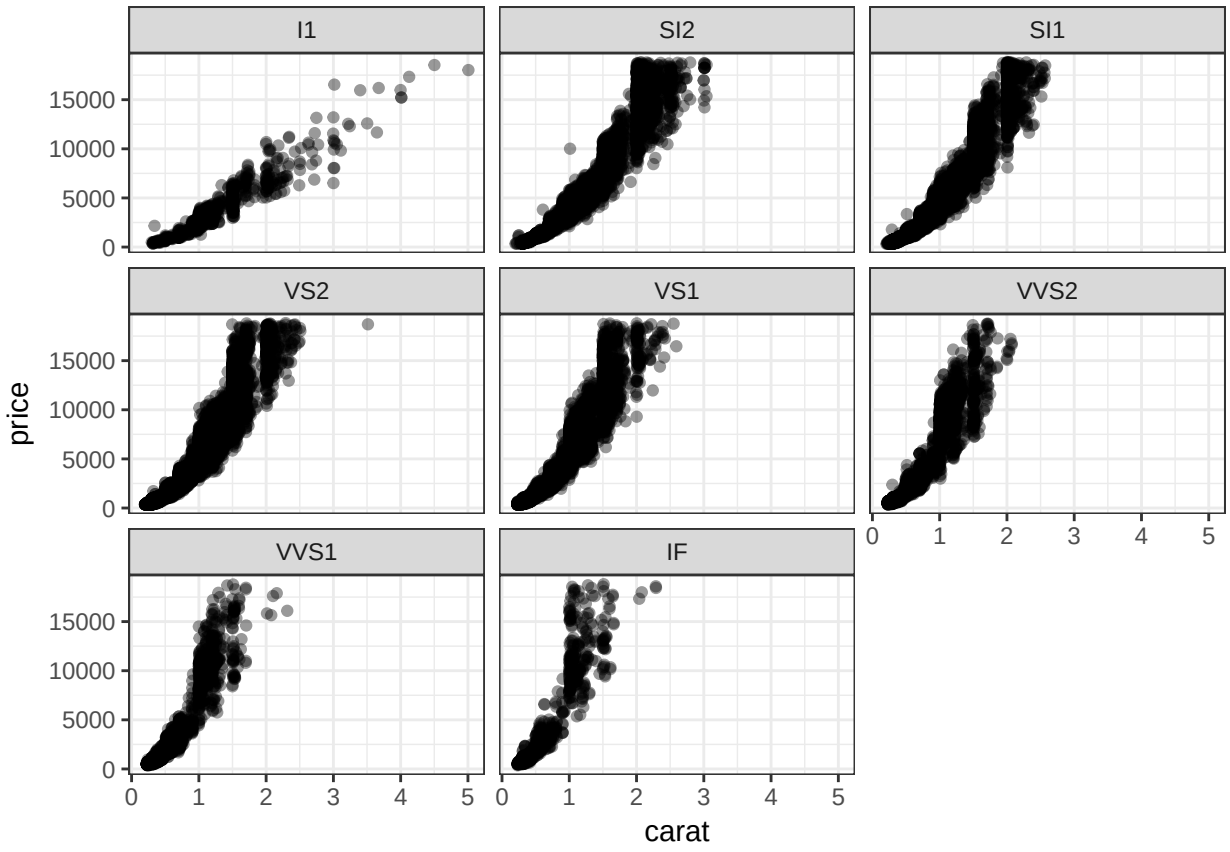
```
ggplot(diamonds, aes(x=carat, y=price)) + geom_point(alpha=.4)
```



4. The `Facet` layer

Now lets split the plot up by `clarity`:

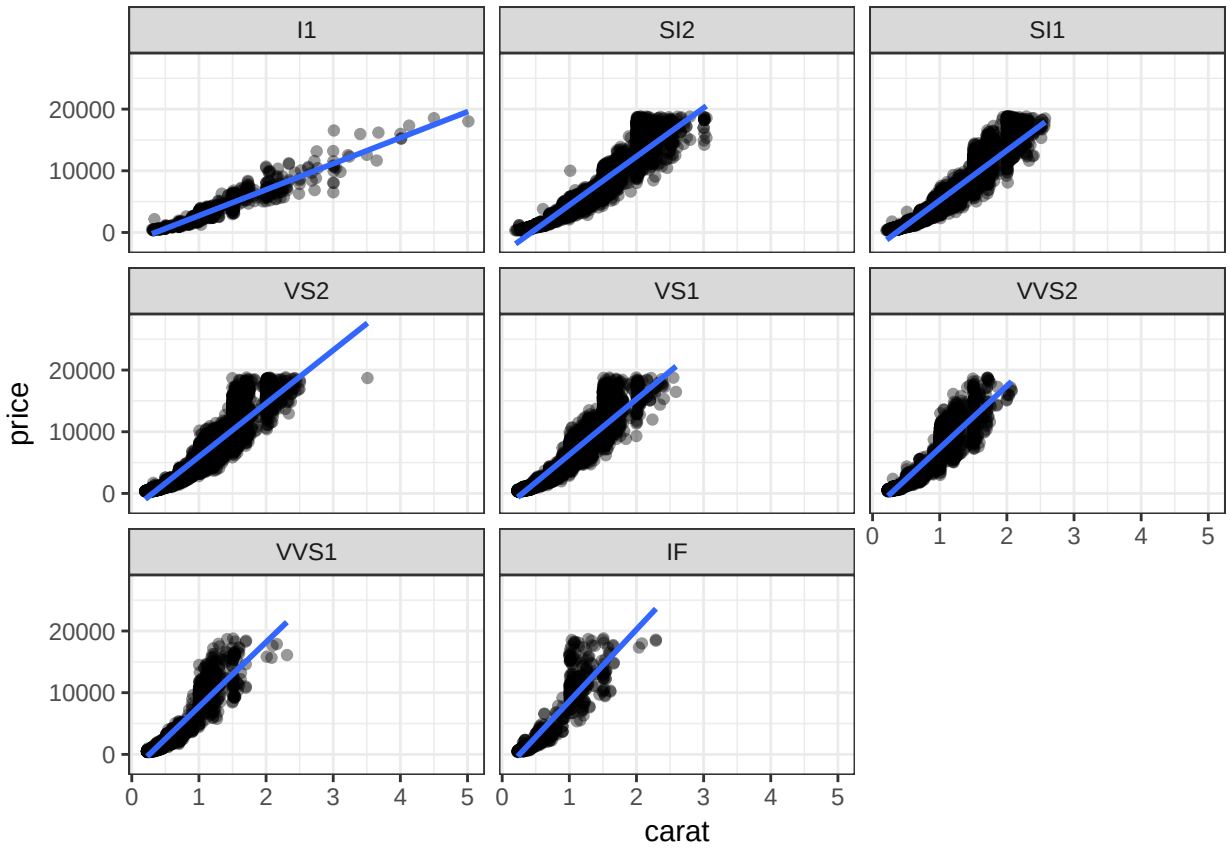
```
ggplot(diamonds, aes(x=carat, y=price)) + geom_jitter(alpha=.4) + facet_wrap(~clarity)
```



5. The `Statistics` layer

Now let's add an graphical element that aids in understanding, namely a linear model:

```
ggplot(diamonds, aes(x=carat, y=price)) + geom_jitter(alpha=.4) + facet_wrap(~clarity) + geom_smooth(me
```



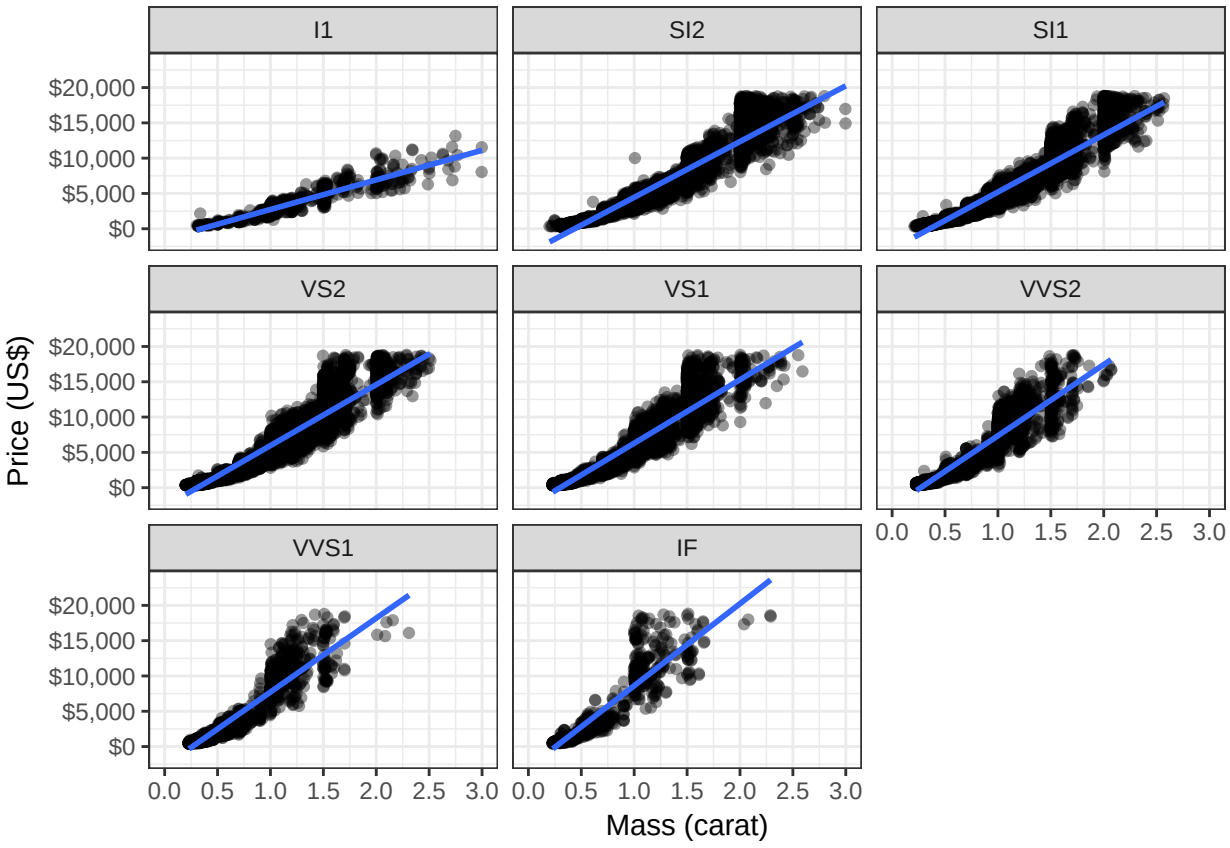
6. The `Coordinates` layer

Now let's ajust the scales and axis levels. Specifically, we'll restrict x-axis carat to be between 0 and 3, report x-axis carat in 0.5 increments, change y-axis price to be in dollar format and provide x and y axis labels.

```
library(scales) # used to print things in dollar ($) format
ggplot(diamonds, aes(x=carat, y=price)) + geom_jitter(alpha=.4) + facet_wrap(~clarity) + geom_smooth(me
```

```
## Warning: Removed 32 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 36 rows containing missing values (geom_point).
```



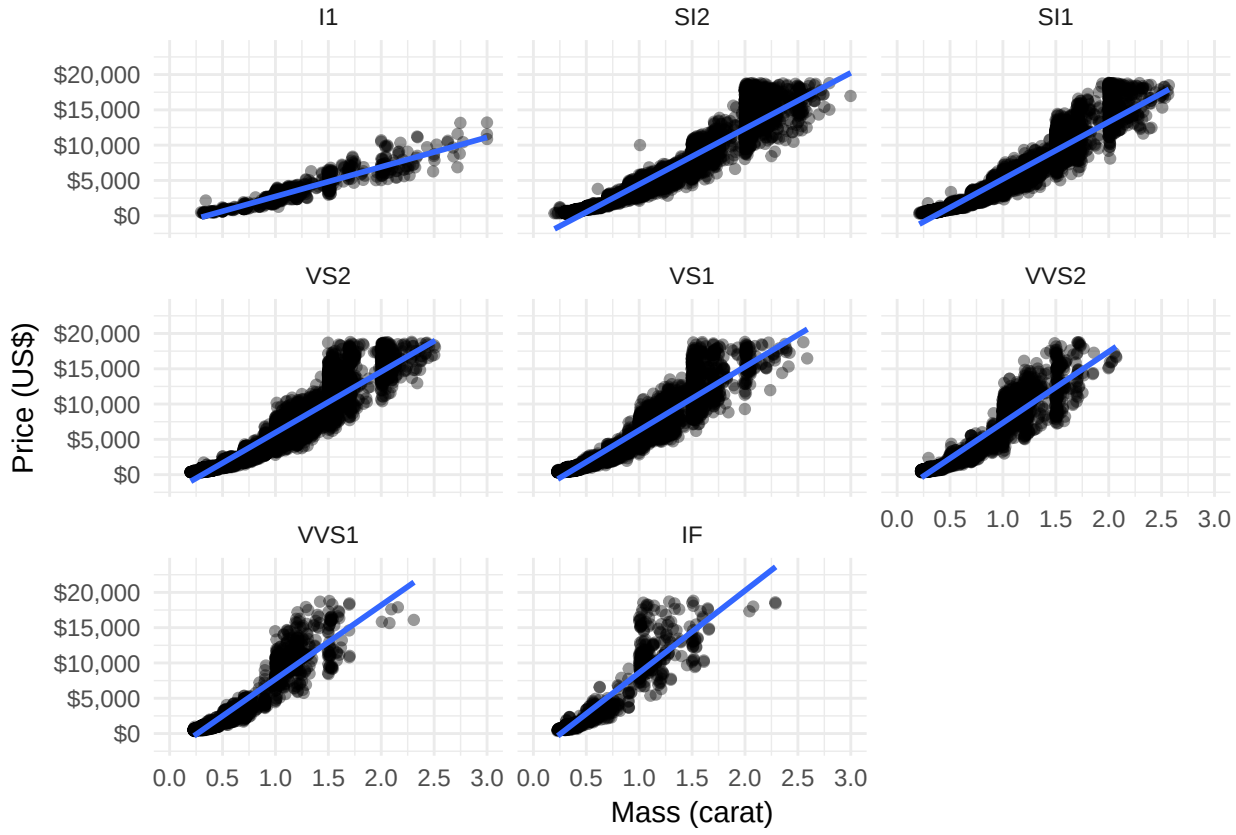
6. The `Theme` layer

Now just change the look and feel:

```
ggplot(diamonds, aes(x=carat, y=price)) + geom_jitter(alpha=.4) + facet_wrap(~clarity) + geom_smooth(me
```

```
## Warning: Removed 32 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 36 rows containing missing values (geom_point).
```



Data that supports Visualisation

Have a look at the `airquality` dataset.

```
summary(airquality)
```

```
##      Ozone           Solar.R          Wind             Temp
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
##  Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
##  NA's   :37       NA's   :7
##      Month            Day
##  Min.   :5.000   Min.   : 1.0
##  1st Qu.:6.000   1st Qu.: 8.0
##  Median :7.000   Median :16.0
##  Mean   :6.993   Mean   :15.8
##  3rd Qu.:8.000   3rd Qu.:23.0
##  Max.   :9.000   Max.   :31.0
##
```

Get the `reshape2` package (not `reshape`).

```
require(reshape2)
```

Look at the top 10 rows of the data:

```
head(airquality)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

Transform to long format.

```
head(melt(airquality))
```

```
## No id variables; using all as measure variables
```

Tranform to long format but keep `Month` and `Day` as columns.

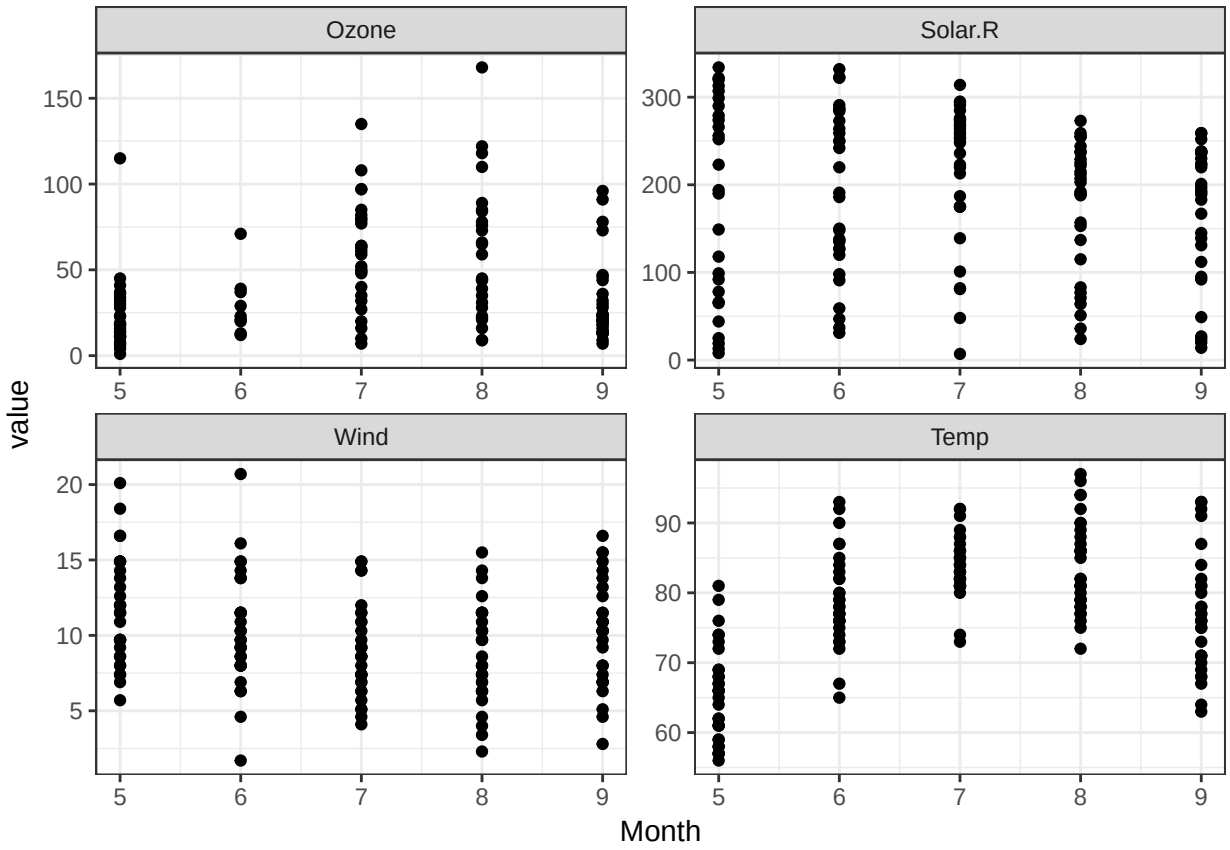
```
long_airquality <- melt(airquality, id.vars=c('Month', 'Day'), variable.name = "Measurement")
head(long_airquality)
```

```
##   Month Day Measurement value
## 1     5   1       Ozone    41
## 2     5   2       Ozone    36
## 3     5   3       Ozone    12
## 4     5   4       Ozone    18
## 5     5   5       Ozone    NA
## 6     5   6       Ozone    28
```

Now the data is in a easier format for plotting.

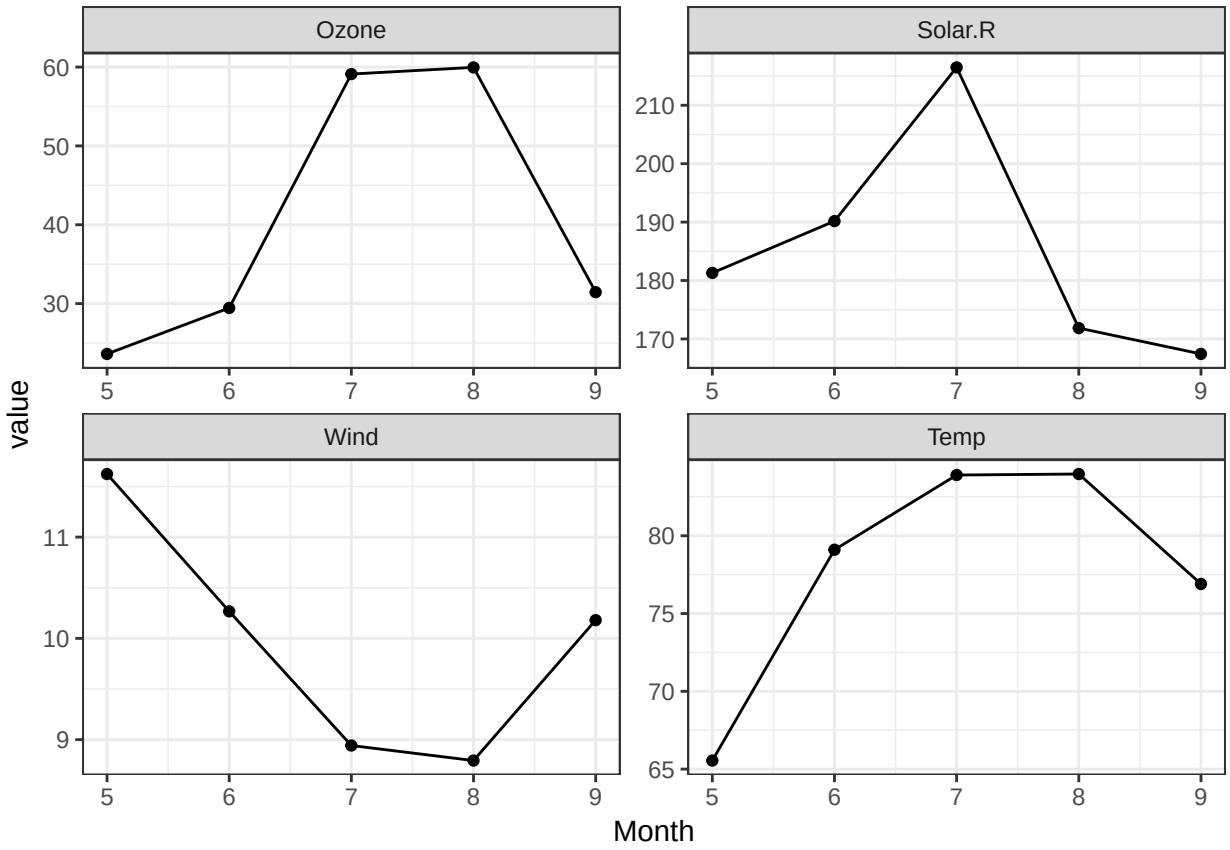
```
ggplot(long_airquality, aes(x=Month, y=value)) + geom_point() + facet_wrap(~Measurement, scales="free")
```

```
## Warning: Removed 44 rows containing missing values (geom_point).
```



Calculate a montly average

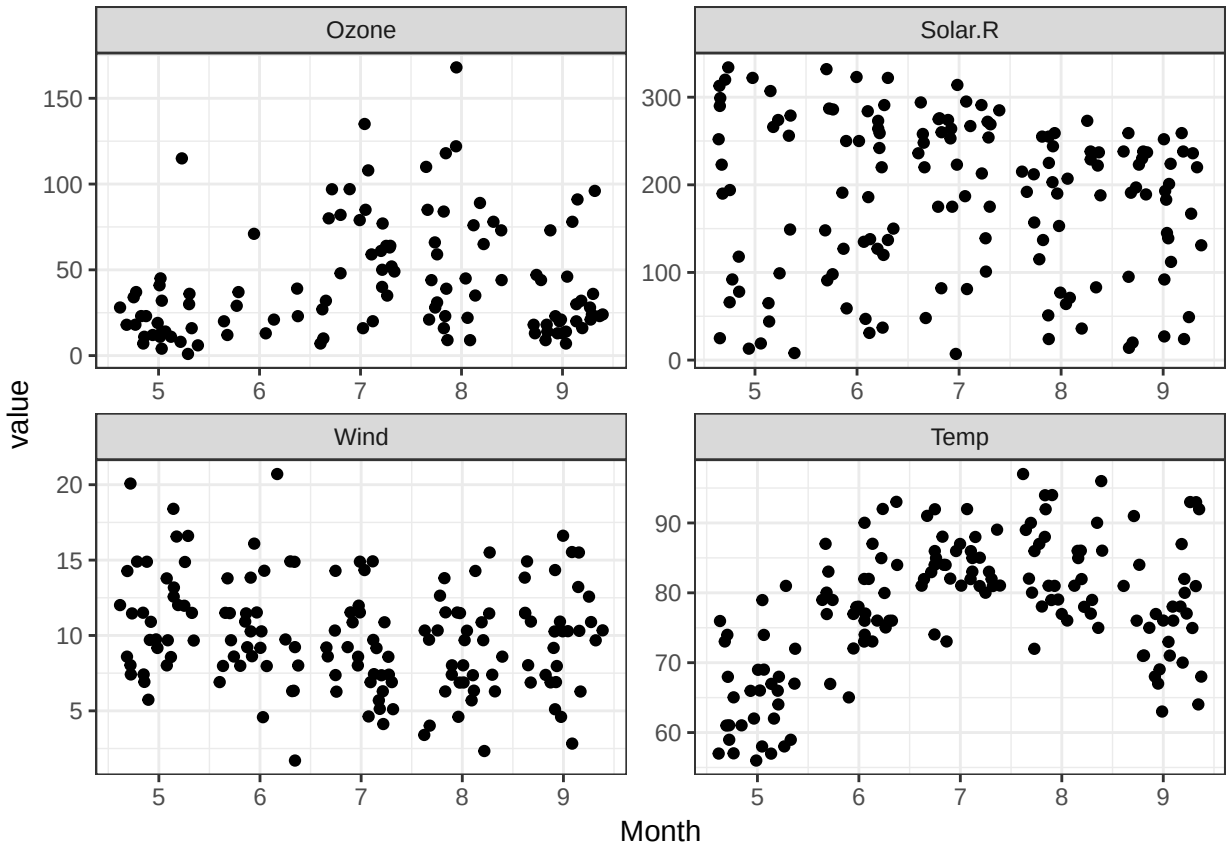
```
agg_long_airquality <- aggregate(value ~ Month + Measurement, long_airquality, mean)
ggplot(agg_long_airquality, aes(x=Month, y=value)) + geom_point() + geom_line() + facet_wrap(~Measureme
```



But rather than calculate the average, why not just use a boxplot:

```
ggplot(long_airquality, aes(x=Month, y=value)) + geom_jitter() + facet_wrap(~Measurement, scales="free")
```

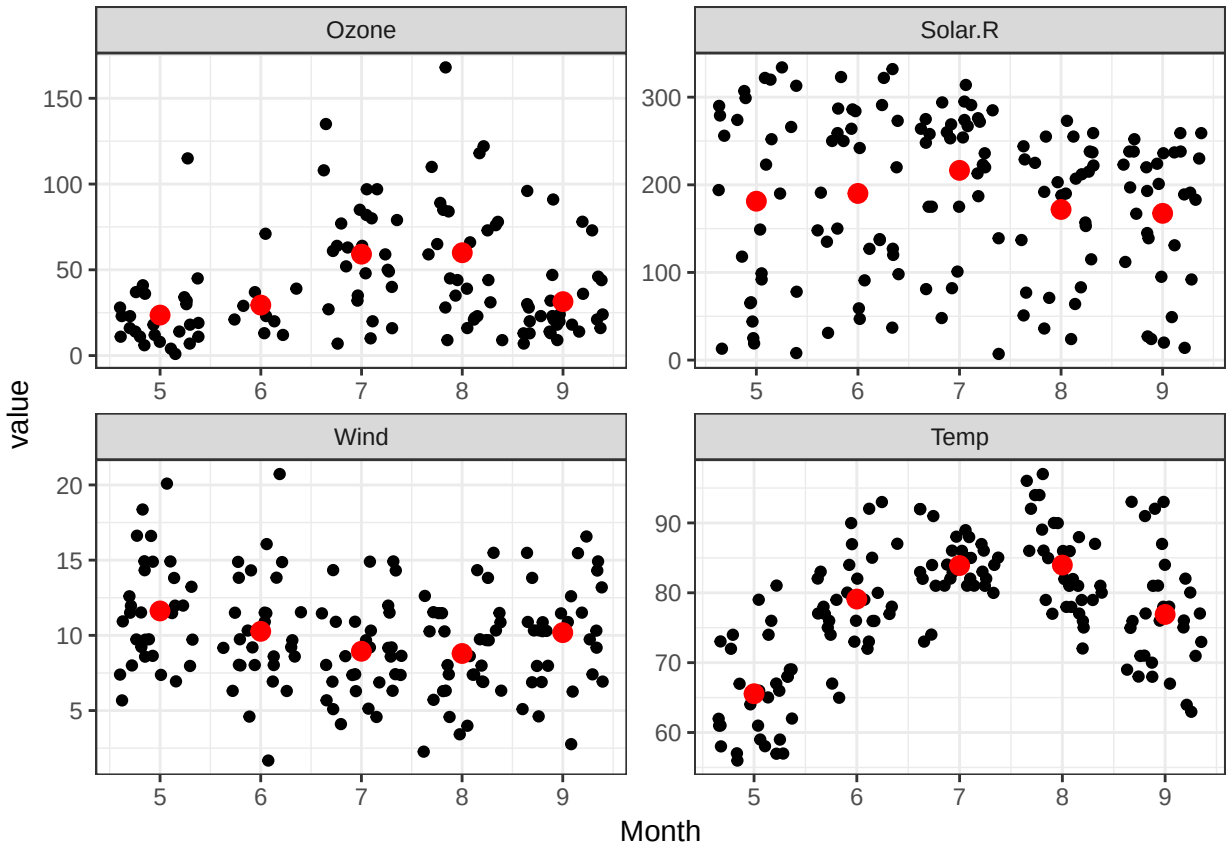
```
## Warning: Removed 44 rows containing missing values (geom_point).
```




```
ggplot(long_airquality, aes(x=Month, y=value)) + geom_jitter() + facet_wrap(~Measurement, scales="free")
```

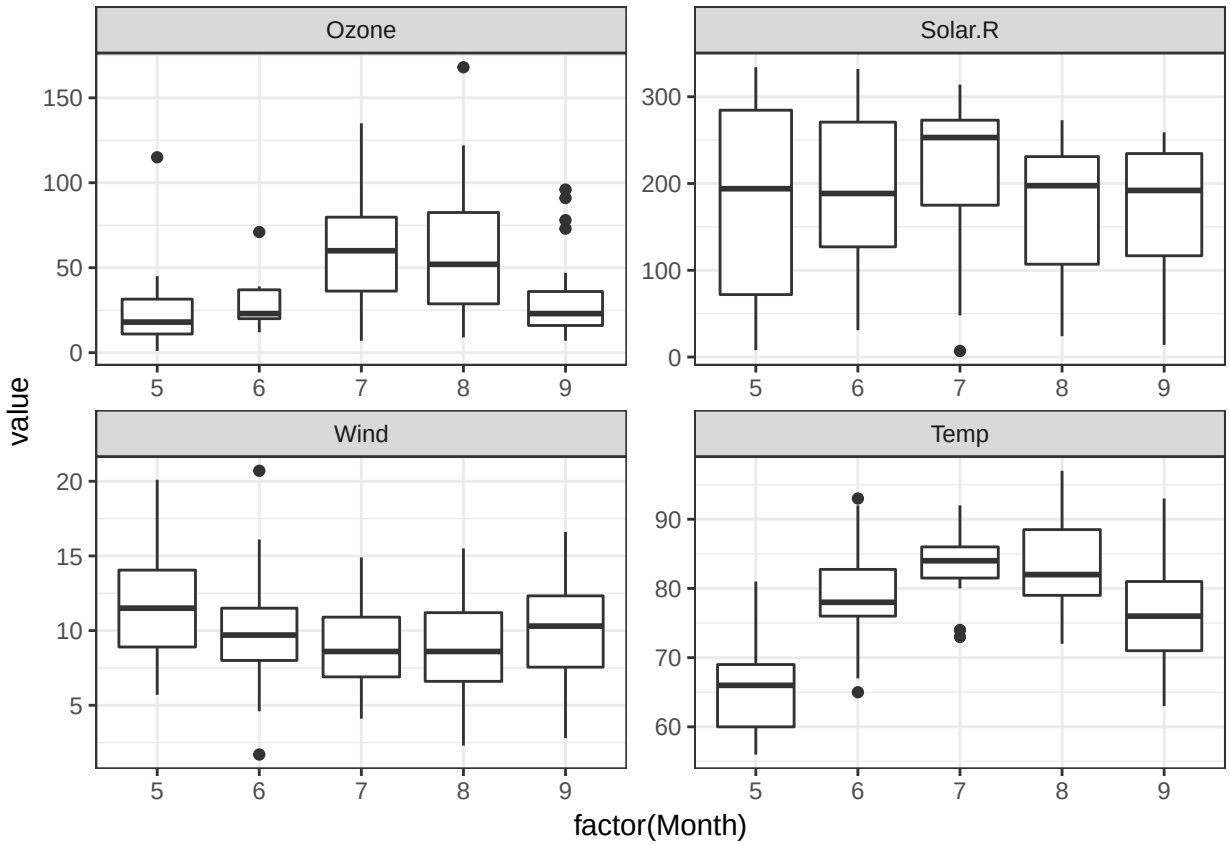
```
## Warning: Removed 44 rows containing non-finite values (stat_summary).
```

```
## Warning: Removed 44 rows containing missing values (geom_point).
```




```
ggplot(long_airquality, aes(x=factor(Month), y=value)) + geom_boxplot(varwidth = T) + facet_wrap(~Measu
```

```
## Warning: Removed 44 rows containing non-finite values (stat_boxplot).
```



Different Plot Types

Let's return to `diamonds` dataset and looks through a bunch of different plot types and what they show.

Just a reminder of the diamonds dataset:

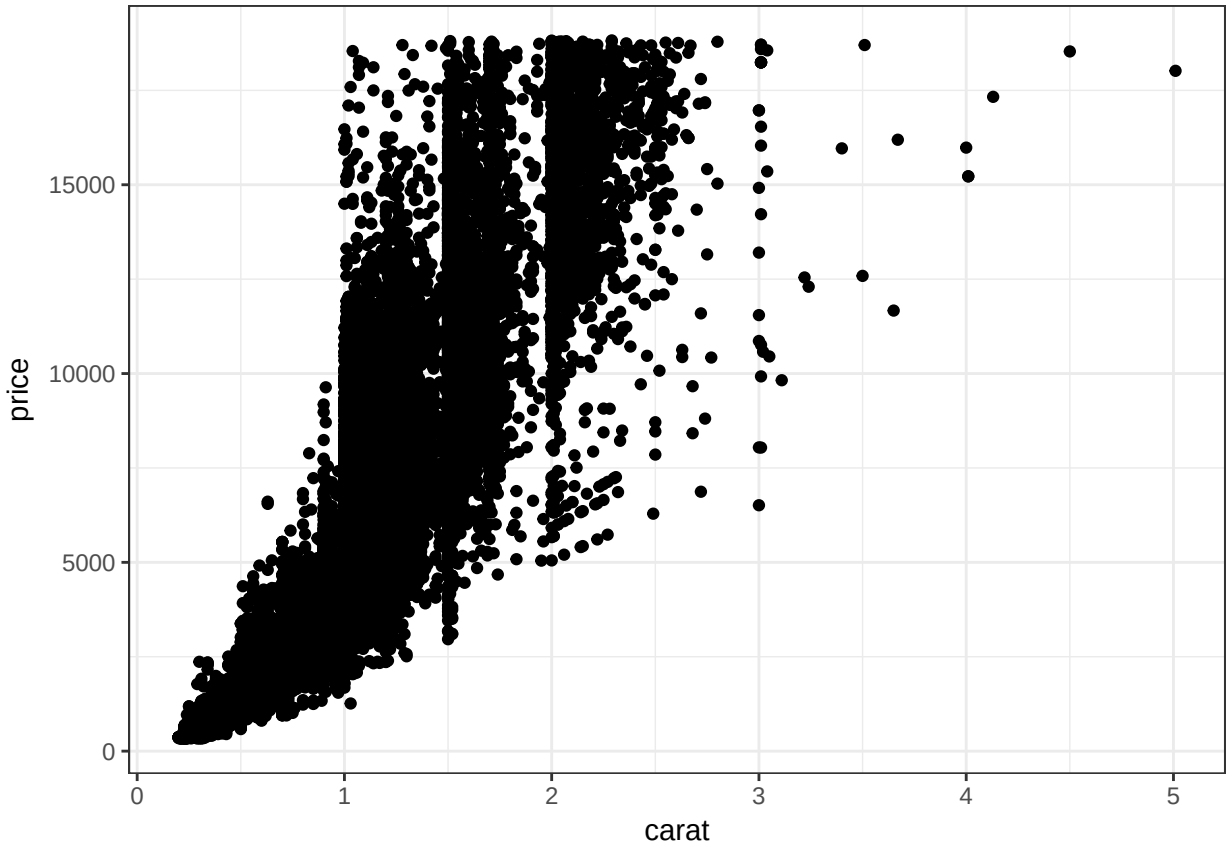
```
summary(diamonds)
```

```
##      carat               cut        color        clarity
##  Min.   :0.2000   Fair      : 1610   D: 6775   SI1    :13065
##  1st Qu.:0.4000   Good      : 4906   E: 9797   VS2    :12258
##  Median :0.7000   Very Good:12082   F: 9542   SI2    : 9194
##  Mean   :0.7979   Premium   :13791   G:11292   VS1    : 8171
##  3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2   : 5066
##  Max.   :5.0100                     I: 5422   VVS1   : 3655
##                                     J: 2808   (Other): 2531
##      depth           table           price             x
##  Min.   :43.00   Min.   :43.00   Min.   :  326   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.:  950   1st Qu.: 4.710
##  Median :61.80   Median :57.00   Median : 2401   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 3933   Mean   : 5.731
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540
##  Max.   :79.00   Max.   :95.00   Max.   :18823   Max.   :10.740
##
##        y               z
##  Min.   : 0.000   Min.   : 0.000
```

```
##  1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.710   Median : 3.530
##  Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :58.900   Max.   :31.800
##
```

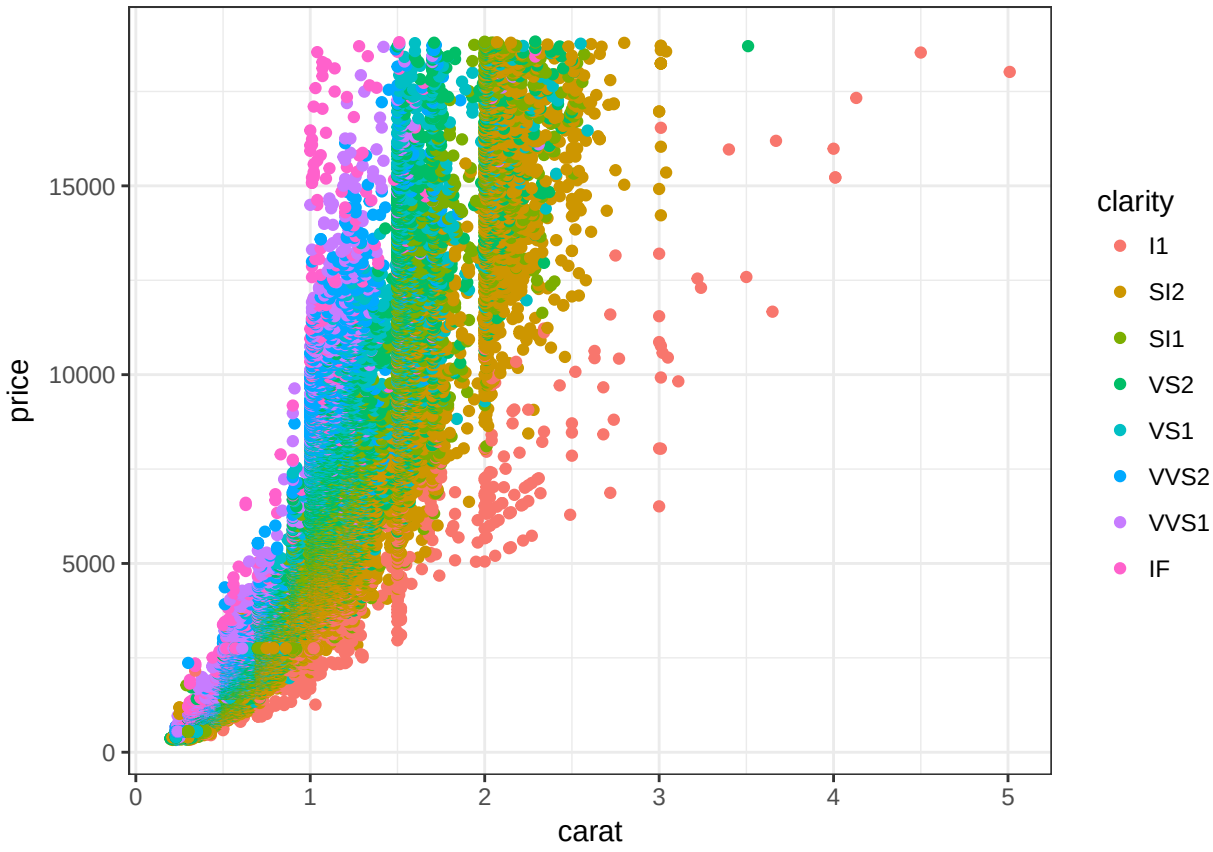
The scatter plot

```
ggplot(diamonds, aes(x=carat, y=price)) + geom_point()
```



Now let's add a visual distinction (as color) for diamond `clarity`:

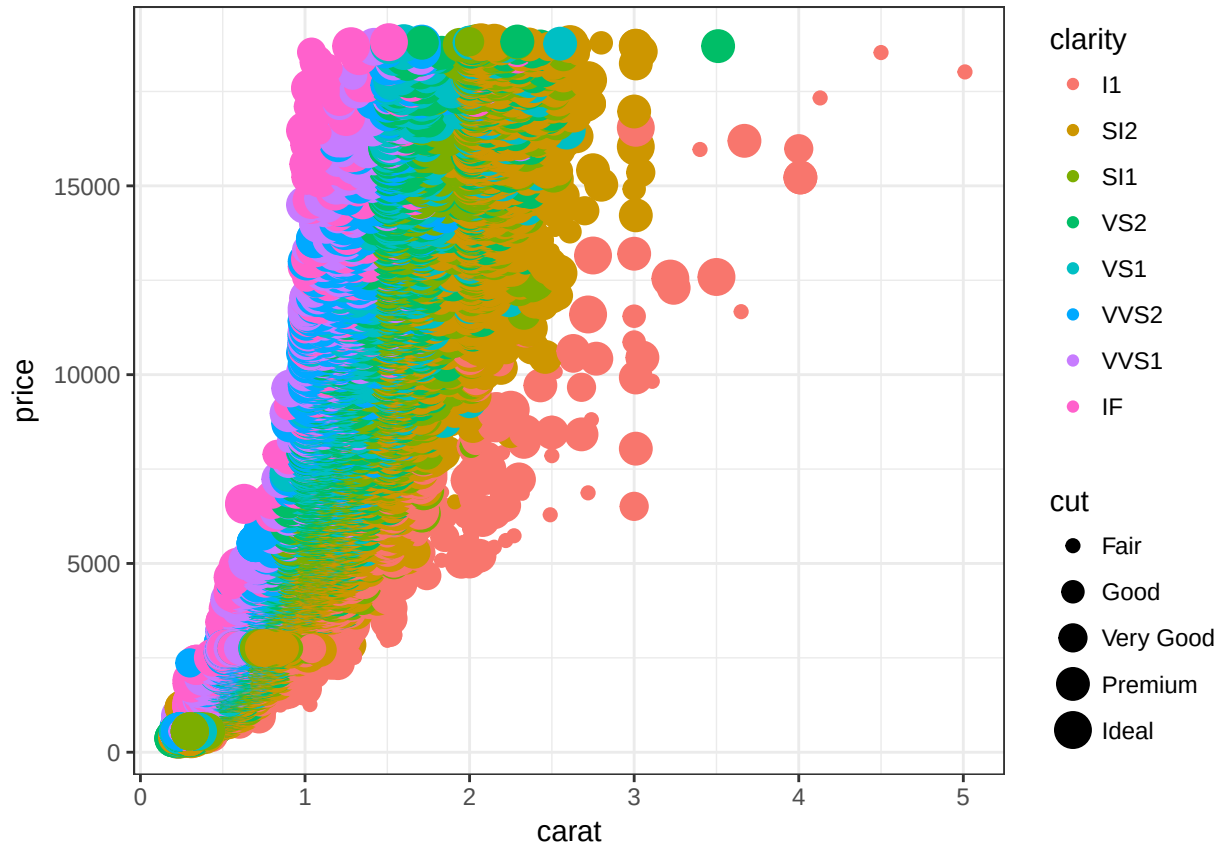
```
ggplot(diamonds, aes(x=carat, y=price, color=clarity)) + geom_point()
```



Now lets add another categorical dimension for the quality of the `cut` as the size:

```
ggplot(diamonds, aes(x=carat, y=price, color=clarity, size=cut)) + geom_point()
```

```
## Warning: Using size for a discrete variable is not advised.
```

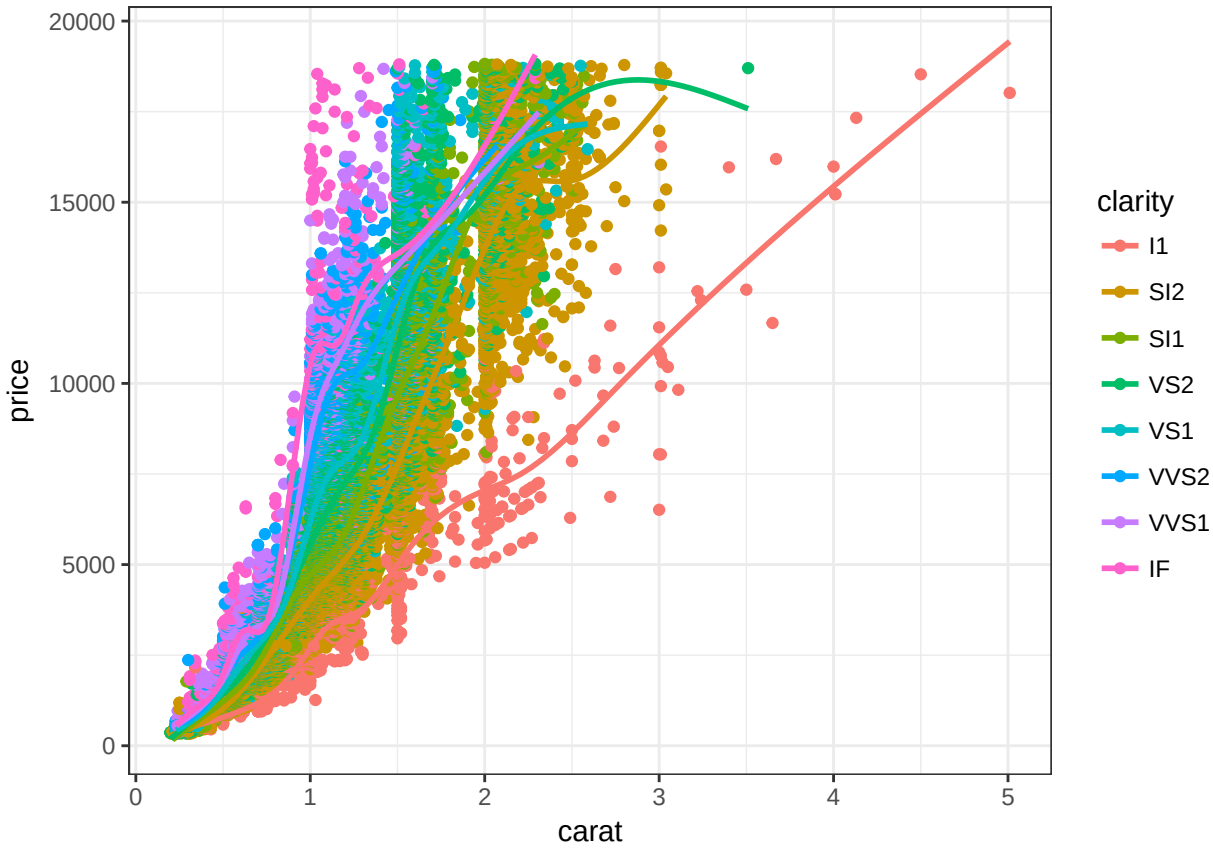


Trends

Now let's apply a smoothing trend:

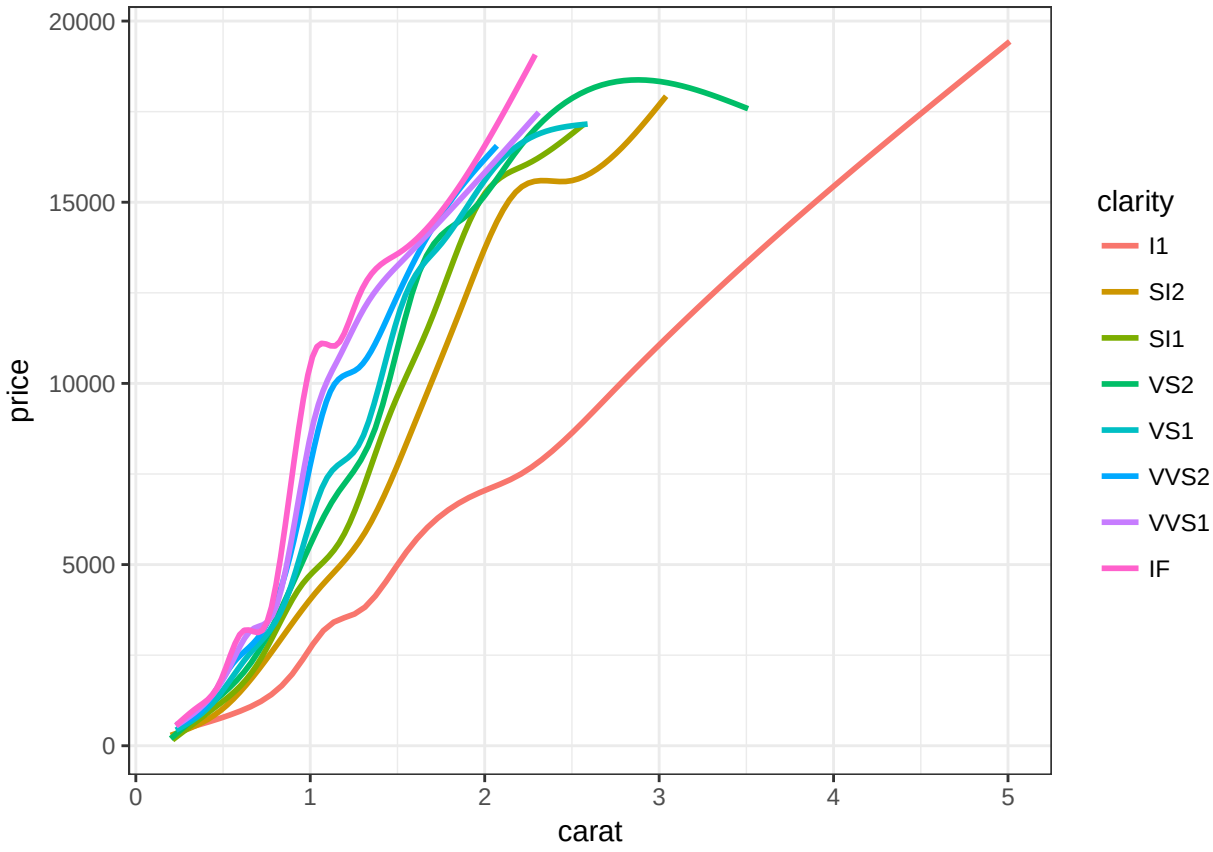
```
ggplot(diamonds, aes(x=carat, y=price, color=clarity)) + geom_point() + geom_smooth(se=F)
```

```
## `geom_smooth()` using method = 'gam'
```




```
ggplot(diamonds, aes(x=carat, y=price, color=clarity)) + geom_smooth(se=F)
```

```
## `geom_smooth()` using method = 'gam'
```

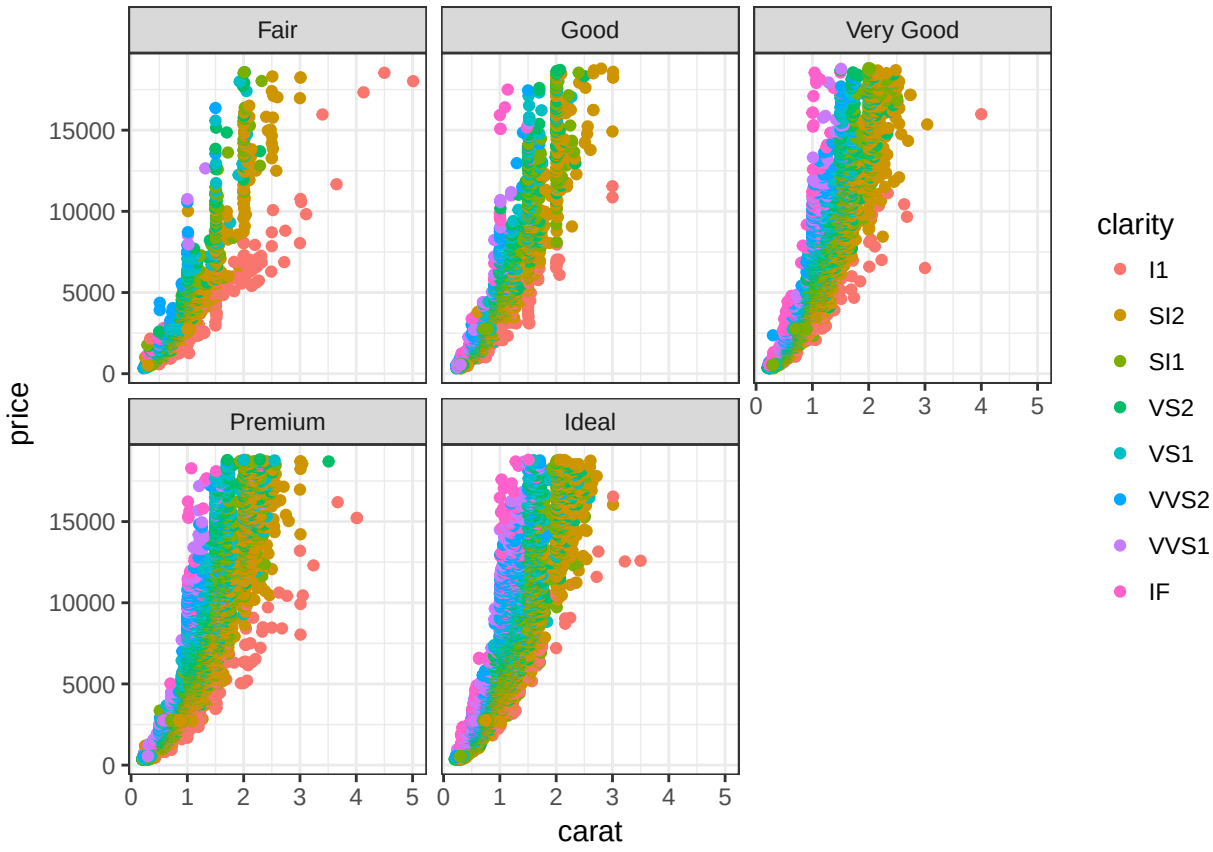


Faceting

The plot is pretty busy so lets rather divide things up according to the categorical data.

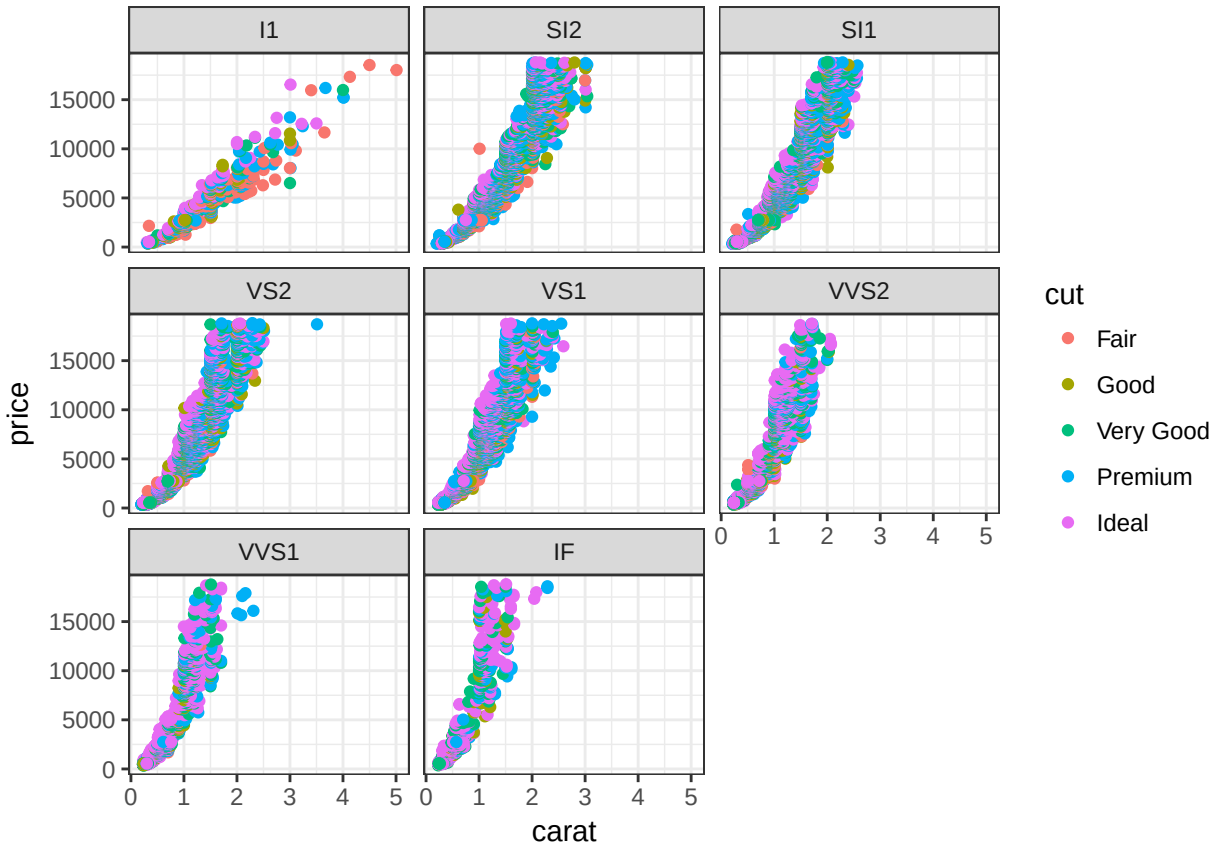
First divide by cut:

```
ggplot(diamonds, aes(x=carat, y=price, color=clarity)) + geom_point() + facet_wrap(~cut)
```



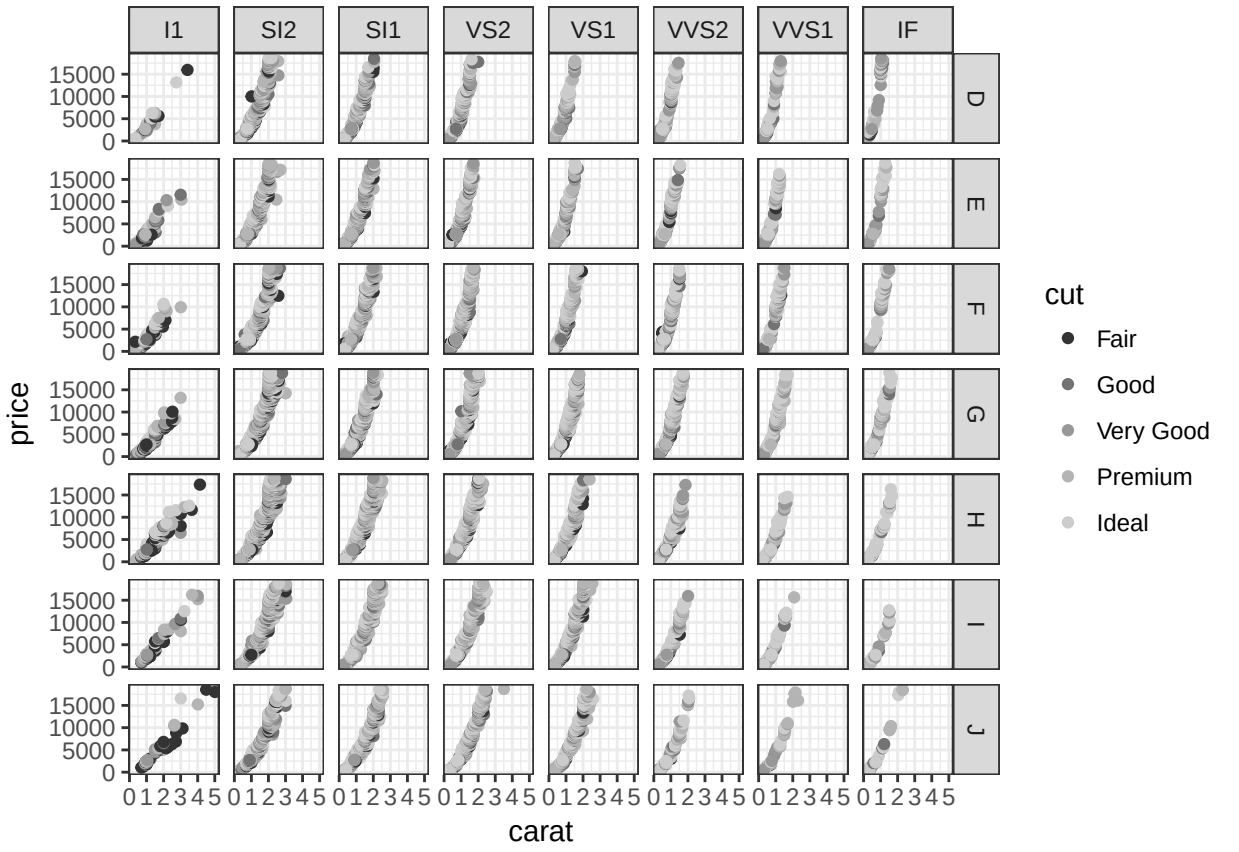
Maybe by clarity instead:

```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) + geom_point() + facet_wrap(~clarity)
```



And add a further facet for color:

```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) + geom_point() + facet_grid(color~clarity) + scale_c
```



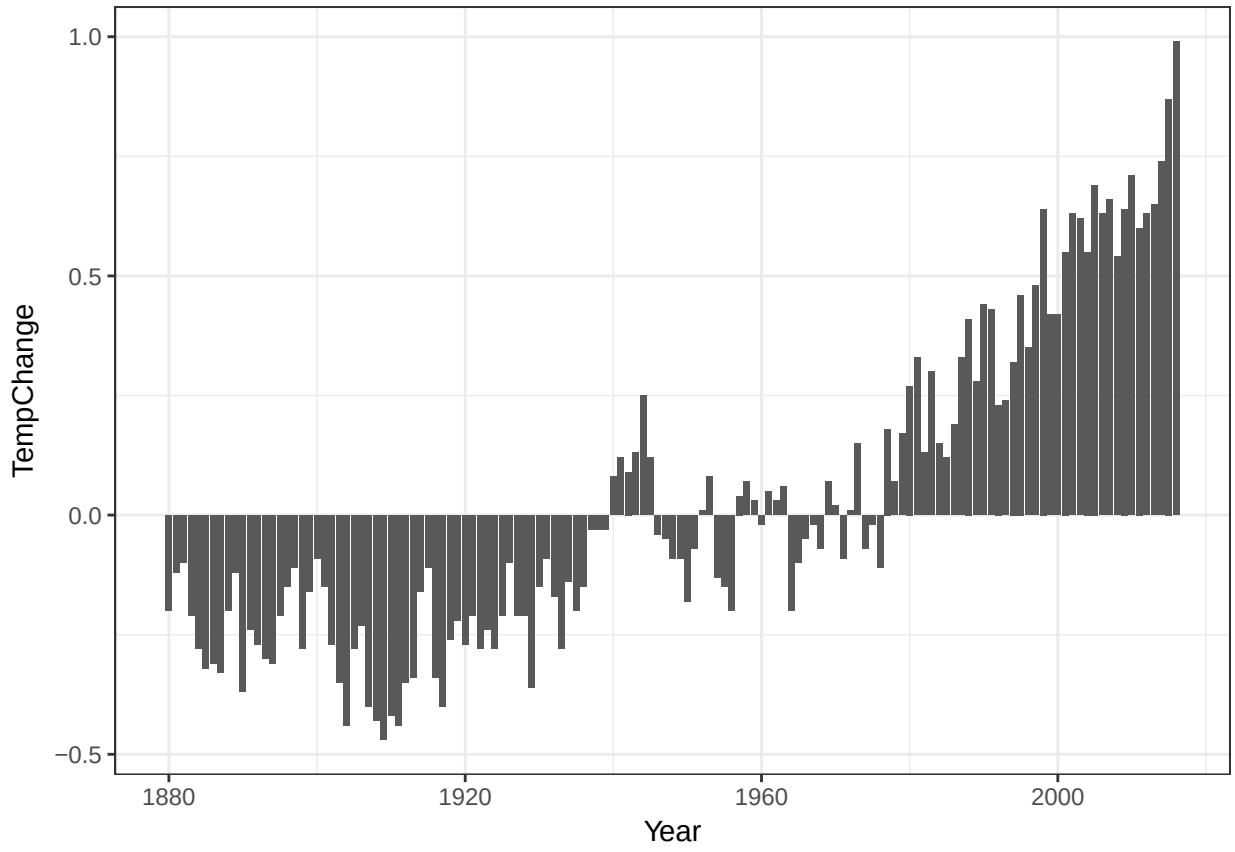
Barplot

Barplot are used for make comparisons. Let's have a look at some climate data.

```
global_temps <- read.table("647_Global_Temperature_Data_File.txt", col.names = c("Year", "TempChange",
```

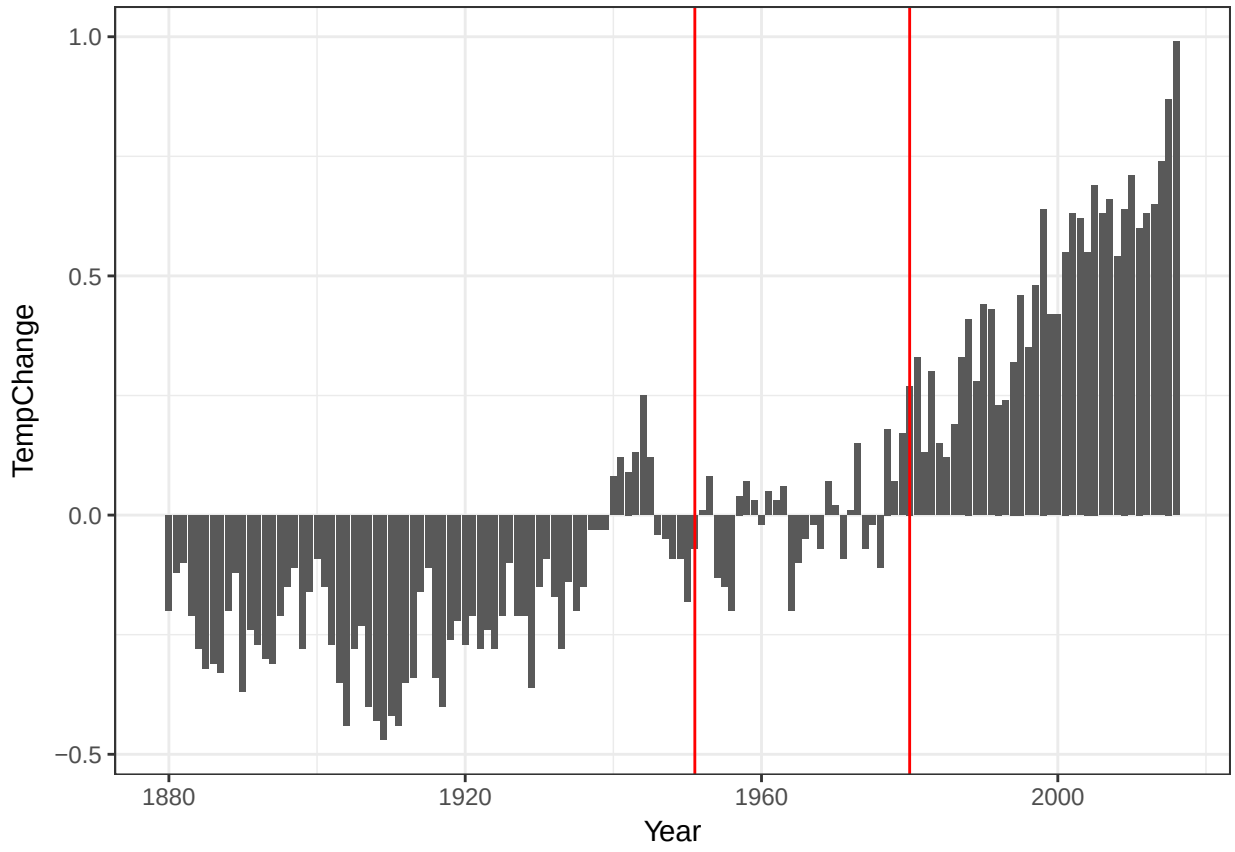
The data shows change in global surface temperature for a 136-year period, relative to 1951-1980 average temperatures.

```
ggplot(global_temps, aes(x=Year, y=TempChange)) + geom_bar(stat="identity")
```



Let's add the bounds for the comparitive years:

```
ggplot(global_temps, aes(x=Year, y=TempChange)) + geom_bar(stat="identity") + geom_vline(xintercept=c(19
```

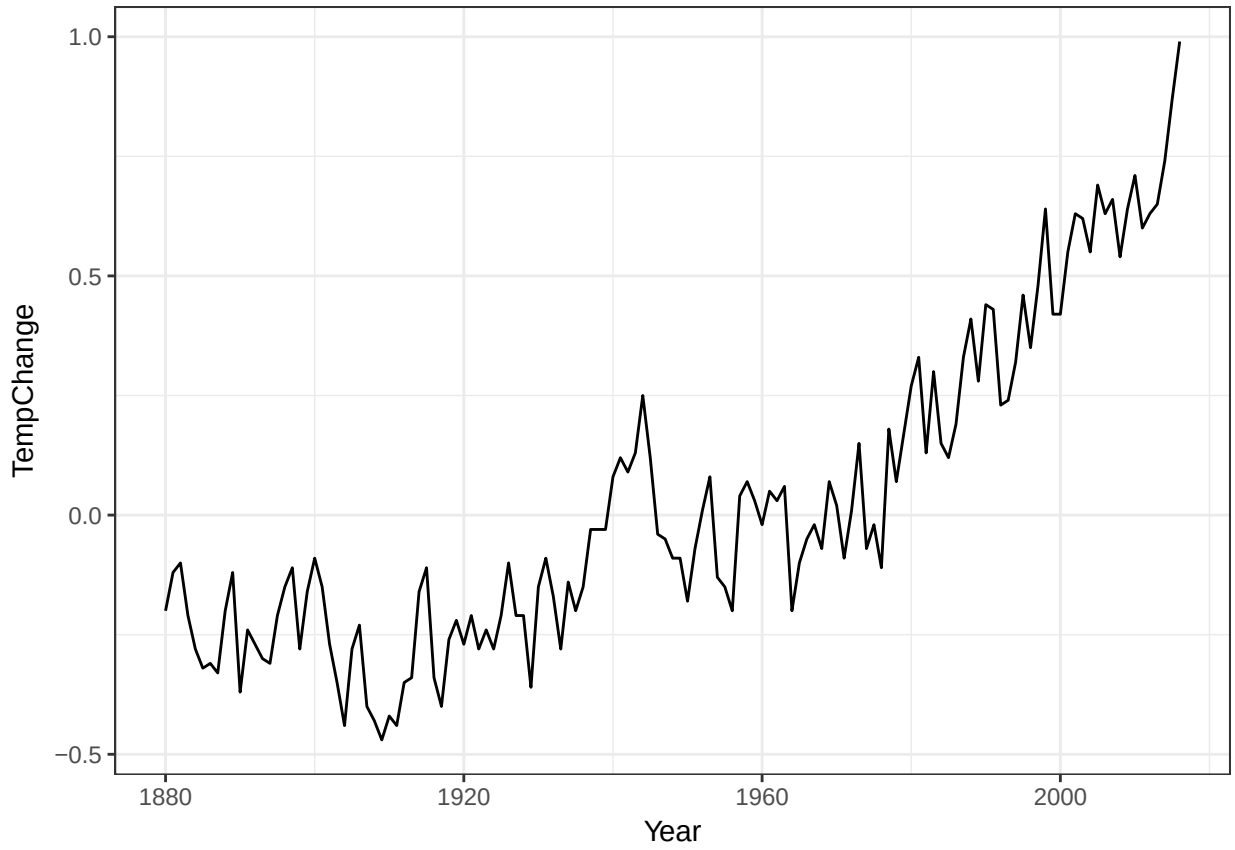


Line plots

Line plots are designed to show trends.

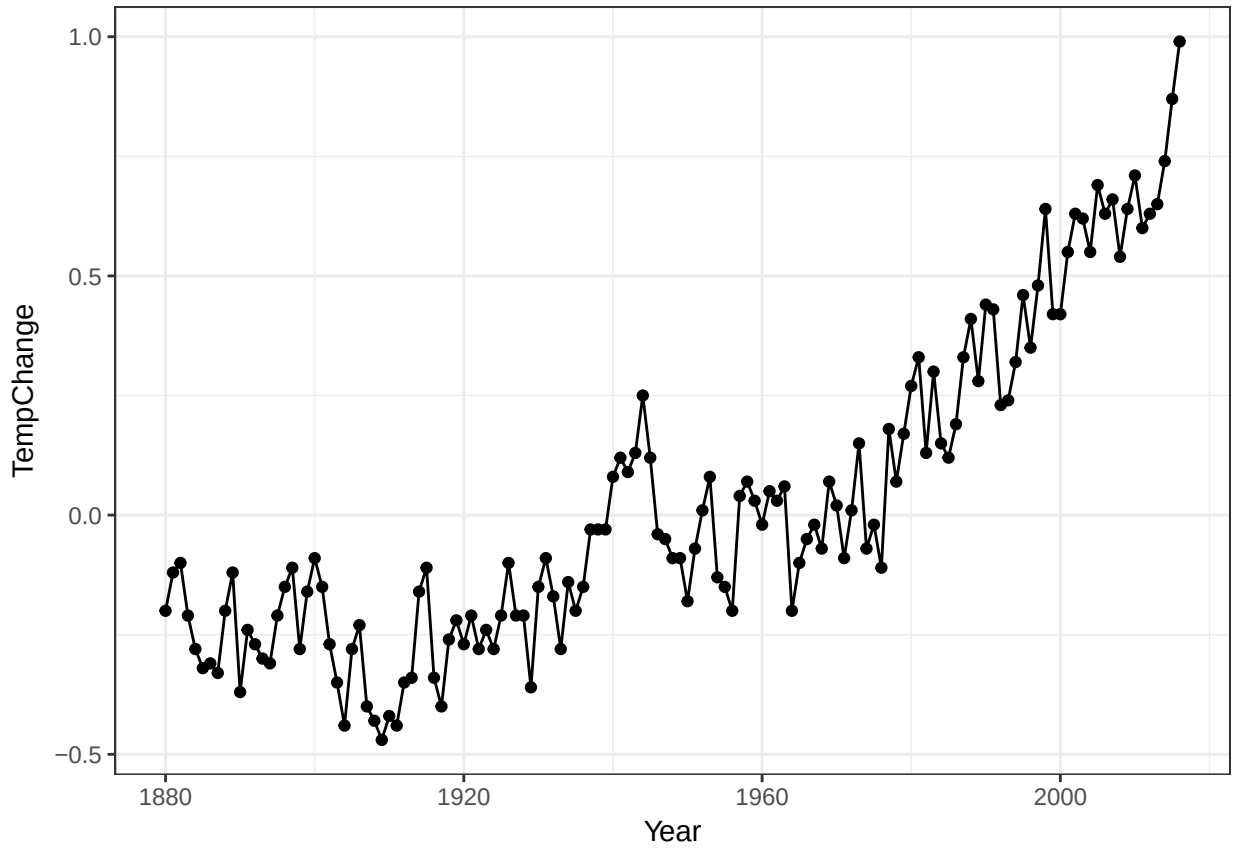
Let's look at the change in temps over the years:

```
ggplot(global_temps, aes(x=Year, y=TempChange)) + geom_line()
```



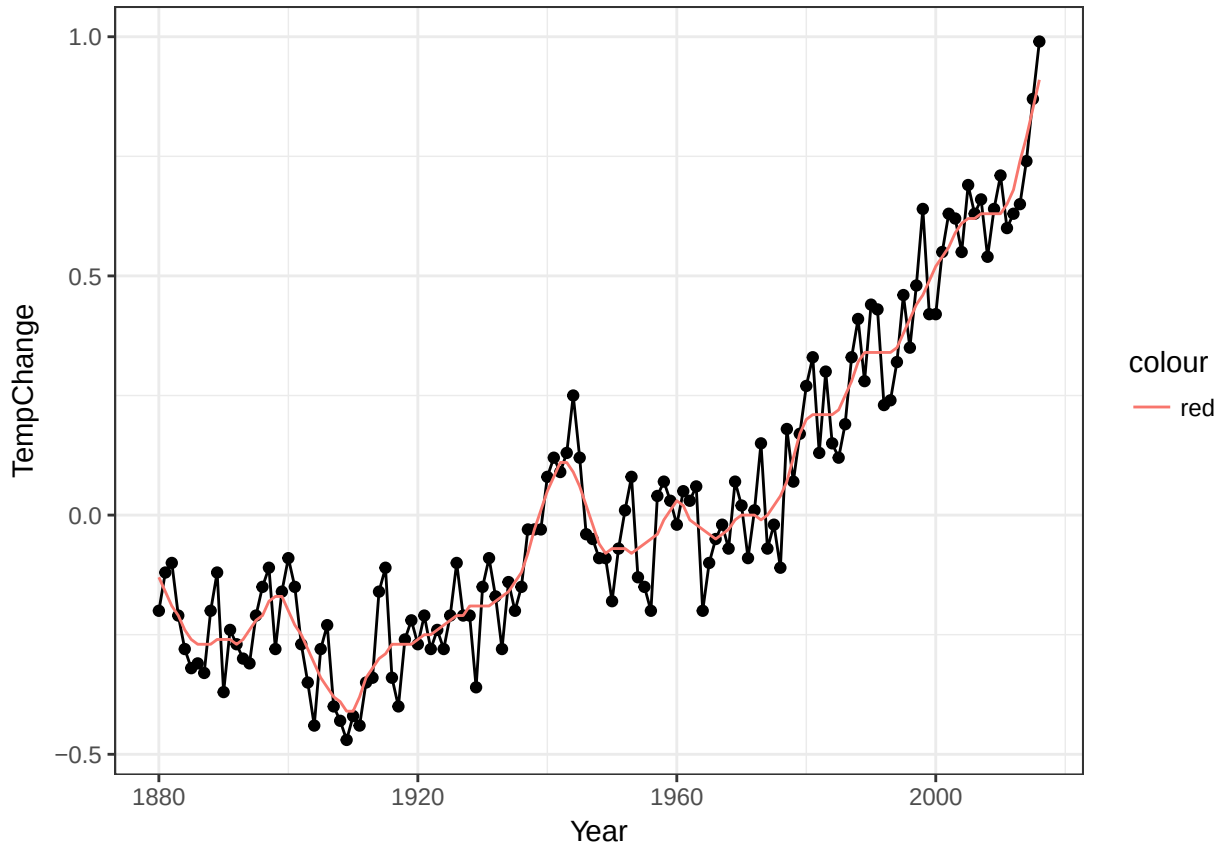
Sometimes we want a trend and to compare indvidual years - what should we do?:

```
ggplot(global_temps, aes(x=Year, y=TempChange)) + geom_line() + geom_point()
```



And adding the five-year average:

```
ggplot(global_temps, aes(x=Year, y=TempChange)) + geom_line() + geom_point() + geom_line(aes(y=TempChang
```

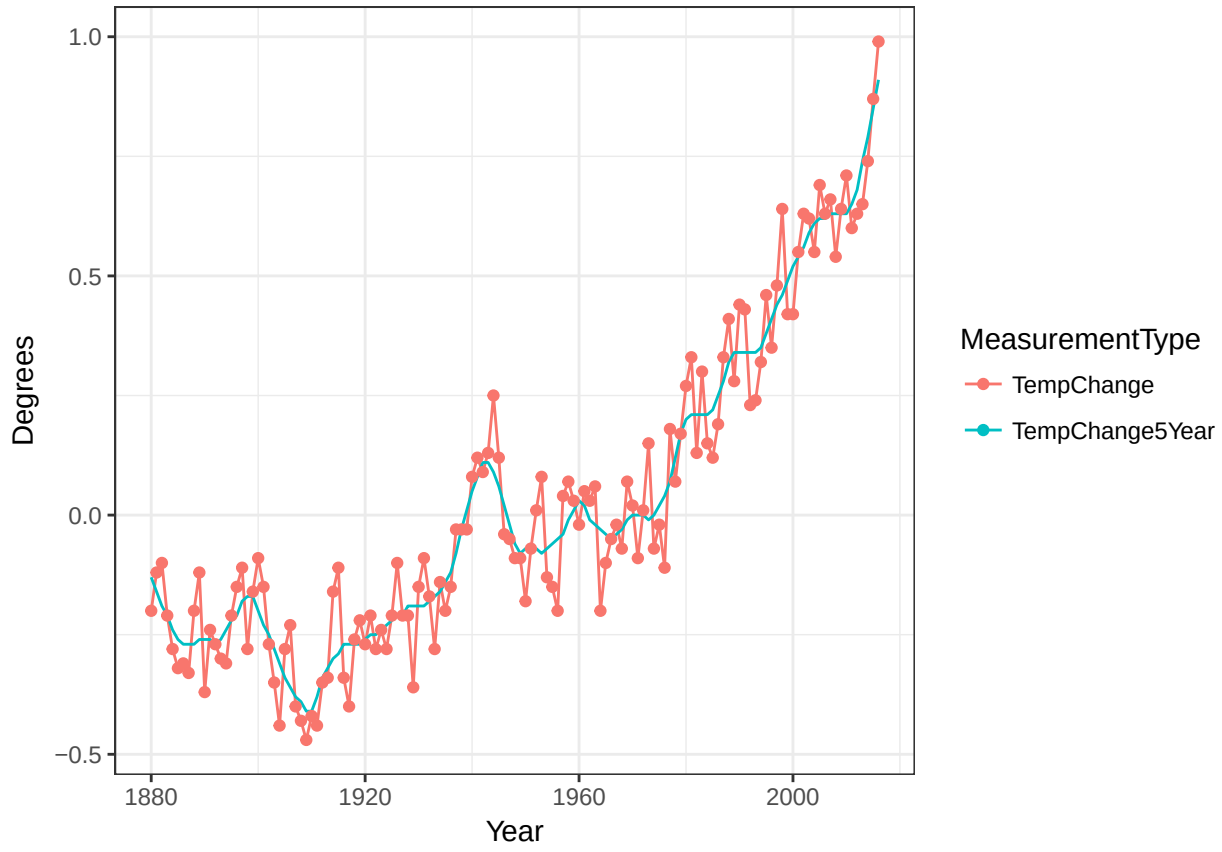


But this is not nice - better to format your data to support vis - i.e., a long format:

```
long_global_temps = melt(global_temps, id.vars = "Year", variable.name = "MeasurementType", value.name =
```

Now we can go a better job:

```
ggplot(long_global_temps, aes(x=Year, y=Degrees, color=MeasurementType)) + geom_line() + geom_point(data
```

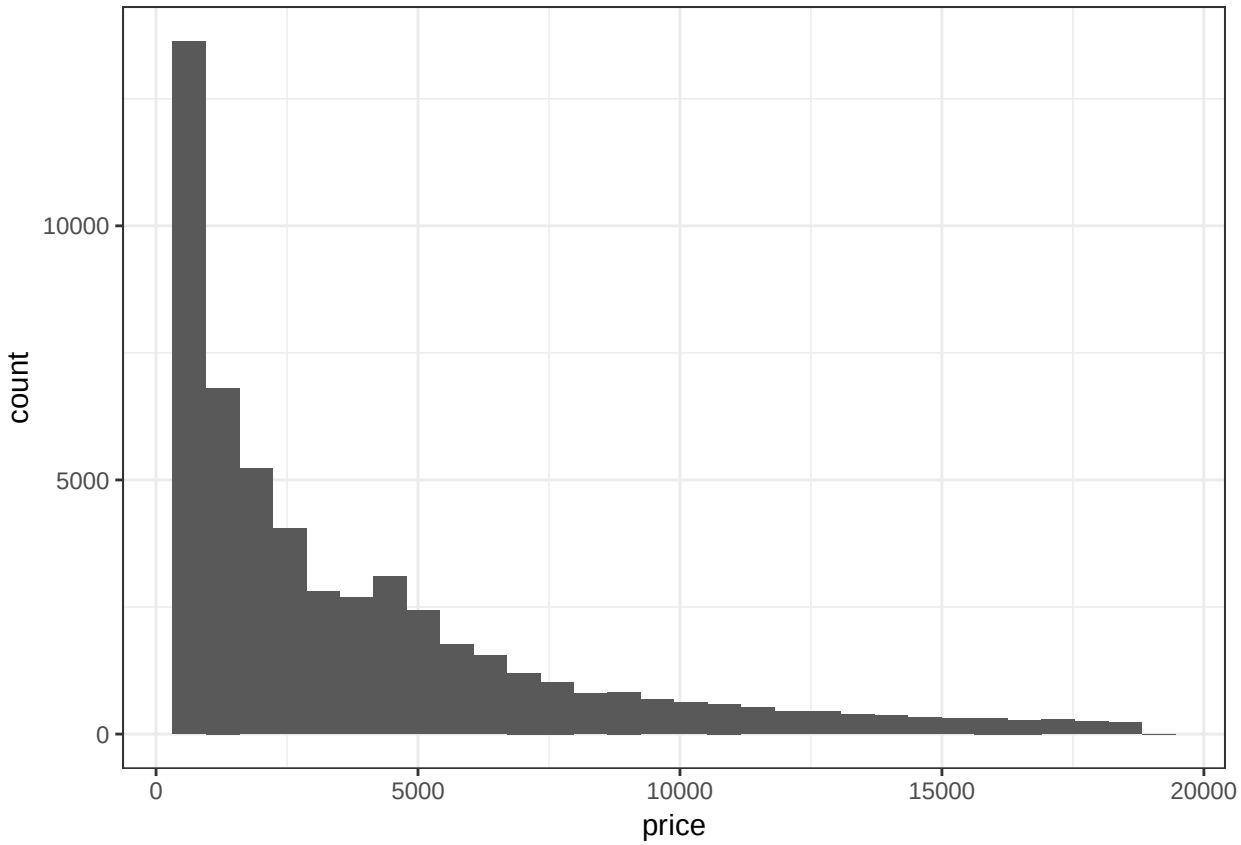


Histograms

Histograms are used to convey an understanding about the distribution of your data. For example, the distribution on `price`:

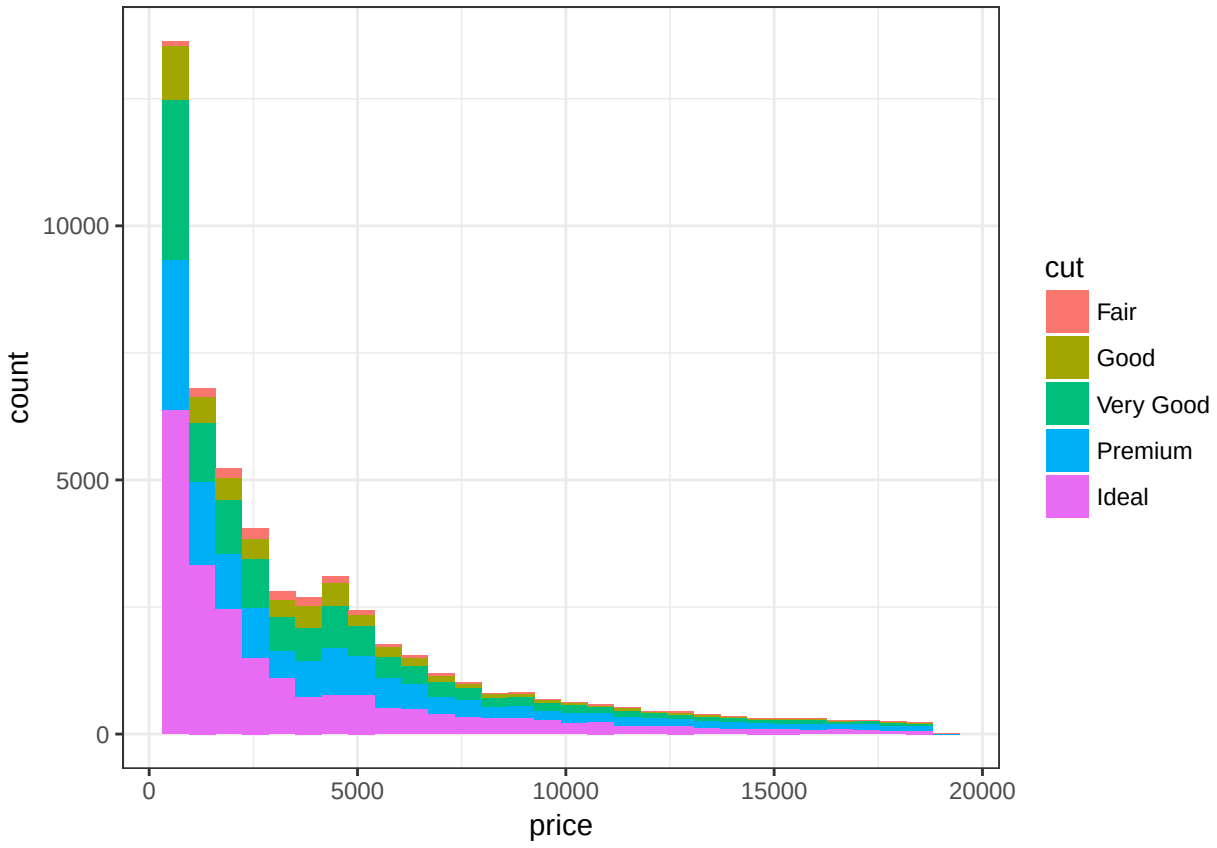
```
ggplot(diamonds, aes(x=price)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(diamonds, aes(x=price, fill=cut)) + geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

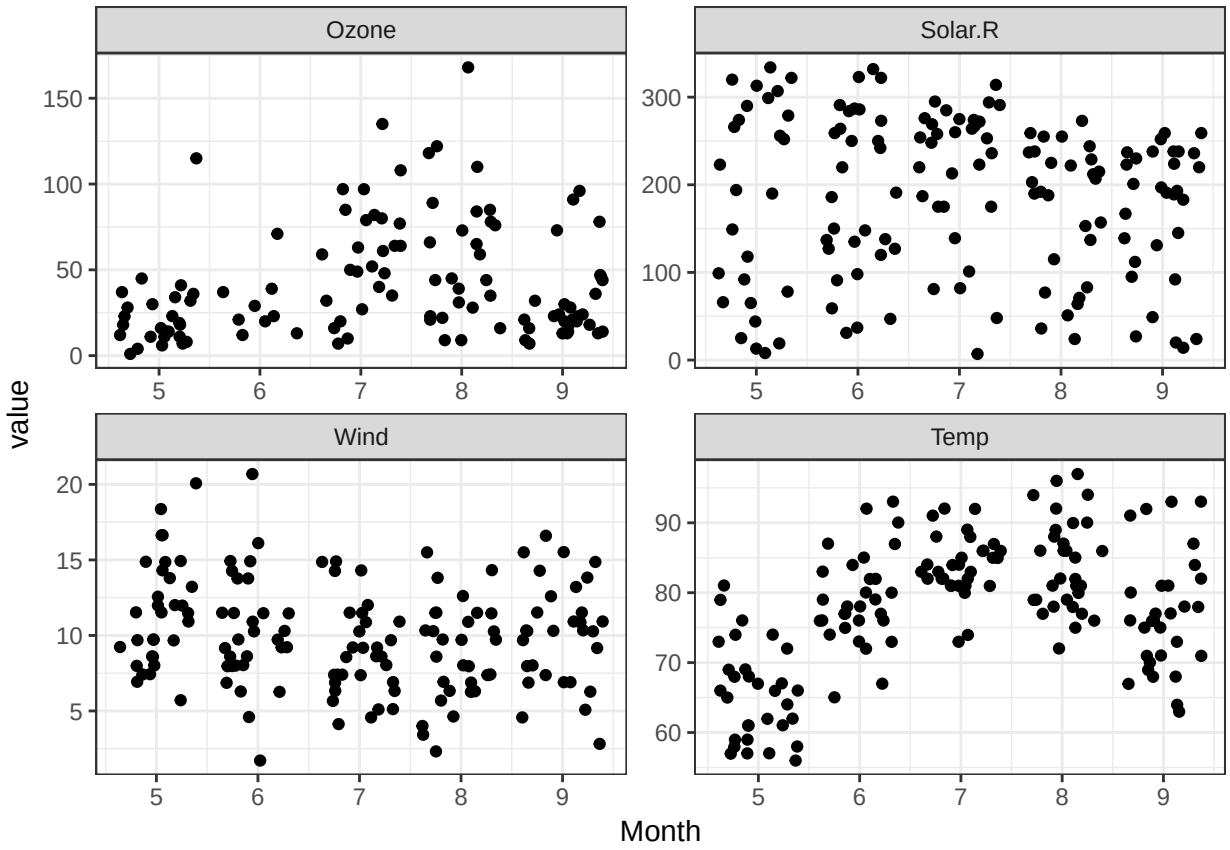


Boxplots

Box plots provide a way of showing distributions and averages. Let's return to the `airquality` dataset:

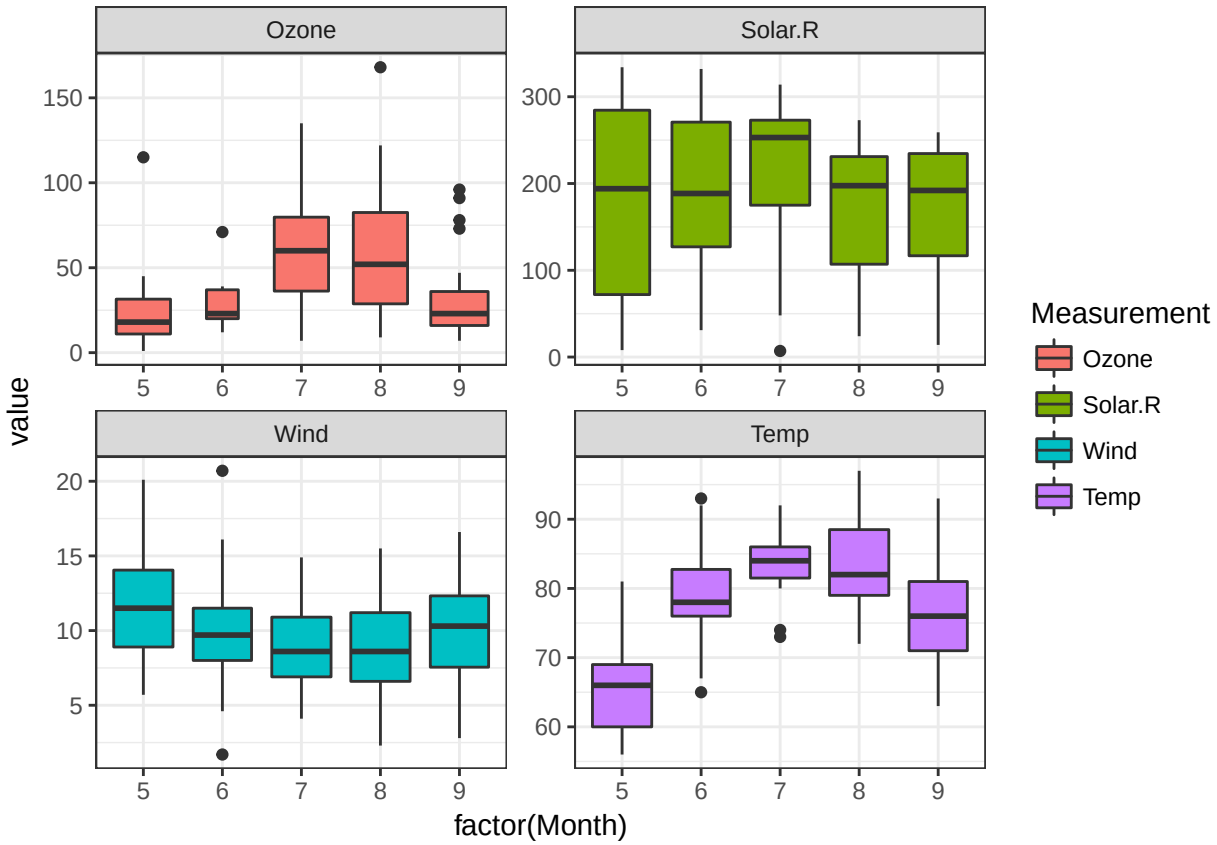
```
ggplot(long_airquality, aes(x=Month, y=value)) + geom_jitter() + facet_wrap(~Measurement, scales="free"
```

```
## Warning: Removed 44 rows containing missing values (geom_point).
```




```
ggplot(long_airquality, aes(x=factor(Month), y=value, fill=Measurement)) + geom_boxplot(varwidth = T) +
```

```
## Warning: Removed 44 rows containing non-finite values (stat_boxplot).
```



What us going on with Ozone measurements for June:

```
subset(long_airquality, Measurement=="Ozone" & Month==6)
```

```
##    Month Day Measurement value
## 32     6   1       Ozone    NA
## 33     6   2       Ozone    NA
## 34     6   3       Ozone    NA
## 35     6   4       Ozone    NA
## 36     6   5       Ozone    NA
## 37     6   6       Ozone    NA
## 38     6   7       Ozone    29
## 39     6   8       Ozone    NA
## 40     6   9       Ozone    71
## 41     6  10       Ozone    39
## 42     6  11       Ozone    NA
## 43     6  12       Ozone    NA
## 44     6  13       Ozone    23
## 45     6  14       Ozone    NA
## 46     6  15       Ozone    NA
## 47     6  16       Ozone    21
## 48     6  17       Ozone    37
## 49     6  18       Ozone    20
## 50     6  19       Ozone    12
## 51     6  20       Ozone    13
## 52     6  21       Ozone    NA
## 53     6  22       Ozone    NA
```

```
## 54      6  23       Ozone    NA
## 55      6  24       Ozone    NA
## 56      6  25       Ozone    NA
## 57      6  26       Ozone    NA
## 58      6  27       Ozone    NA
## 59      6  28       Ozone    NA
## 60      6  29       Ozone    NA
## 61      6  30       Ozone    NA
```

Heatmap

A heat map is a two-dimensional representation of data in which values are represented by colors. A simple heat map provides an immediate visual summary of information. More elaborate heat maps allow the viewer to understand complex data sets.

Let's use the diamond dataset to look at how clarity, color and price are related.

First we will do some aggregation on the data:

```
agg_diamonds <- aggregate(price ~ clarity + color, diamonds, mean)
```

```
ggplot(agg_diamonds, aes(x=clarity, y=color, fill=price)) + geom_tile(colour = "white") + scale_fill_gra
```