MONASH University

ETX2250/ETF5922: Data Visualization and Analytics

Basic Visualization

Lecturer: *Lauren Kennedy*

Department of Econometrics and Business Statistics

lauren.kennedy1@monash.edu

Week 2

The grammar of graphics

- At first using `ggplot2` can seem too complicated.
- Once mastered it can be used to very easily create detailed plots.
- It is built on the ideas of *Grammar of Graphics* a text by Leland Wilkinson.
- The objective is to find an abstract set of rules for creating almost any graphic.

Data

- The starting point for all visualisation is a dataset.
- In these slides, we will consider the datasets `diamonds`, `mpg` and `economics` which come built in with the `ggplot2` package.
- Later on we learn how to read in data.
- The `diamonds` data contains data on the price, size and quality of over 50000 diamonds.

Aes and Geom

- Think of an *aesthetic* (or aes) as a way of perceiving a variable:
  - Position on x or y axis
  - Color
  - Size
- Think of a *geometry* (or geom) as a way of representing a variable:
  - Points
  - Lines
- `ggplot` maps aesthetics to geometries

Histogram

Histogram

- Consider a histogram of the variable price
- In a histogram, values of the variable we are interested in lie along the horizontal (x) axis.
- The histogram creates bins then counts the number of observations in each bin.
- To get started type


```
ggplot(data = diamonds,mapping = aes(x=price))
```

What do we see?

What do we see?

- We do have an x axis with a label price and some values.
- Otherwise we see nothing.
- We need to **add** a geometry to the plot.
- We do this with the `geom_histogram` function.


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

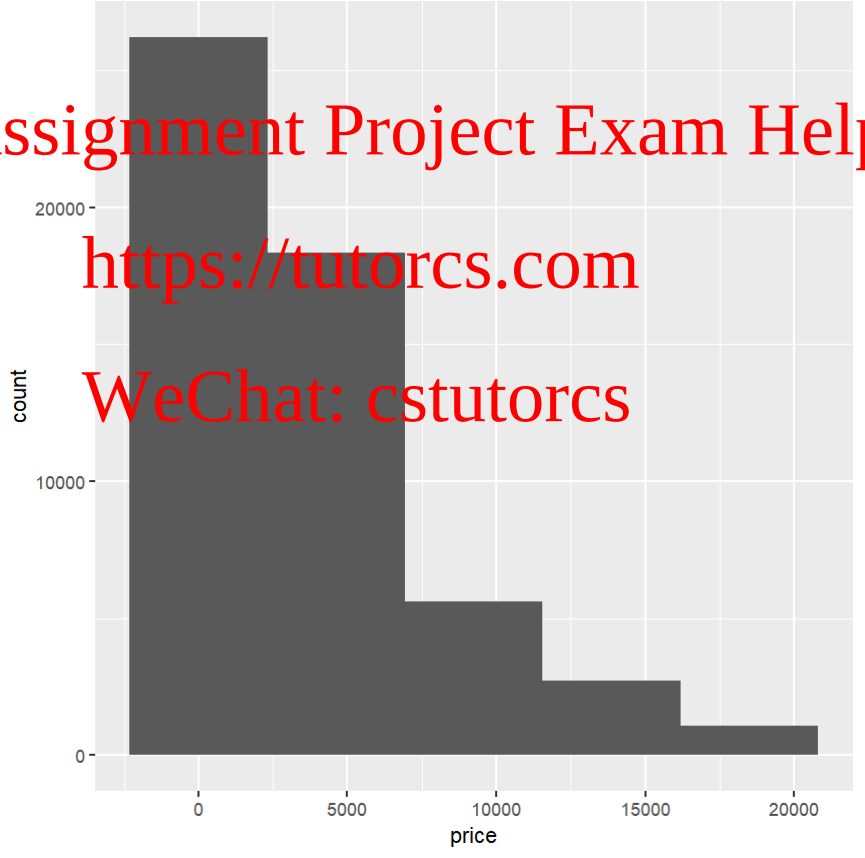
What do we see?

Modification

- Suppose want to use a different number of bins or change the color of the bins?
- These are not features of the data or the aes
- These are features of the *geom*.
- So these are controlled by arguments in the `geom_histogram` function.

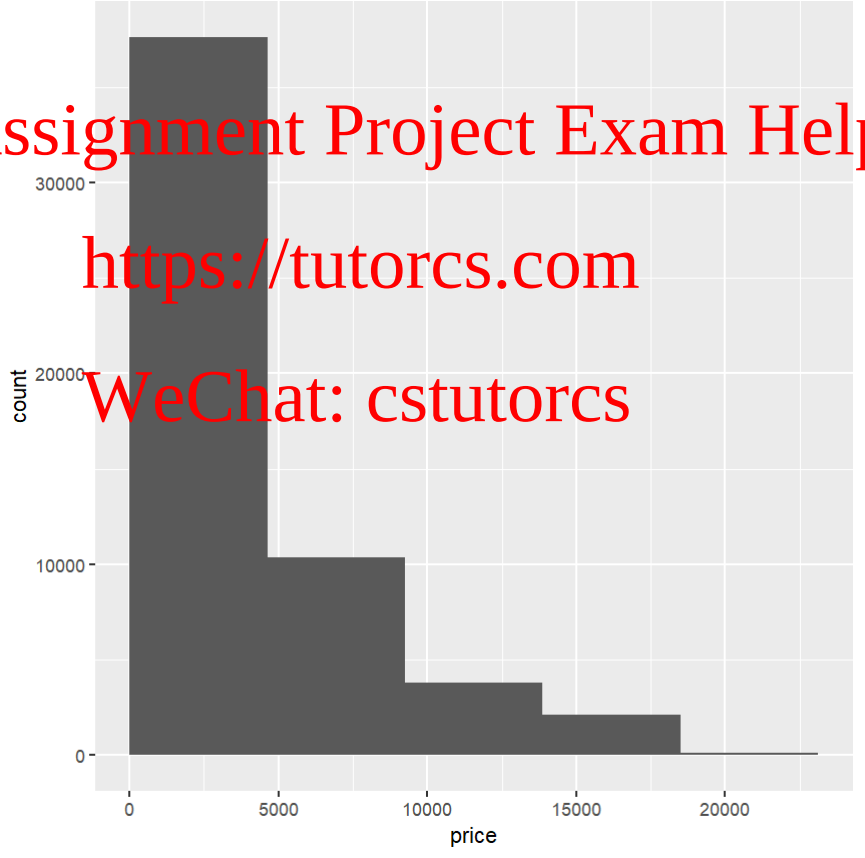
Change bins

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_histogram(bins = 5)
```

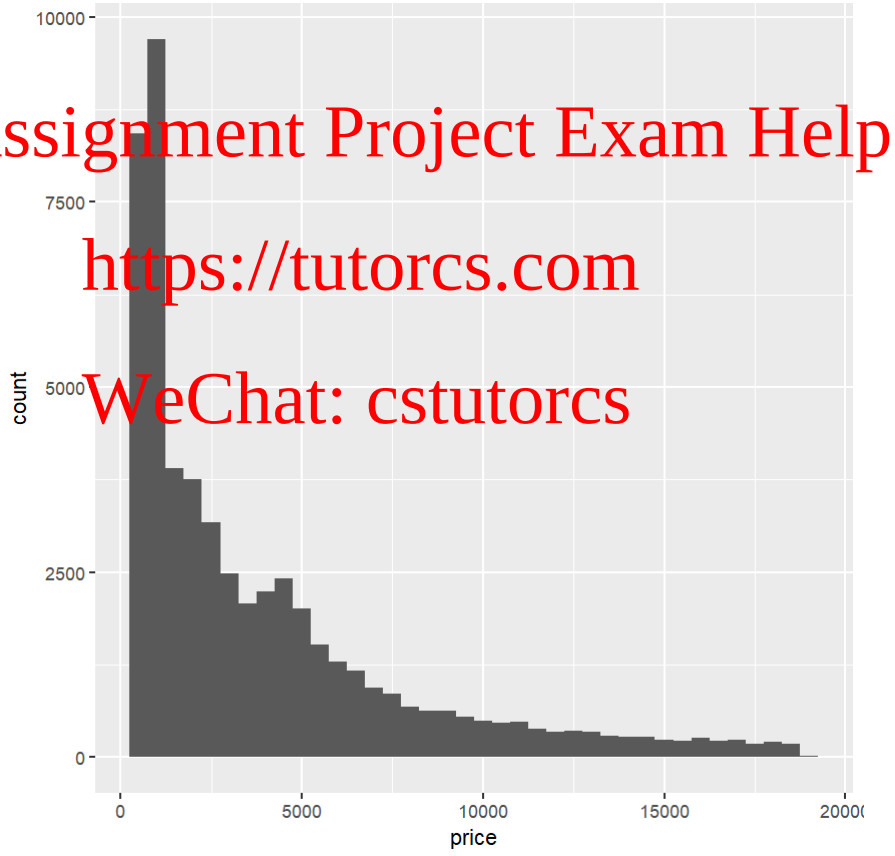
Change boundary

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_histogram(bins =  5, boundary=0)
```



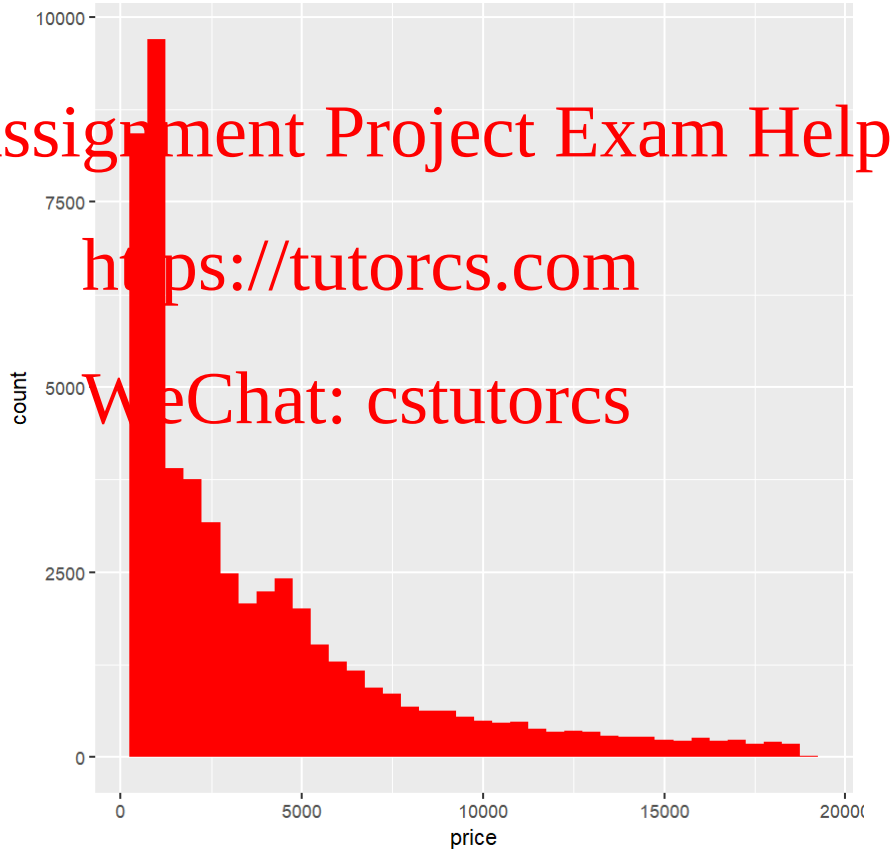
Change binwidth

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_histogram(binwidth =  500)
```



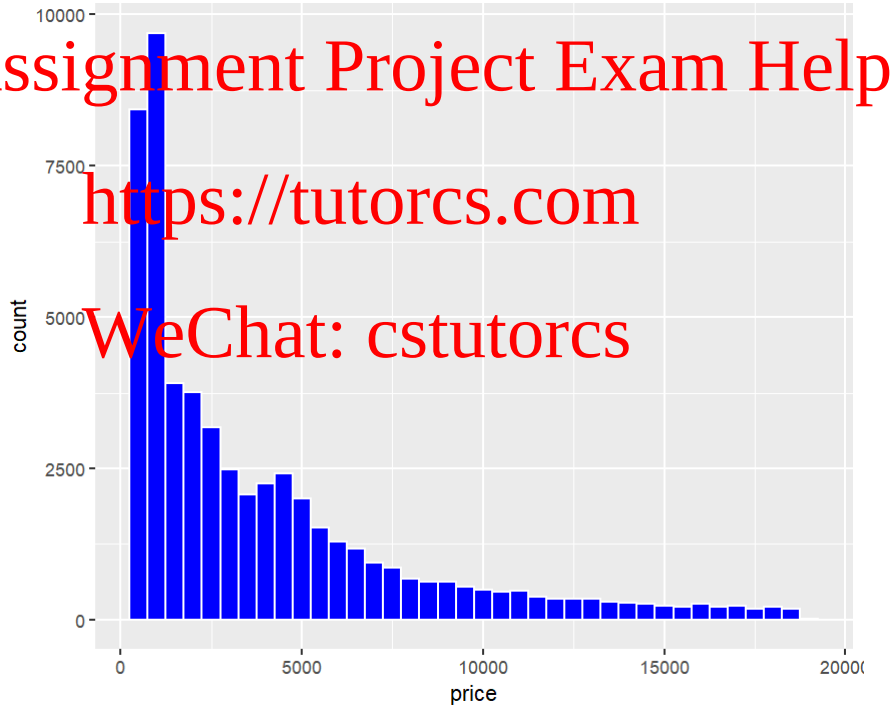
Change color

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_histogram(binwidth =  500,fill = 'red')
```



Change border color

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_histogram(binwidth =  500,color='white',
                 fill = 'blue')
```



An aside on colour

Customise colour

- Many colours come built in to R.
- In some cases you may wish to select your own color.
- Customising colour requires appreciating how a computer understands color.
- We will do this by looking at RGB hex codes.
- Using this system, to a computer *#ff0000* is red.

The RGB system

- One color model used by computers encodes every colour by the amount of *red*, *green* and *blue* light mixed to make that colour.
- This is called the RGB color model.
- A value between 0 and 255 indicates the strength of red, green and blue.
- These values between 0 and 255 are represented in two hexadecimal digits.

Hexadecimal

- In hexadecimal:
  - **a** is ten,
  - **b** is eleven,
  - **c** is twelve...
  - **f** is fifteen.
- Take the first digit and multiply by 16 and add the second digit
- Hexadecimal is used since each digit corresponds to 4 bits in computer memory.

Examples

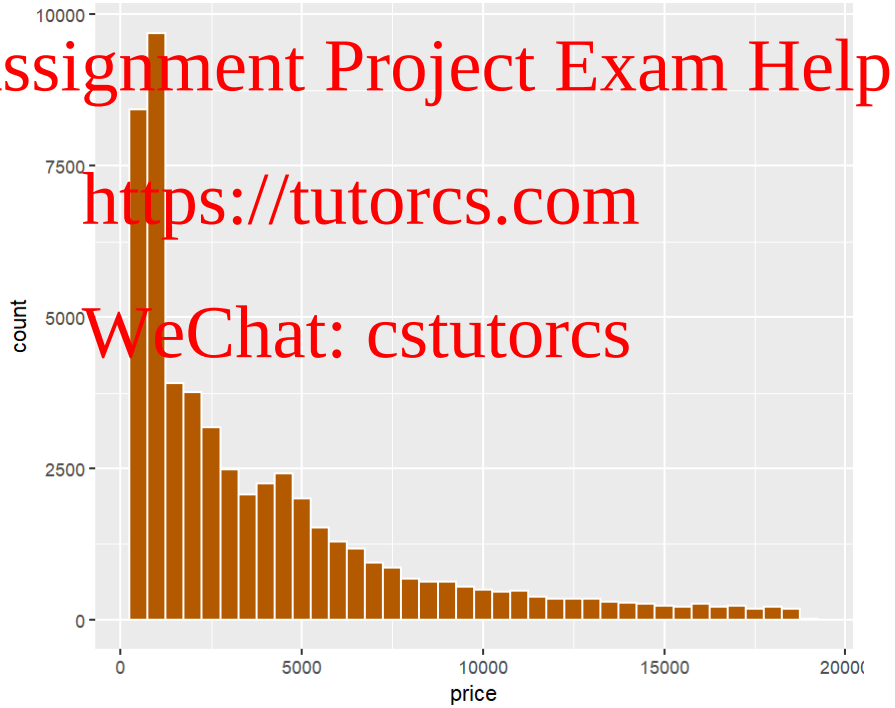
- **10** in hexadecimal is $1 \times 16 + 0 = 16$ in decimal
- **1a** in hexadecimal is $1 \times 16 + 10 = 26$ in decimal
- **2b** in hexadecimal is $2 \times 16 + 11 = 43$ in decimal
- What is e4 in decimal?

Color picker

- One online tool to find the hex code of a color is here.
- Suppose we want to the histogram to be this brown color.
- The hex code is #b35900 which is 179/256 red, 89/256 green and no blue.
- This can be provided as a string, to the `fill` or `color` argument of `geom_histogram`.

Brown histogram

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_histogram(binwidth =  500,color='white',
                 fill = '#b35900')
```

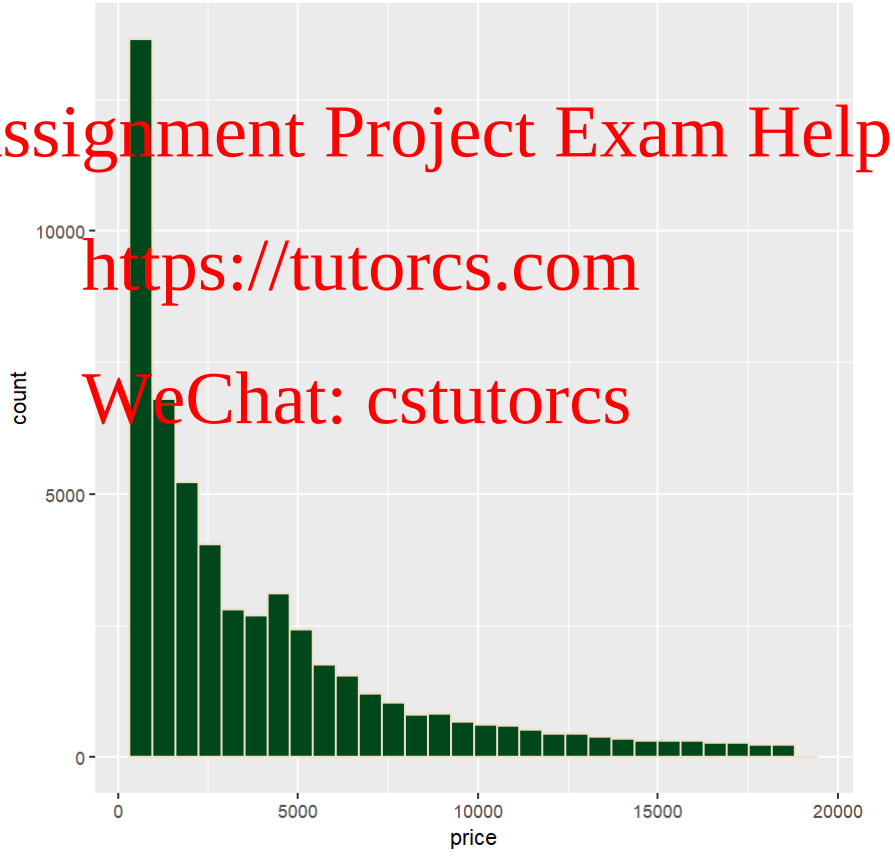


Finding hex codes

- It is useful to know hex codes since at times you may want to match colors for a specific purpose.
- For instance you may want the colors to match the brand colors of a client.
- For example a simple online search tells us that Coca Color red is #f40000
- The green color worn by NBA team the Milwaukee Bucks is #00471b.

Histograms (Bucks Colors)

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_histogram(fill='#00471b',color='#eee1c6')
```

An exercise

- Find the hex codes for a color(s) associated with:
  - A brand you like, or
  - A sports team you like, or
  - Your country's flag,
  - Anything else
- Construct a histogram of the variable *carat* with these colors.

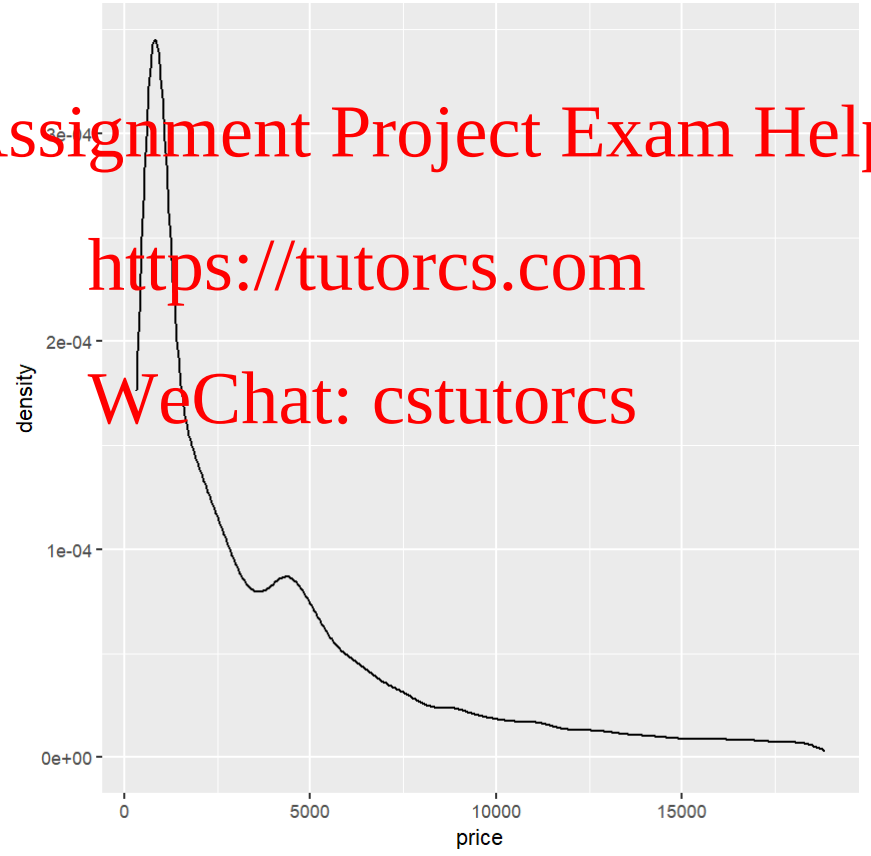
Density plot

Density

- For a smoother version of a histogram we can use a different geom called `geom_density`.
- This in fact computes a kernel density estimate of the variable.
- The level of smoothness is controlled by a *bandwidth* parameter
- All the computation is done by `ggplot2`.

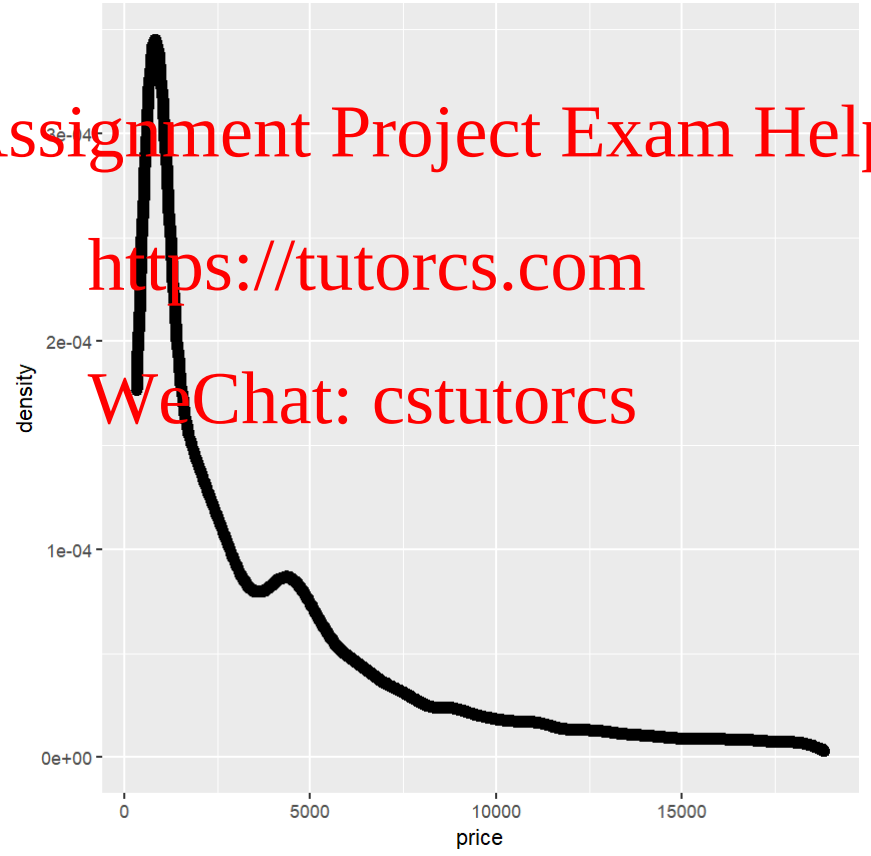
Density plot

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_density()
```

Density plot (thicker)

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_density(size=3)
```



How is density calculated?

- The kernel density estimate is a popular *nonparametric* technique that estimates a density as

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i)$$

- Here, $K_h()$ is a kernel function that depends on a bandwidth h.

Uniform kernel

- The simplest kernel function is the uniform kernel
  - $K_h(u) = 1/h$ if $|u| < h$
  - $K_h(u) = 0$ otherwise
- At a point x, the estimated density is proportional to the number of points that are *close* to x.
- By *close*, we mean within h units of x.

Extremes

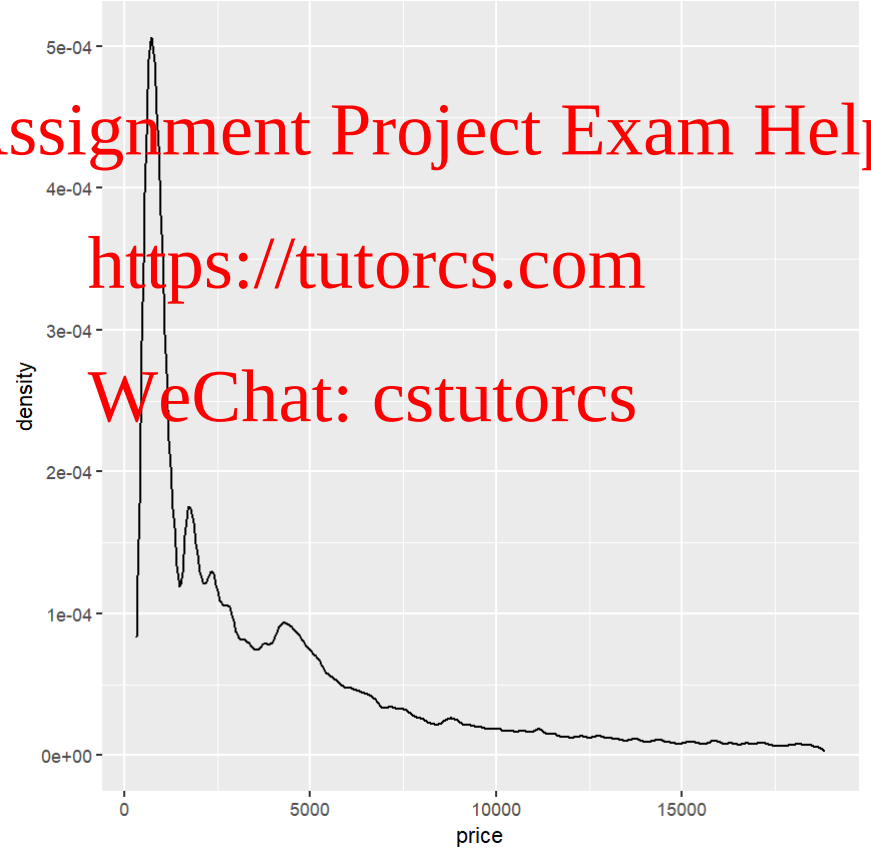
- If the bandwidth gets extremely large then for any x, all sample points are considered close.
- The formula for the kernel density becomes a flat line.
- If the bandwidth gets extremely small then for any x we choose, the density is just the number of points in the sample equal to x.
- The kernel density is made up of spikes at the sample points.

Defaults

- By default, `geom_density`
  - Uses a Gaussian kernel
  - Selects the bandwidth using *Silverman's rule of thumb*
- The same principles apply:
  - Large bandwidth leads to more smoothness
  - Small bandwidth leads to more bumpiness

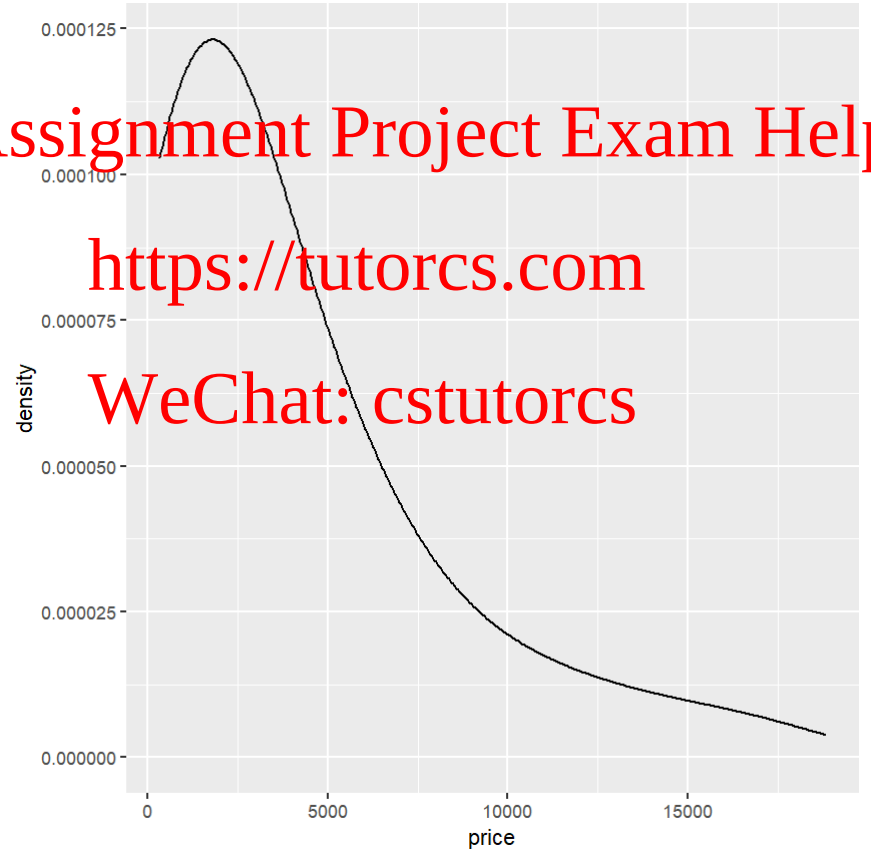
Density plot: Low bandwidth

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_density(bw=100)
```



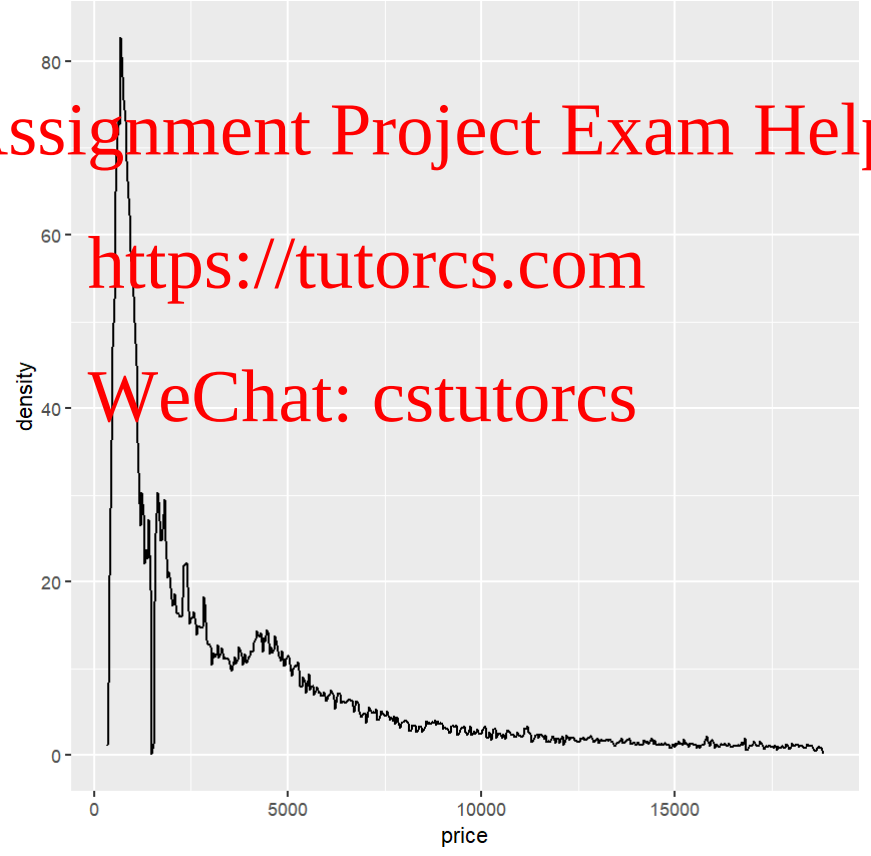
Density plot: High bandwidth

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_density(bw=2000)
```



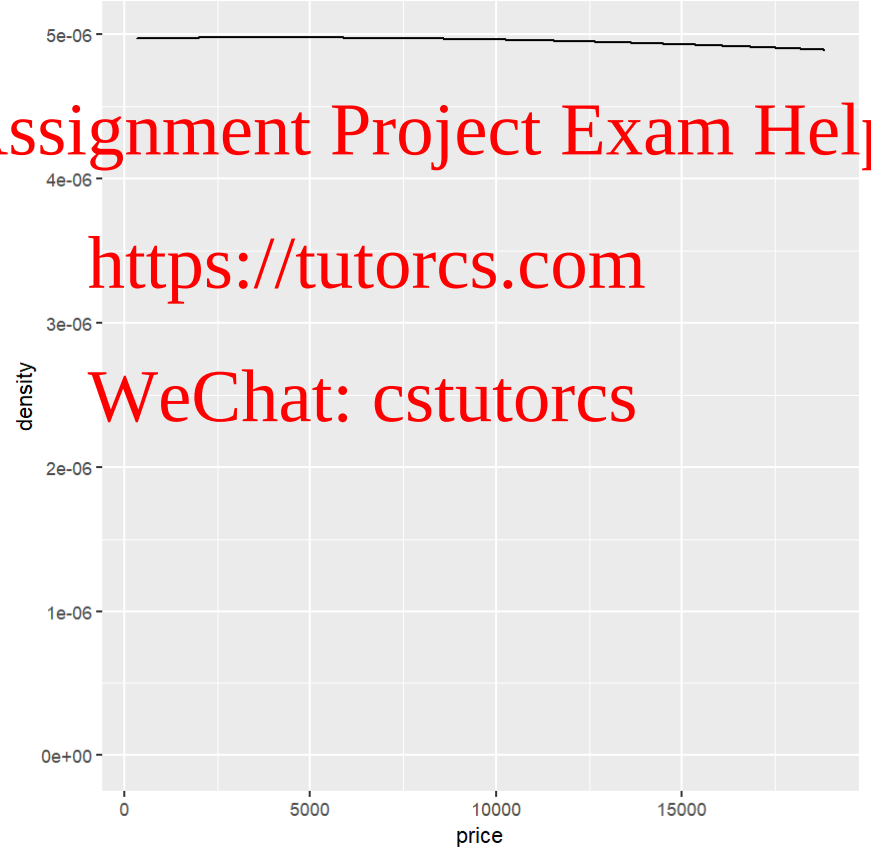
Density plot: Low bandwidth

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_density(bw=0.0001)
```



Density plot: High bandwidth

```
ggplot(data = diamonds,mapping = aes(x=price))+
  geom_density(bw=80000)
```



Summary

- With both histograms and density plots
  - If the bin width or bandwidth is too small the plot may look bumpy. This can exaggerate features that are not significant.
  - If the bin width or bandwidth is too large the plot may smooth over important features like local modes.
- Always try a few different values of bin width or bandwidth.

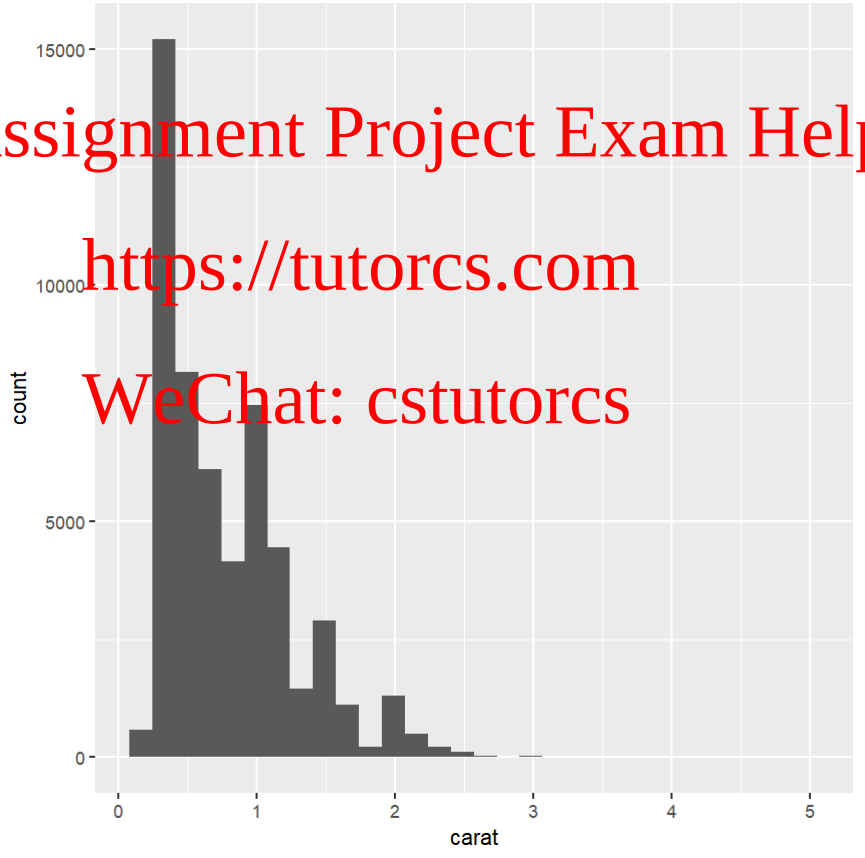
Finding outliers

Outliers

- Histograms and density plots give a good idea of shape and local modes.
- Sometimes they can obscure outliers.
- For finding outliers a rug plot can be useful
- For finding outliers while still getting a good idea of skew, boxplots can be useful.
- We can investigate using the variable *carat*

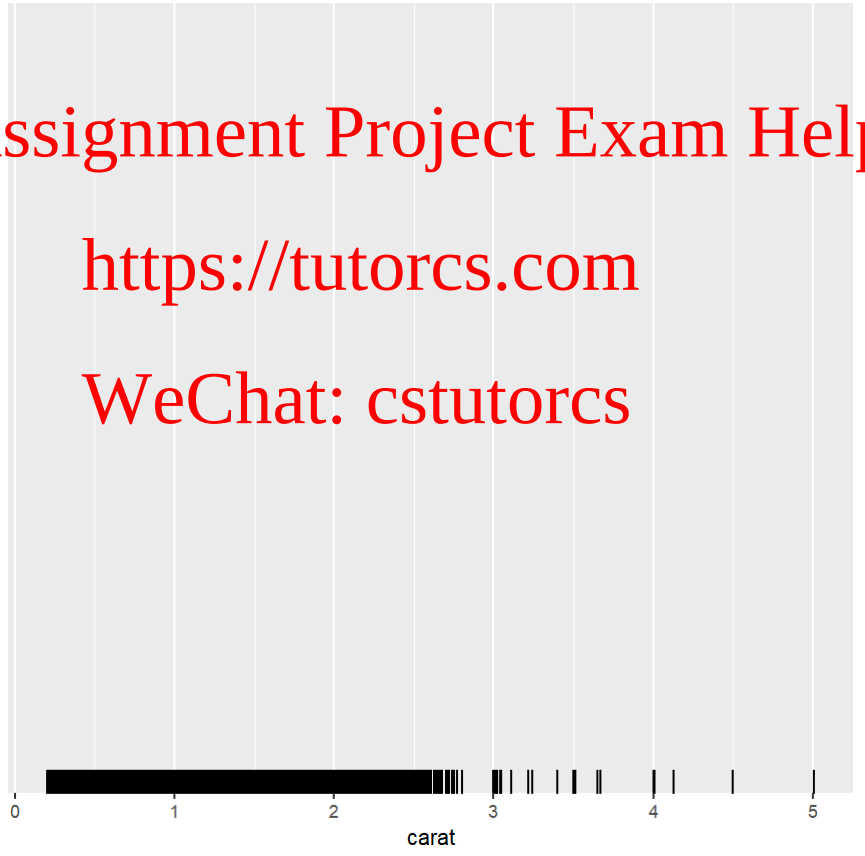
Carat: Histogram

```
ggplot(data = diamonds,mapping = aes(x=carat))+
  geom_histogram()
```



Carat: Rug plot

```
ggplot(data = diamonds,mapping = aes(x=carat))+
  geom_rug()
```

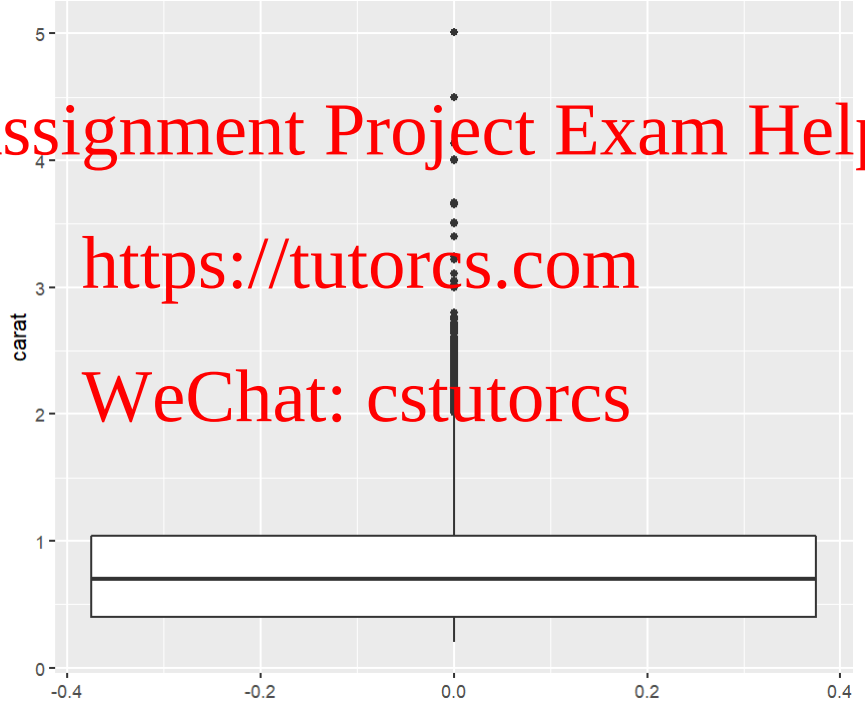



Box plot

- The box plot summarises 5 numbers
  - Median
  - First quartile $Q1$
  - Third quartile $Q3$
  - Upper Fence $U = Q3 + 1.5 \times (Q3 - Q1)$
  - Lower Fence $L = Q1 - 1.5 \times (Q3 - Q1)$
- Anything lying outside the fences represented as dots.
- When no points lie outside the fence, the fence is set to the maximum or minimum.

Carat: Boxplot

```
ggplot(data = diamonds,mapping = aes(y=carat))+
  geom_boxplot()
```

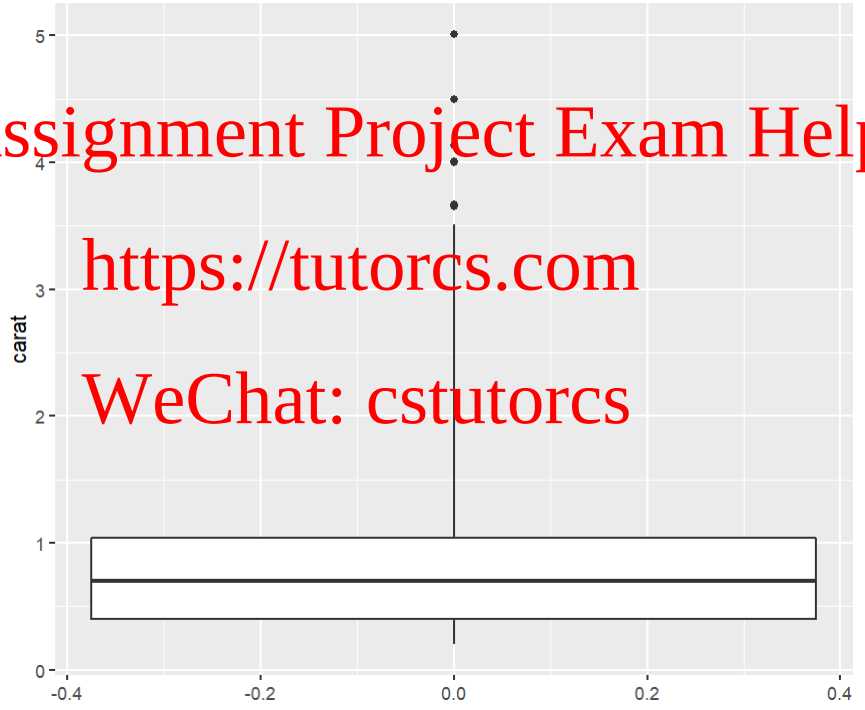


Change of aesthetic

- Notice that the aesthetic changed!
- In the boxplot, the value of the variable is represented by the vertical (or y axis).
- We can change the definition of the upper and lower fence by passing the `coef` argument to `geom_boxplot`.
- This changes the 1.5 used in calculating the fence to whatever you specify

Changing fences

```
ggplot(data = diamonds,mapping = aes(y=carat))+
  geom_boxplot(coef=4)
```

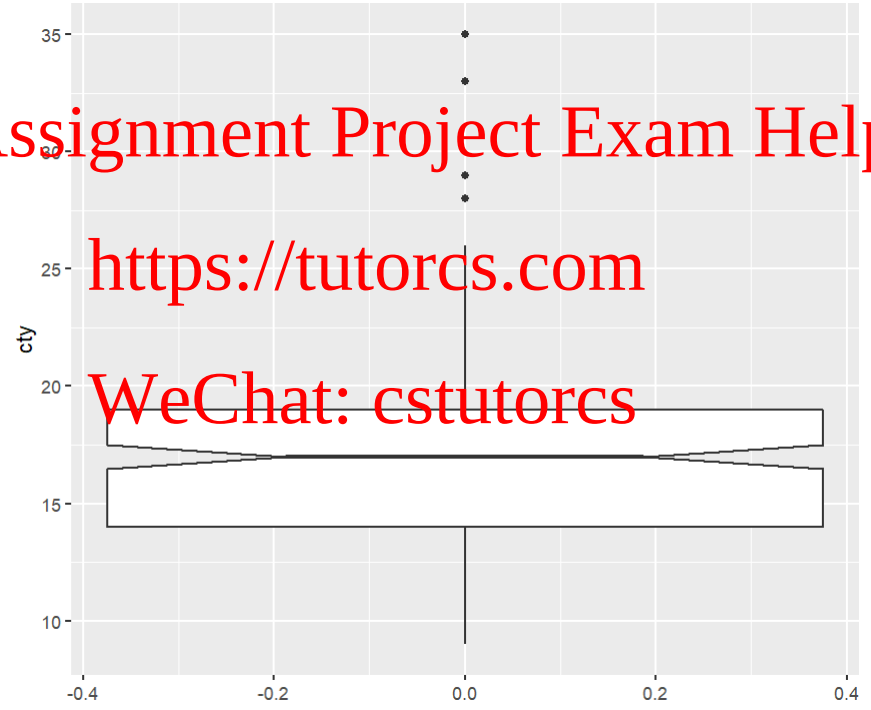


Notches

- Notches can be added to a boxplot
- These are set to $\frac{1.58 \times (Q3 - Q1)}{\sqrt{n}}$
- This roughly gives a 95% confidence interval for the median.
- We will use a smaller dataset on the mileage of cars for this example to clearly illustrate the notches.

Notches

```
ggplot(data = mpg,mapping = aes(y=cty))+
  geom_boxplot(notch = T)
```



One Non-Metric Variable

Nominal v Ordinal

- Non-metric variables are made up of **nominal** and **ordinal** variables.
- Nominal variables have no ordering in the categories of data:
  - Manufacturer of car (Audi, Toyota, etc).
- Ordinal variables do have an ordering in the categories:
  - Quality of diamonds (Fair, Good, etc).

Non-metric variables in R

- Non-metric variables can be stored in R as
  - Character variables (nominal data)
  - Factors (nominal data)
  - Ordered factors (ordinal data)
- You can check with the `str` function

Diamonds data

```
str(diamonds)

## tibble [53,940 x 10] (S3: tbl_df/tbl/data.frame)
##  $ carat  : num [1:53940] 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 .
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ..
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 .
##  $ depth  : num [1:53940] 61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 .
##  $ table  : num [1:53940] 55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int [1:53940] 326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num [1:53940] 3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num [1:53940] 3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 .
##  $ z      : num [1:53940] 2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 .
```

Mpg data

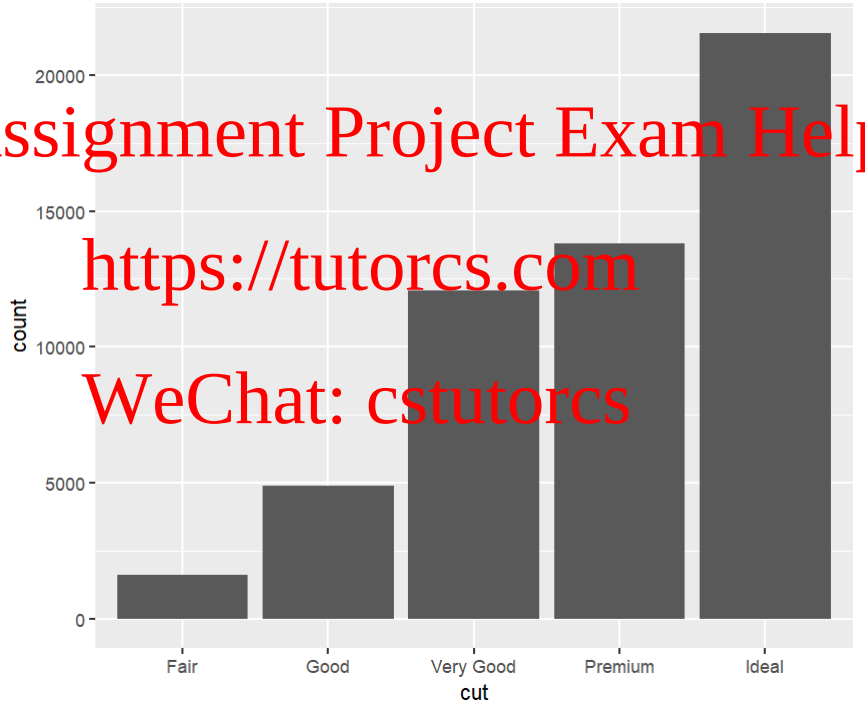
```
str(mpg)

## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 200
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)"
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

Bar plot

- A common plot for non-metric data is the bar plot for the frequency of observations for each level of the factor.
- The height of each bar indicates the number of observations in a particular category.
- This can be done using `geom_bar`

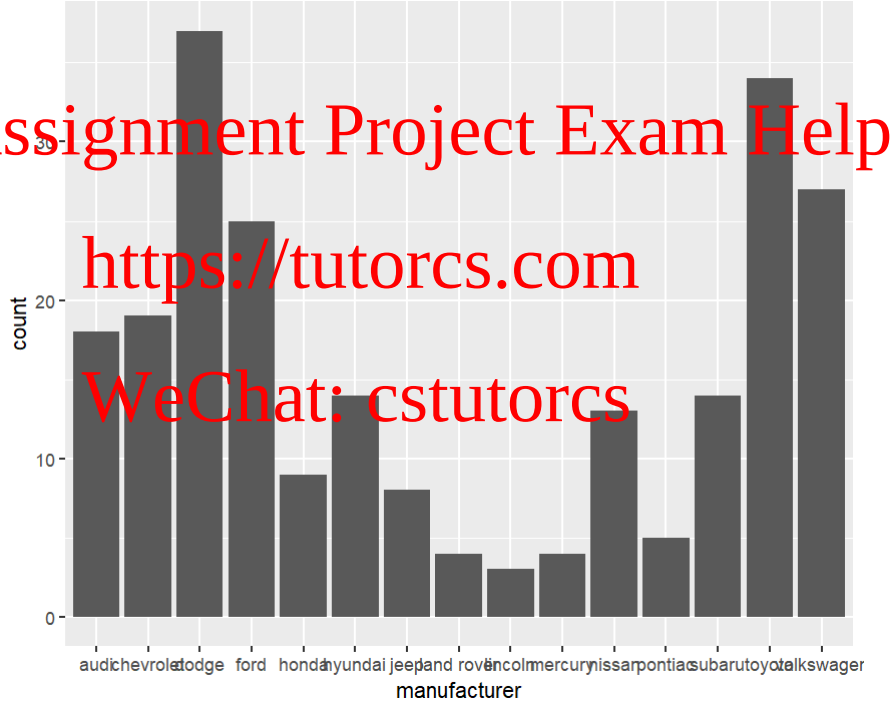
Bar plot

```
ggplot(data = diamonds, mapping = aes(x=cut))+
  geom_bar()
```



Bar plot

```
ggplot(data = mpg, mapping = aes(x=manufacturer))+
  geom_bar()
```



Mosaic plot

- Used to visualize the counts of two discrete variables (can be difficult to read!)
- Need to use an additional package `ggmosaic`

```
library(ggmosaic)
ggplot(data = mpg)+
  geom_mosaic(mapping = aes( x= product(drv,cyl), fill = drv))
```

Two Continuous Variables

What to look for

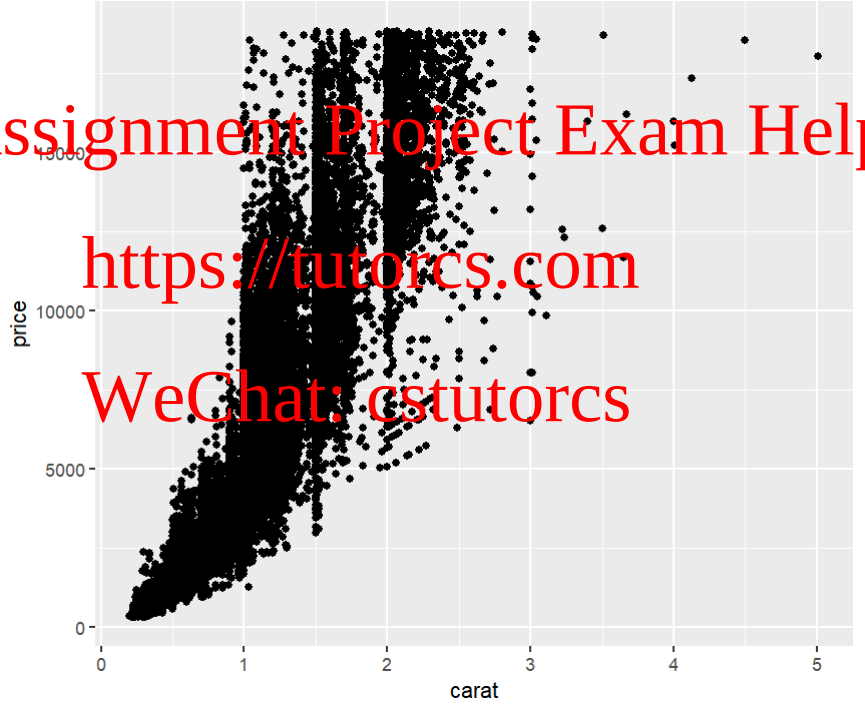
- Outliers
- Dependence or correlation
- Remember that correlation does not imply causation!
- Non linear relationships.

Scatter plot

- For two metric variables use a scatter plot
  - One variable is represented by the `x` aesthetic
  - The other is represented by the `y` aesthetic
  - The geometry we use is `geom_point`.
- We will continue to use the `diamonds` dataset

Scatterplot

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

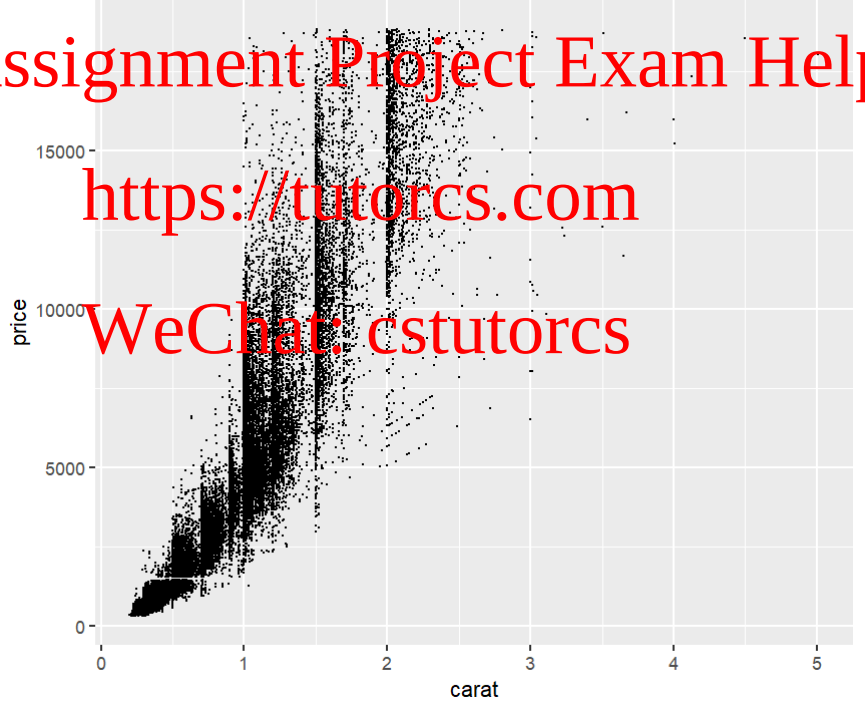


Overplotting

- When using big datasets, sometimes the points cover one another or are too close.
- This is sometimes called **overplotting**.
- Some solutions:
  - Try smaller points (size)
  - Try more transparent points (alpha)
  - Try a different geom

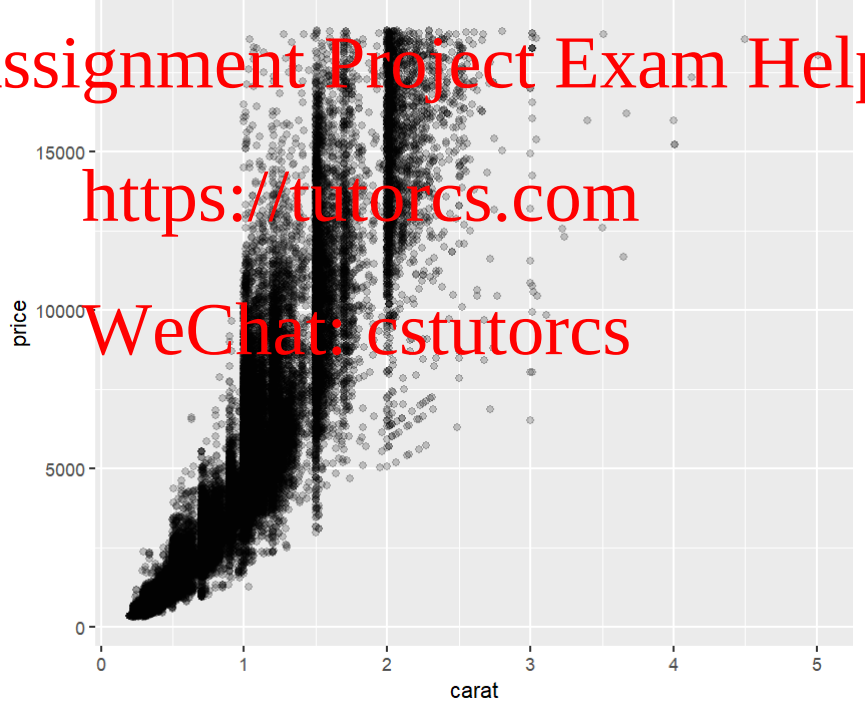
Changing size

```
ggplot(data = diamonds,
       mapping = aes(x=carat,y=price))+
  geom_point(size=0.1)
```

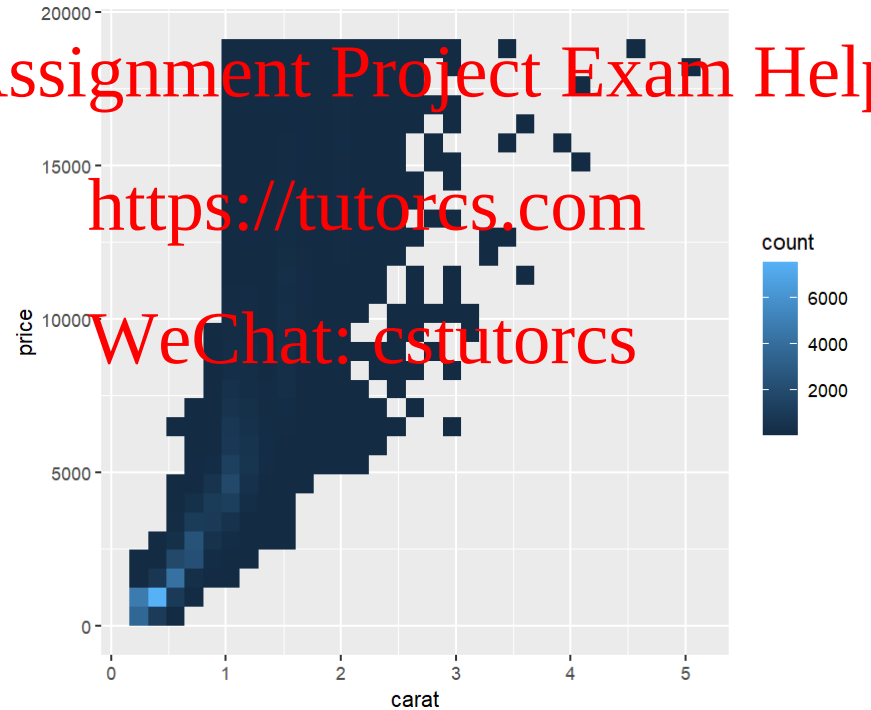
Changing alpha

```
ggplot(data = diamonds,
       mapping = aes(x=carat,y=price))+
  geom_point(alpha=0.2)
```

Changing geom

```
ggplot(data = diamonds,
       mapping = aes(x=carat,y=price))+
  geom_bin2d()
```

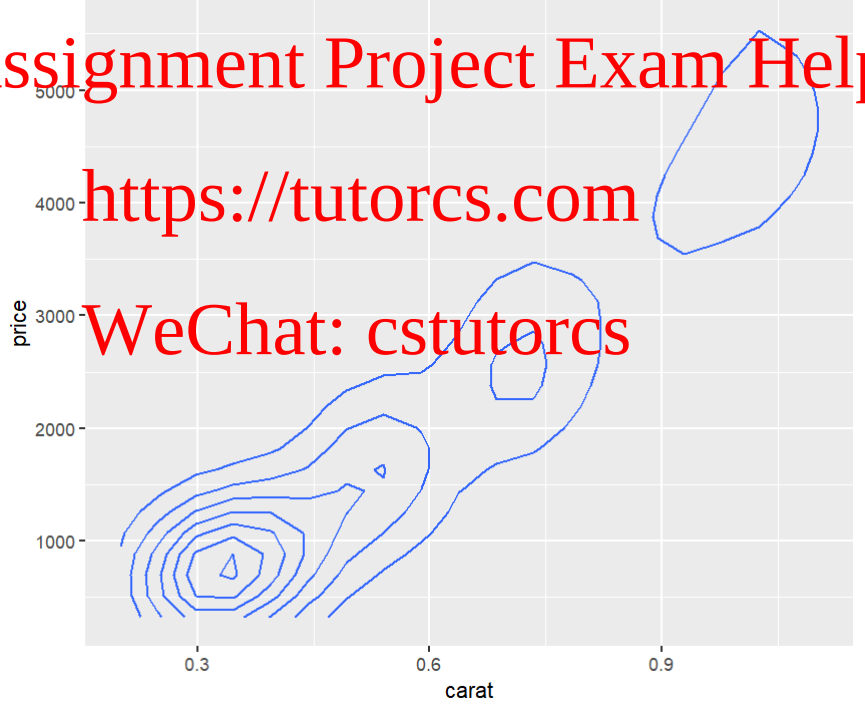


Hexagonal bins

```
ggplot(data = diamonds,
       mapping = aes(x=carat,y=price))+
  geom_hex()
```


Changing geom

```
ggplot(data = diamonds,
       mapping = aes(x=carat,y=price))+
  geom_density2d()
```

Time series plots

- When the x variable is time, it often makes more sense to join dots with a line.
- This way we can see
  - Trend
  - Seasonality
  - Outliers
  - Structural break

Economics dataset

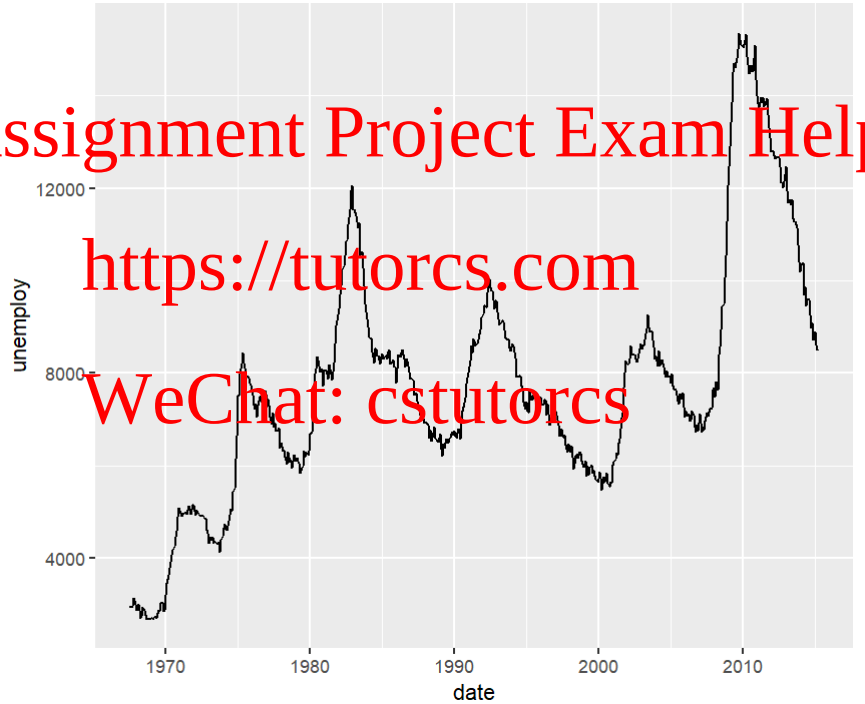
- We will use the economics dataset (comes with ggplot2)

```
str(economics)

## spec_tbl_df [574 x 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ date    : Date[1:574], format: "1967-07-01" "1967-08-01" ...
##  $ pce     : num [1:574] 507 510 516 512 517 ...
##  $ pop     : num [1:574] 198712 198911 199113 199311 199498 ...
##  $ psavert : num [1:574] 12.6 12.6 11.9 12.9 12.8 11.8 11.7 12.3 11.7 12.3 ..
##  $ uempmed : num [1:574] 4.5 4.7 4.6 4.9 4.7 4.8 5.1 4.5 4.1 4.6 ...
##  $ unemploy: num [1:574] 2944 2945 2958 3143 3066 ...
```


- Notice date is its own type of variable

Unemployed persons

```
ggplot(economics, aes(x=date, y=unemploy))+
  geom_line()
```



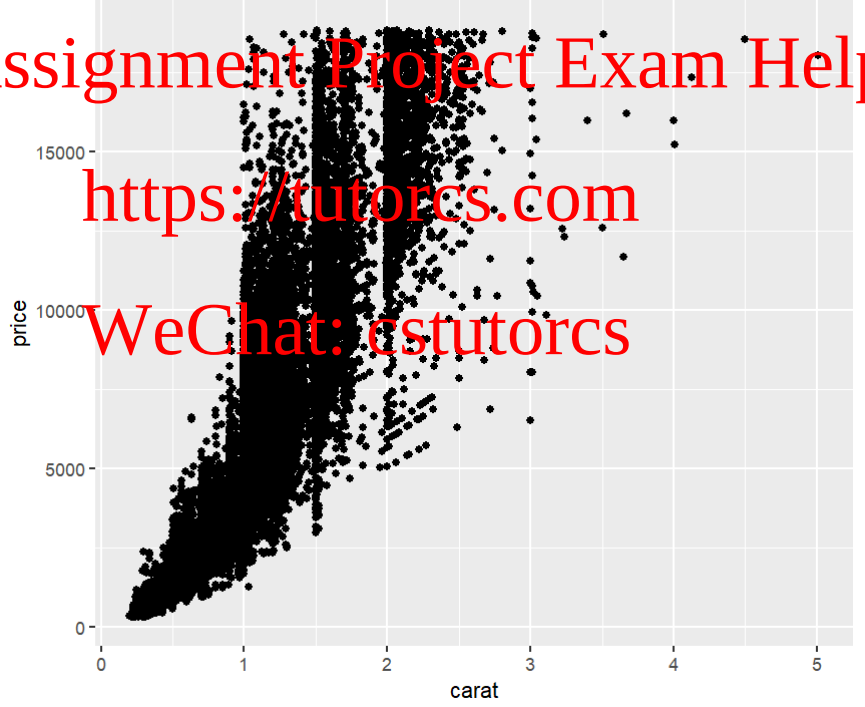
An aside on log scales

Scale

- For variables that are heavily skewed it can be better to look at a log scale.
- For a regular scale you *add* as you move up the scale.
- For a log scale you *multiply* as you move up the scale.
- The log scale has the effect of putting more distance between smaller values and compressing higher values.

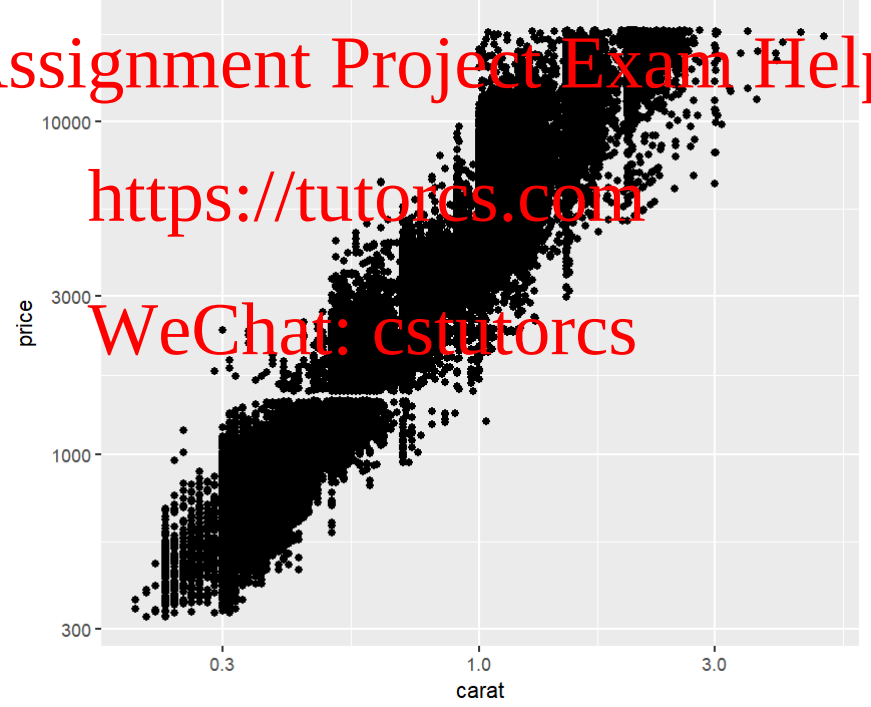
Regular scale

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

Log scale

```
ggplot(data = diamonds,
       mapping = aes(x=carat,y=price))+
  geom_point()+scale_x_log10()+scale_y_log10()
```

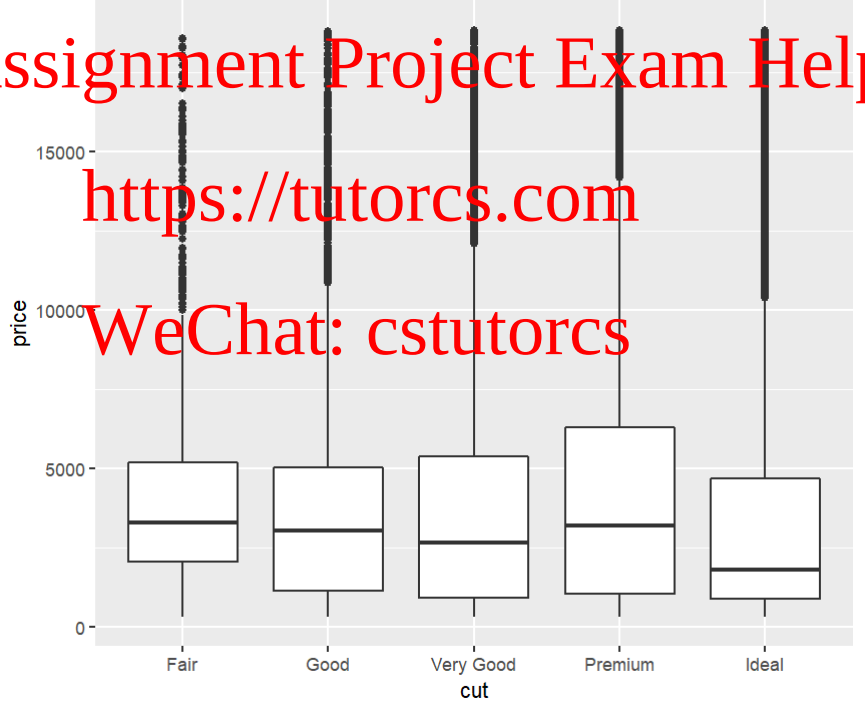
Metric and Non-Metric Data

Side by side plots

- When one variable is metric and the other non-metric we can easily put plots next to one another side by side.
- Simply map the non-metric variable to the x aesthetic and the metric variable to the y aesthetic.

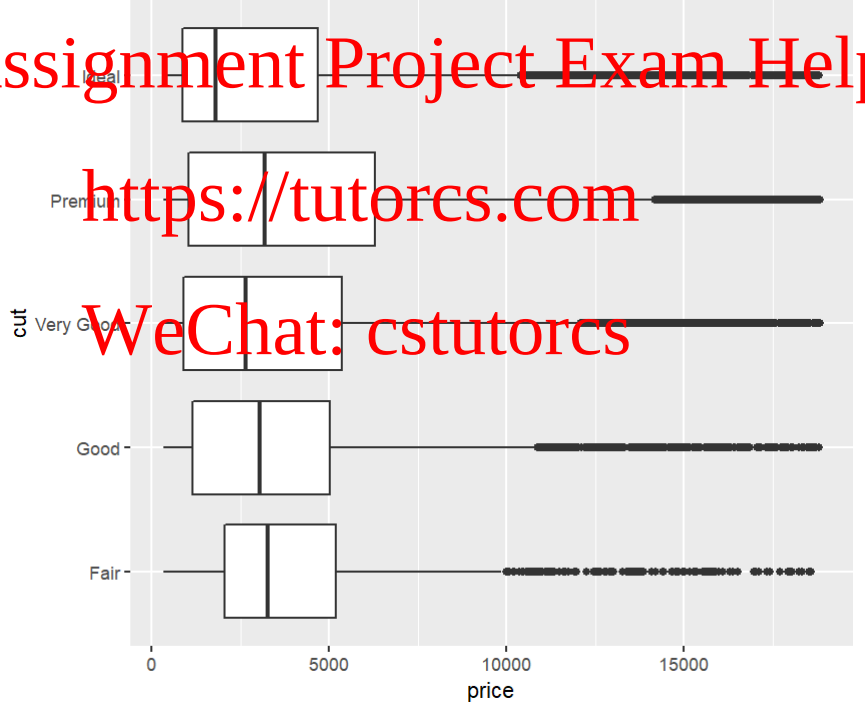
Boxplots

```
ggplot(data = diamonds,
       mapping = aes(x=cut,y=price))+
  geom_boxplot()
```

Change axes

```
ggplot(data = diamonds,
       mapping = aes(x=price,y=cut))+
  geom_boxplot()
```

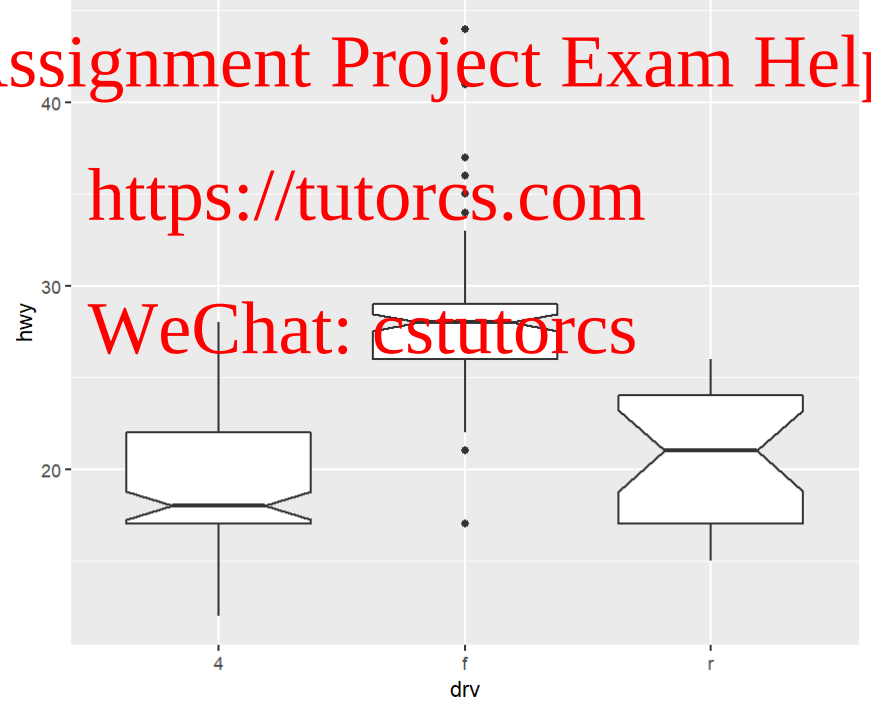


With notches

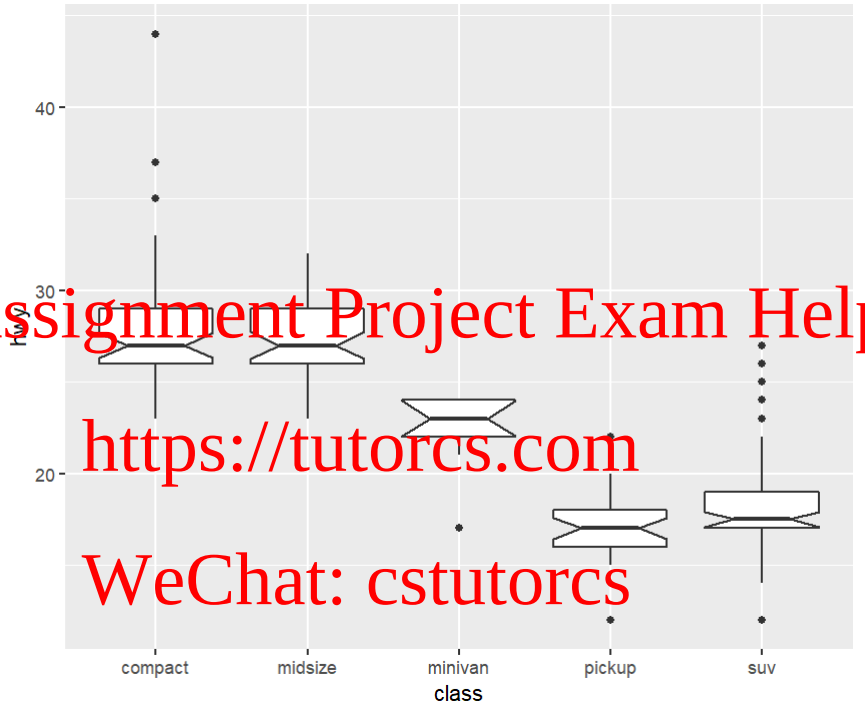
- Recall that the notches provide a confidence interval around the median.
- These are particularly useful when comparing boxplots to one another.
- In general, if the confidence intervals overlap then the medians are not signficantly different.
- This is NOT a formal test, but still gives a useful indication.

Boxplots (no overlap)

```
ggplot(data = mpg,
       mapping = aes(x=drv,y=hwy))+
  geom_boxplot(notch=T)
```

Boxplots (some overlap)

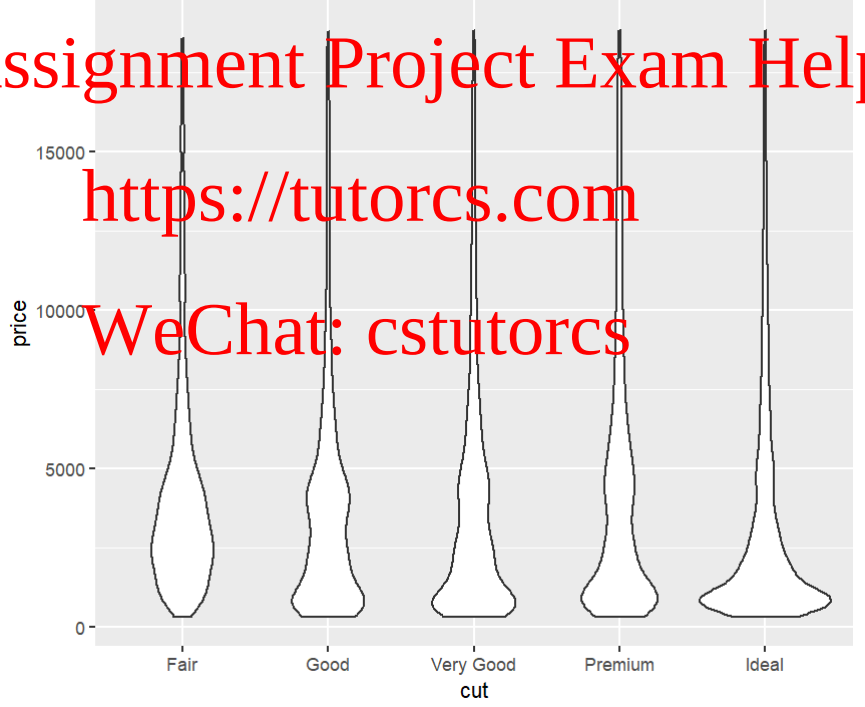


Violin plot

- A violin plot is a newer visualisation.
- A kernel density is mirrored then arranged vertically.
- Specify the same way but use *geom_violin*

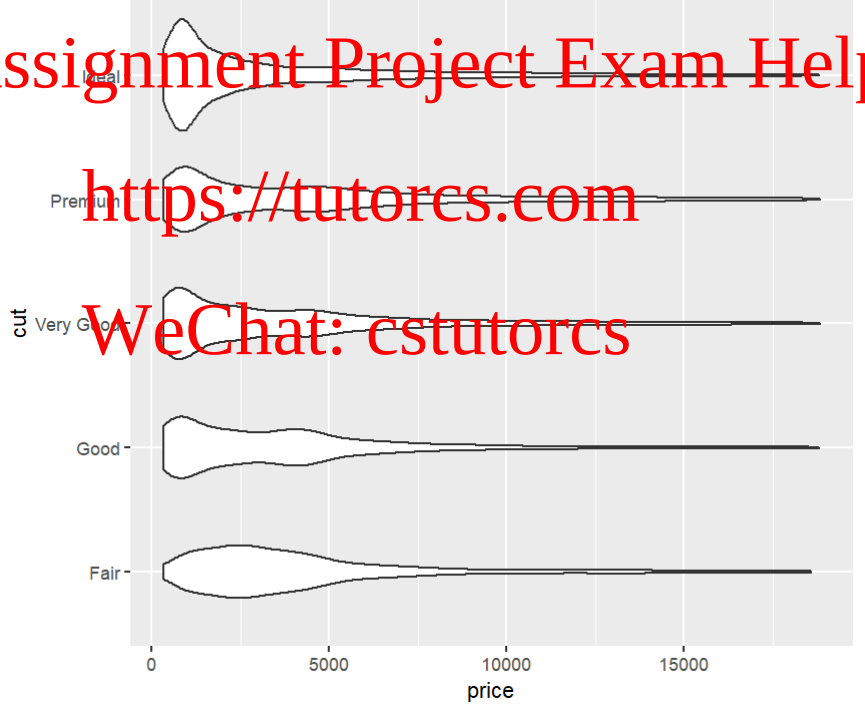
Violin plot

```
ggplot(data = diamonds,
       mapping = aes(x=cut,y=price))+
  geom_violin()
```

Violin plot

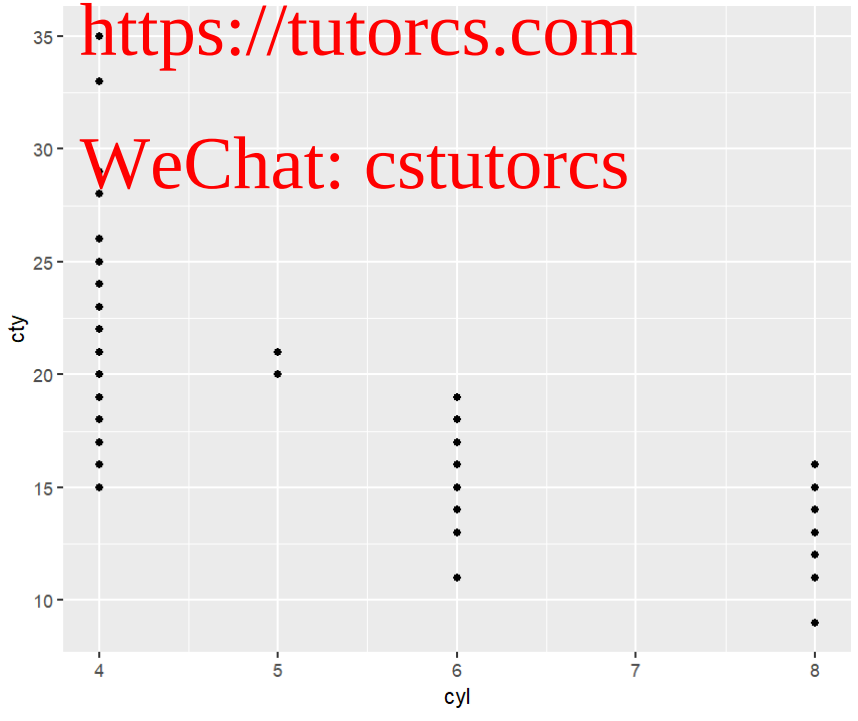
```
ggplot(data = diamonds,
       mapping = aes(x=cut,y=price))+
  geom_violin()+coord_flip()
```



Jittering

- A scatter plot can be used for non-metric data but can easily suffer from overplotting (one point on another).

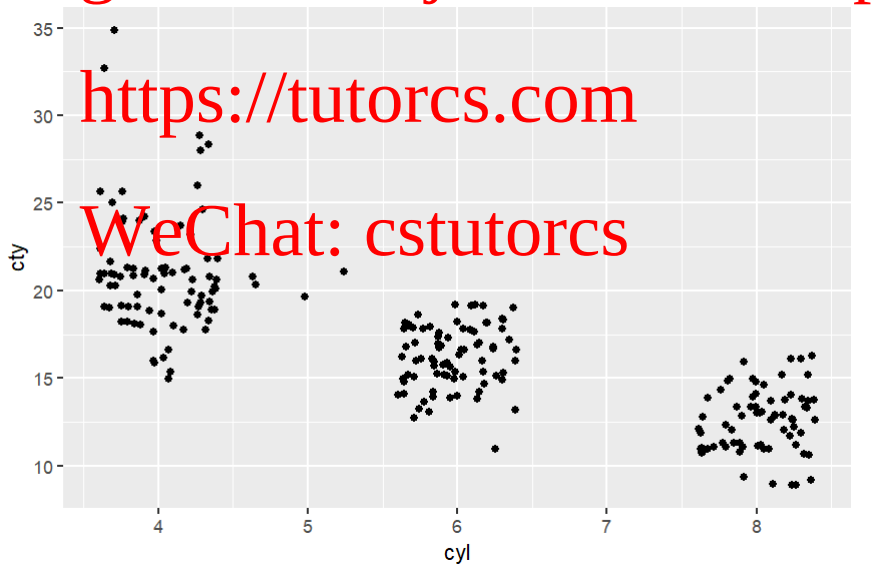
```
ggplot(data = mpg,
       mapping = aes(x=cyl,y=cty))+
  geom_point()
```

Jittering

- Add random noise by jittering

```
ggplot(data = mpg,
       mapping = aes(x=cyl,y=cty))+
  geom_point(position = 'jitter')
```

MONASH University

Lecturer: *Lauren Kennedy*

Department of Econometrics and Business Statistics

lauren.kennedy1@monash.edu

Week 2