Data Visualization

Notes for CS 6232: Data Analysis and Visualization
Georgia Tech (Dr. Guy Lebanon), Fall 2016
as recorded by Brent Wagenseller

Lesson Preview

Learn how to use base graphics
Learn how to use ggplot2
Understand basic graph types and when to use them

Base Graphics

Base Graphics is a package in R that allows for simple graphs
Its installed by default
Example
plot(x=dataframe$col1, y=dataframe$col2)
title(main=”figure title”) #add title
Base graphics have several low-level functions / helper functions

- ‘title’ adds or modifies labels of title and axes
- ‘grid’ function adds a grid
- ‘legend’ displays a legend connecting symbols, colors, and line-types to descriptive strings
- ‘lines’ adds a line plot to an existing graph

GGPlot2

ggplot2 is a popular visualization package with a different philosophy than base graphics
Advantages

- It implements the grammar of graphics
  - The **grammar of graphics** is a way to structure and define different graphics elements using a basic vocabulary
- It enables **logical separation** of graphics and data
  - It has its own variables so you can easily graph the data again later
  - More effort is spent on creating the data frame and less on graphing it
  - The effort is shifted from graphing into crating an appropriate data frame
- Its concise and maintainable

Options on how to use

- Use the qplot function
  - Simple
  - Pass the data frame name, data frame column names, geometry, and graphing options
  - Example
    qplot(x=x1, y=x2, data=myDataFrame, main=”figure title”, geom=”point”)
        “point” is for a scatterplot
- Use ggplot function
  - More complicated
  - The data frame and column names must be passed through the ‘aes’ function

Both qplot and ggplot can compose the function output with additional layers using the plus (+) operator

- qplot and ggplot return an object
- If you have multiple such objects, you can compose them with a figure with multiple objects in it

The qplot and ggplot functions (and addition separator) returns an object that can be printed or saved for later

- We don’t have to re-create the object every time we want to re-create the graph

Strip Plots

Strip plots display a vector of values with the x axis corresponding to the index and the y value corresponding to that component

Example

```
library(ggplot2)
data(faithful)
plot(faithful$eruptions, xlab="sample number", ylab="eruption times (min)", main = "Old Faithful Eruption Times")
```

Histograms

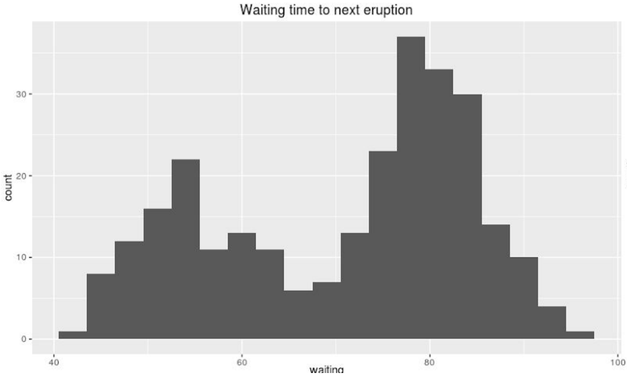
Histograms graph one-dimensional numeric data by dividing the range into bins and counting number of occurrences in each bin

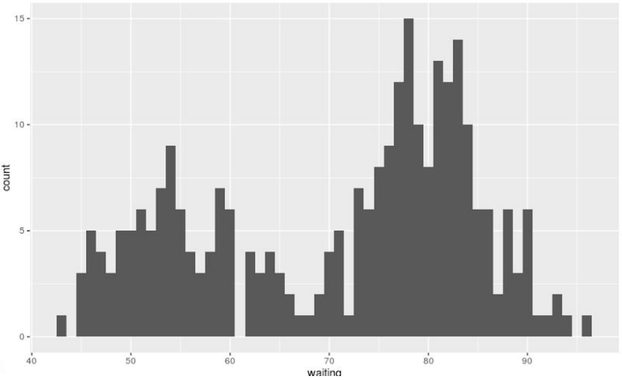
The bin width value MUST be set correctly!

The Y axis is typically the count, but it can be replaced with the probability / frequency (converting it into a probability mass function)

Example

```
qplot(x=waiting, data=faithful, binwidth=3, main="Waiting time to next eruption")
ggplot(faithful, aes(x=waiting)) + geom_histogram(binwidth=1)
```





- The first example uses qplot and the second uses ggplot
- The range of both is the same, but qplot uses a binwidth=3 and ggplot uses a binwidth=1
  - Notice how the ggplot is more granular – this is due to the more granular binwidth!
  - If qplot used binwidth=1 it would be the same histogram
- ggplot example
  - note the use of aes() – recall this is how you can set data is through this function
    You can add more data / lines with additional aes() calls – just add it at the end with a plus
  - Also note that you can save the entire ggplot to an object like so:
    `mySavedGGPlot <- ggplot(faithful, aes(x=waiting))`
  - And then you can add more later (with the plus (+) operator) like so:
    `mySavedGGPlot <- mySavedGGPlot + geom_histogram(binwidth=1)`

Recall that the Y can be replaced with the probability / frequency (converting it into a **probability mass function**), with the area under that function = 1

- In other words: the high of the bin times the width of the bin is the probability of getting values within that range
- To do this in ggplot, assign the y (example):
  `ggplot(faithful, aes(x=waiting, y= ..density..)) + geom_histogram(binwidth=1)`

As stated before, the bin size must be selected carefully

- If the bin is too big its difficult to pick out patterns

- If the bin is too small, you will see too much noise that will add confusion to the histogram

Line Plots

A line plot is a graph displaying a relation between x and y as a line in a Cartesian coordinate system

- In other words, a plot that shows y values as a function of x values
- The relation may correspond to an abstract mathematical function or to a relation between two samples (for example, dataframe columns)

A line plot is very common

Example

```
###Line plots
data(mtcars)
#create a vector that contains values between -2 and 2 and have 30 values exually spaced
x<- seq(-2,2,length.out = 30)
y<-x^2

#create line plot using qplot
qplot(x, y, geom = "line")

#create a line and point plot
#same as above but put in the points
qplot(x, y, geom= c("point", "line"))

#now do the same thing but use ggplot
myData <- data.frame(x=x, y=y)
ggplot(myData, aes(x=x, y=y)) + geom_line() + geom_point()

#same thing, yet save the ggplot to an object that can be called later
mySavedGGPlot <-ggplot(myData, aes(x=x, y=y)) + geom_line()
mySavedGGPlot<-mySavedGGPlot + geom_point()
mySavedGGPlot
```

- The geom_line() function returns the line geometry

Another Example

```
#more with line plots
data(mpg)#x = city mpg, ix = indices of sorted values of city mpg
#when using sort.int, it stores the values in x - sorted - and the original indexes in ix
S <- sort.int(mpg$cty, index.return = T)
#plot sorted city MPG values with a line plot using the base graphics package
#type = "l" means 'show a line plot'
#lty = 2 means 'what type of line' - in this case 2 is dashed
plot(S$x, type = "l", lty=2, xlab = "sample number (sorted by city mpg)", ylab="mpg")
lines(mpg$hwy[S$ix], lty = 1) #add an additional line - a solid line of highway mpg.  Note we must order by S$ix
legend("topleft", c("highway mpg", "city mpg"), lty = c(1,2))#create a legend, place it on the screen, and label it
```

Smoothed Histograms

A **smoothed histogram** is similar to a histogram, but instead of showing bins it shows a curve

- Mathematically:

$$f_h: \mathbb{R} \to \mathbb{R}_+ \qquad f_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x^{(i)})$$

- Measurements are denoted by $x^{(1)}$ .. $x^{(n)}$
- Where the Kernel function $K_h: \mathbb{R} \to \mathbb{R}$ typically achieves its maximum at 0, and decreases as $|x-x^{(i)}|$ increases. We also assume that the kernel function integrates to one $\int K_h(x)dx=1$ and satisfies the relation $K_h(r) = h^{-1}K_1(r/h)$.

  The h value is also called the bandwidth, which is similar to binwidth in the traditional histogram

  We refer to K_1 as the base form of the kernel and denote it as K.
- Explained

  We take each data point $x^{(1)}$ .. $x^{(n)}$ and replace it with the kernel function that is centered at that data point and we take the average of all of those functions

Selecting the kernel function is very important (much like the bin width of the normal histogram)

Selecting a value h is also important

- too large will cause over-smoothing, causing the resulting f ˆ to be nearly constant.
- too small will have an under-smoothing effect, resulting in a wiggly and noisy curve.

Kernel choices

- Popular choices
  - Tricube

    Smooth bump
  - Triangular

    Triangle
  - Uniform

    Looks like a bump

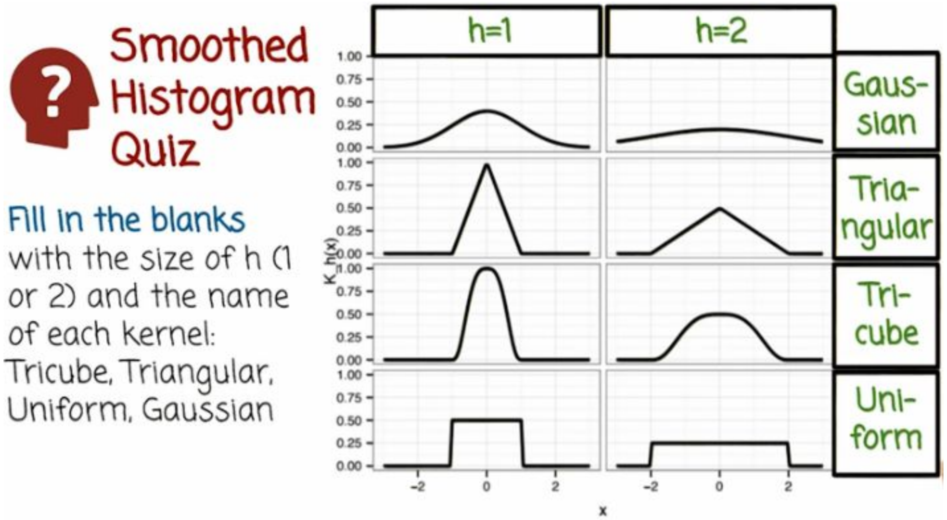
    Equals 0 except for an area around 0 where it =1
  - Gaussian

    Probability Gaussian Density
- $$K(r) = (1-|r|^3)^3 \cdot 1_{\{|r|<1\}} \quad \text{(Tricube)}$$
  $$K(r) = (1-|r|) \cdot 1_{\{|r|<1\}} \quad \text{(Triangular)}$$
  $$K(r) = 2^{-1} \cdot 1_{\{|r|<1\}} \quad \text{(Uniform)}$$
  $$K(r) = \exp(-x^2/2)/\sqrt{2\pi} \quad \text{(Gaussian).}$$
  - The notation corresponds to a function that is 1 if the condition inside the curly brace holds, 0 otherwise
  - For example, the uniform kernel is 2^-1 (which is ½) * 1 if the absolute value is between -1 and 1; at this point it will be ½

    The uniform kernel is 0 everywhere except for
- Defined as satisfying $K_h(r)=x^{-1}K(r/h)$ where the K(*) functions are respectively
- As h increases (for all kernels), the kernel functions K_h become wider
  - This explains why we call h the bandwidth
  - Smaller h will create a smooth histogram that has a lot of noise

    - A large h may create a smoothed histogram that is very smooth, but it may hide important details in the data

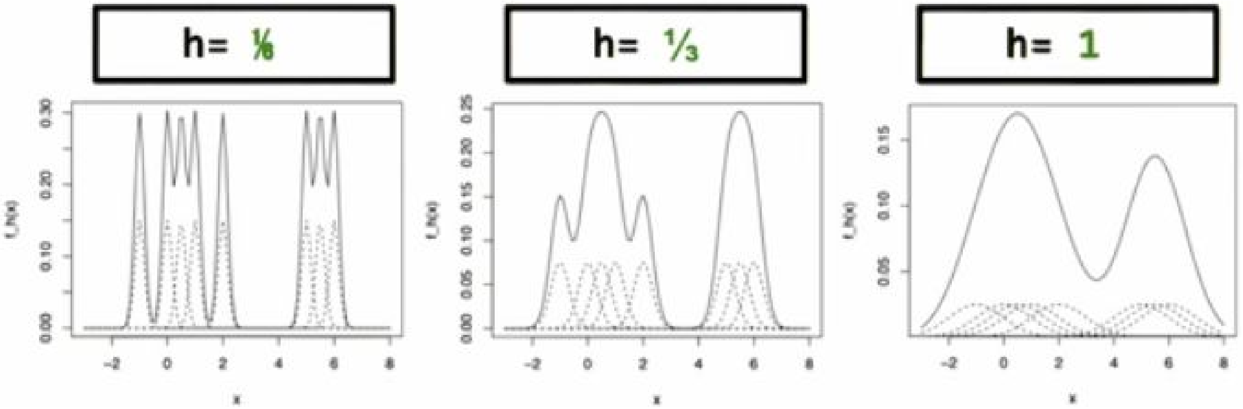
Example



Smoothed Histogram Quiz

Fill in the blanks with the size of h (1 or 2) and the name of each kernel: Tricube, Triangular, Uniform, Gaussian

Varied H Quiz

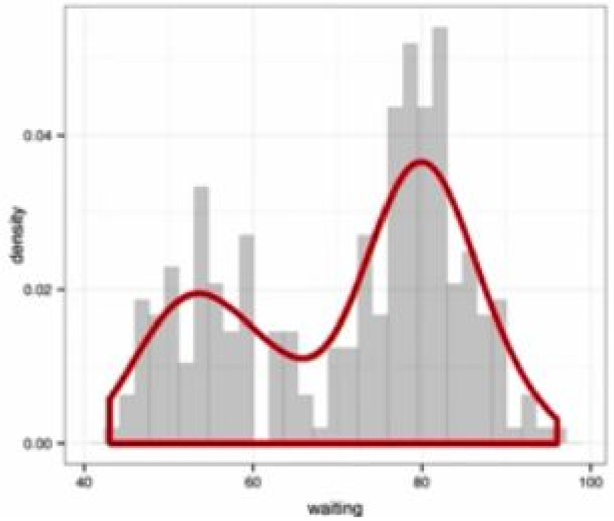


- Note that this is the same data set, yet a different h in a kernel function
- Each of the data points is replaced by a kernel function denoted using the dashed lines
    - These are then added up and the result is divided by the number of data points

R code for a smooth histogram

- Code

```
##Plot a smooth histogram over top a traditional histogram
#the geom_density() puts in a smooth histogram
#size=2 - the red line thickness
#alpha=0.3 means make the histogram slightly transparent
ggplot(faithful, aes(x=waiting, y=..density..)) +
  geom_histogram(alpha=0.3) + geom_density(size=2, color='red')
```



- The traditional histogram is faded a bit (alpha=.3) because we don't want it to be too dark

Comparing to the traditional histogram

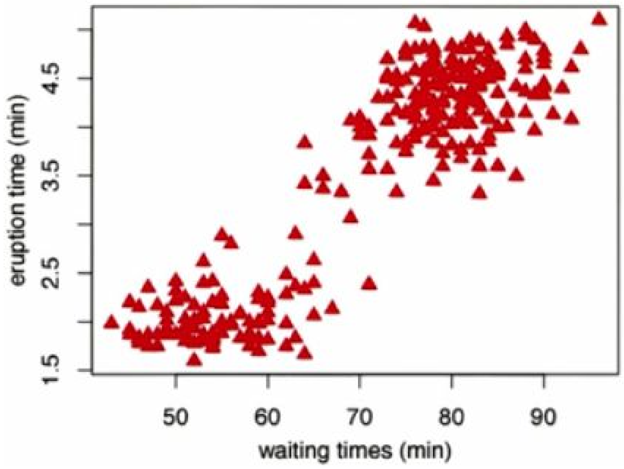
- A smoothed histogram can sometimes can be more useful than a histogram as it is a better approximation of the underlying density
- smoothed histograms do a better job of uncovering the mathematical relationship between the variable and the frequency of density
    - This is backed mathematically - the smoothed histogram is a better estimator for the underlying density than the histogram.
- In contrast, histograms features a more direct relationship between the graph and the underlying data that is sometimes important in its own right.

Scatter Plots

A **scatter plot** graphs the relationships between two numeric variables. It graphs each par of variables as a point in a two dimensional space whose coordinates are the corresponding x, y values

Code

```
##Plot a scatter plot using the bas graphics
plot(faithful$waiting, faithful$eruptions,
    pch=17, #type of marker
    col=2, #color
    cex=1.2, #size
    xlab="Waiting Times (min)", ylab = "Eruption Time (min)")
```



- This represents 'old faithful' eruption times

- These two clusters make sense, as the longer it waits to erupt the longer it will take to finish

Variable Relationships and Scatter Plots

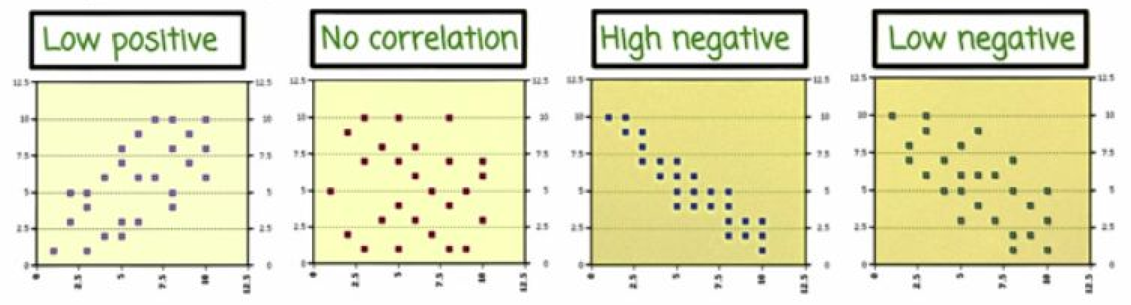
Thus far we have had graphs that describe 1 dimensional data or 2 dimensional data

- 1D data can be graphed with a strip plot or a histogram
- 2D data can be visualized using a scatter plot or a line plot

The relationship between two numeric variable and a 3rd categorical variable controls the size, color, or shape of the marker

- “Categorical” means a non-number – something that cannot be ordered

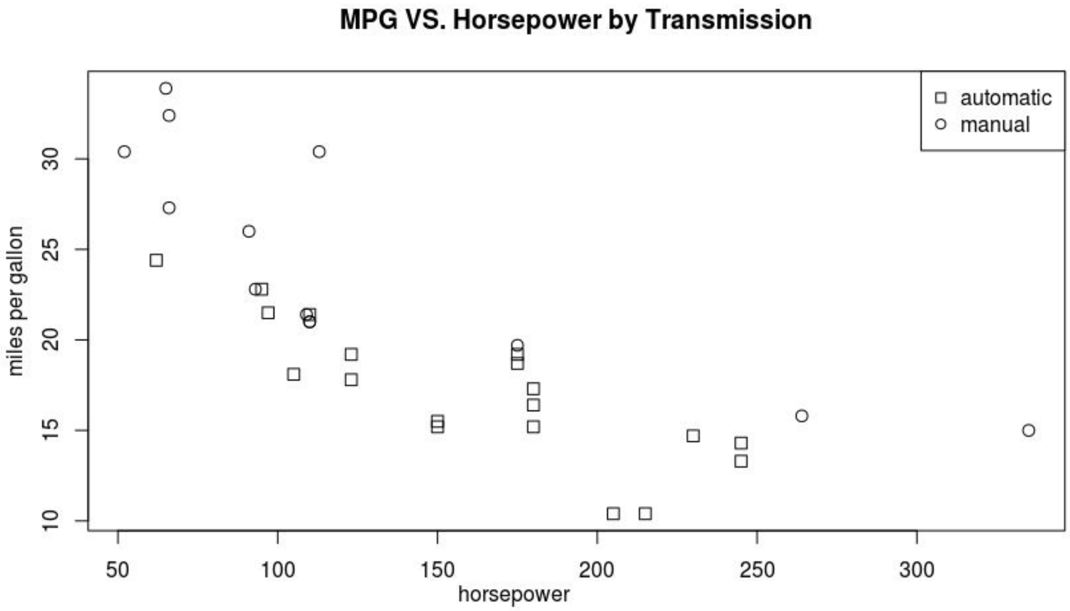
Scatter plots often reveal a correlation between two variables



- The ‘tighter’ the plot, the higher the correlation
- For a positive correlation, one variable increases as the other increases
- For a negative correlation, one variable increases as the other decreases
- Low correlation means there is a correlation but its noisy
- High correlation means its crystal clear that a correlation exists

Code

```
#show the relationship between two numeric variables plus a categorical variable
#the example shows horsepower vs miles per gallon, with the transmission as the categorical variable
plot(mtcars$hp, mtcars$mpg,
    pch = mtcars$am, #notice the slick use of the category (represented as a number) to set the type of marker
    cex=1.2, #size
    xlab="horsepower",
    ylab = "miles per gallon",
    main = "MPG VS. Horsepower by Transmission")
legend("topright", c("automatic","manual"), pch = c(0,1))#set the name and type of marker (0 or 1)
```



- There is an inverse relationship between horsepower and mpg)the bigger the horsepower, the lower the mpg)
- For a given horsepower amount, manual transmission care are generally more fuel efficient
- Cars with the highest horsepower tend to be manual

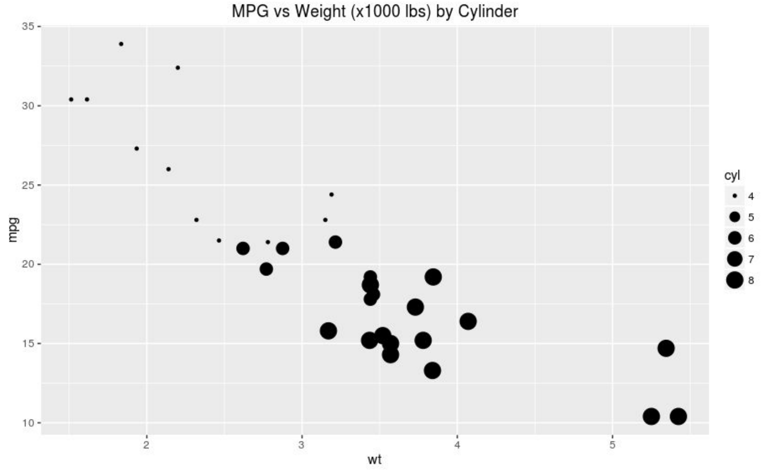
Multivariate Plots

We are going to use the mtcars dataset

We are going to change the marker size to reveal a relationship between three different numeric variables in a scatterplot

Code

```
#we are going to attempt to visually indicate the relationship of 3 separate data features across a 2D graph
#to do this we are going to use the marker size to indicate the 3rd feature - the cylinder
qplot(x=wt, y=mpg, data = mtcars, size=cyl, main = "MPG vs Weight (x1000 lbs) by Cylinder")
```



- According to the graph
  - As the cylinder count increases the mpg decreases (inverse relationship)
  - As the cylinder count increases the weight increases
  - As the weight increases the mpg decreases (inverse relationship)

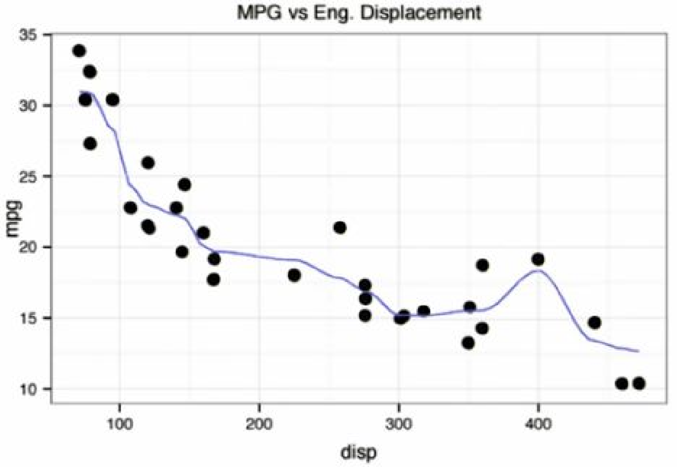
Noisy Data

When data is noisy, its useful to add a smooth line curve to visualize median trends

- For example, when we plot mpg against displacement or the size of the engine, its hard to see trends beyond a general trend of inverse relationship

One thing that can be done is to add a smooth line curve that shows a finer relationship

- For the example, a possible increase around 400 which may be due to noise or may be legitimate



- Sometimes its hard to differentiate between noise and a legitimate change in the trend without more data

Adding a smooth line curve y_s, which is a weighted average of the original data ($y^{(i)}$, $x^{(i)}$) where i = 1 to n:

- Assumes there are n pairs of values
- We wish to add a smooth line to these n pairs
- Equation

$$ys(x)=\sum_{i=1}^{n} \frac{K_h(x-x^{(i)})}{\sum_{i=1}^{n} K_h(x-x^{(i)})} y^{(i)}$$

  - Where the K h functions above are the kernel functions described earlier

    $y_s(x)$ = weighted average of $y^{(i)}$ values

    The denominator ensures the weights sum to 1
  - This is a function that maps every value on the real line to another value on the real line
  - Its defined as a weighted average of the y values where the weights are given by this ratio
  - The ratio has a numerator and a denominator

    The numerator has kernel functions (like we saw in the smooth histograms)

    The denominator is a normalization coefficient that makes sure the numerators all sum to 1

    If we did not normalize, the weights would not sum to 1, the smooth line would be either inflated or deflated

    The smooth line is the weighted average of the y values where the weights are higher for y values that are closer to the current position we are evaluating

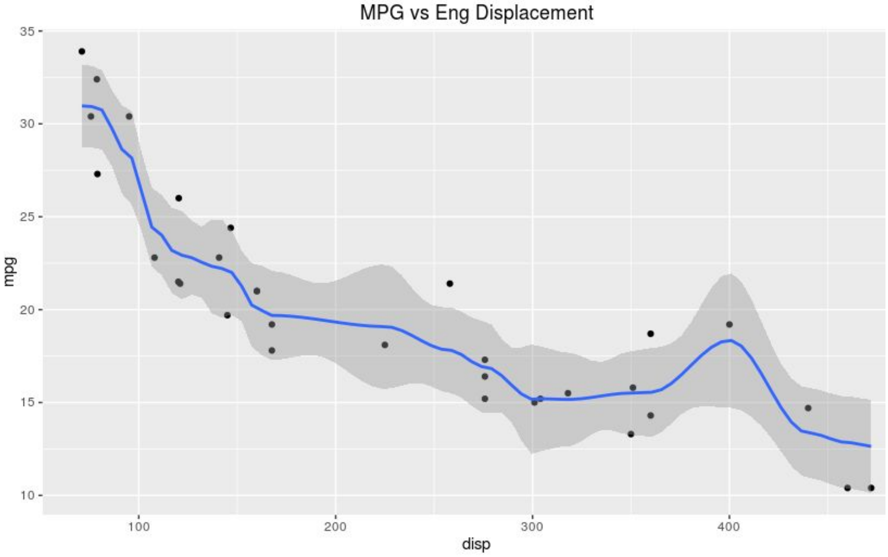
    - Since closer points have a higher weight, this is known as a **local average**
      - This specific model (equation above) is known as **local regression**

        Also **locally constant regression** or **IO Watson regression**

The value of the line – or the y coordinate corresponding to a specific x coordinate is determined by the average of the y coordinate of the points that are nearby to where you are evaluating it (local average)

Code

```
qplot(disp, mpg, data=mtcars,main="MPG vs Eng Displacement") +
stat_smooth(
  method="loess", #technique - maybe Kernel function?
  method.args=list(degree=0), #degree is a parameter associated with the local smoothing
          #operation
    #0 here corresponds relatively closely to the mathematical definition we saw in the
          #equation to locally constant regression
  #degree=0, #this is the method for setting degree as defined in the example but it throws
          #an error - use the above instead
  span=.2, #related to the bandwitdh we saw in kernel smoothing – aka ‘h’; the larger this is,
the more
          #the line smooths out; omitting this line entirely lets R pick a value it thinks will be best
  se=TRUE #standard errors - setting this to TRUE graphs the standard error or confidence
bounds around that line
  )
```



Facets

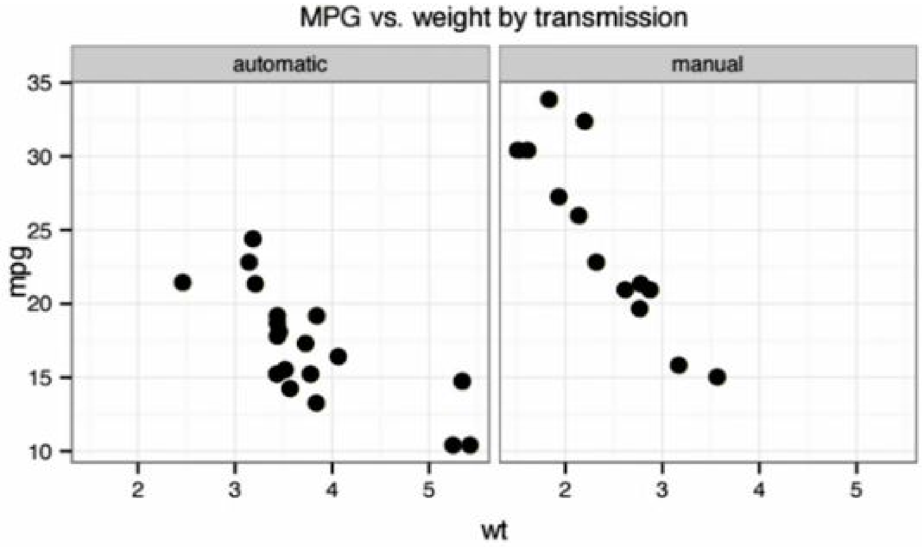
Facets allow us to visualize more than two dimensions

- Similar to the technique we used to plot 3 concepts earlier in the lesson

Facets have two separate panels

- The two panels are aligned perfectly
- Each have same axis concepts and ranges
- Each have a unique 3rd concept that is clearly labeled in the title

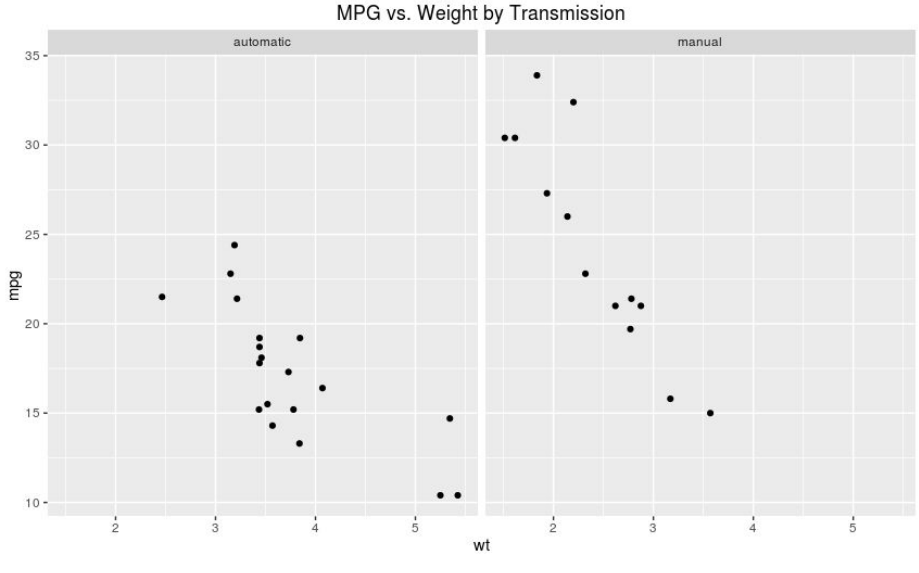
Example



- The argument ‘facets’ in qplot or ggplot takes a formula a~b where a and b specify the variables according to which way the column and rows are organized
  - For our example, we are going to modify the mtcars dataframe to have new columns with more appropriate names for better axes labeling
    - Two new columns: amf and vsf

Code – One pair of facets side by side

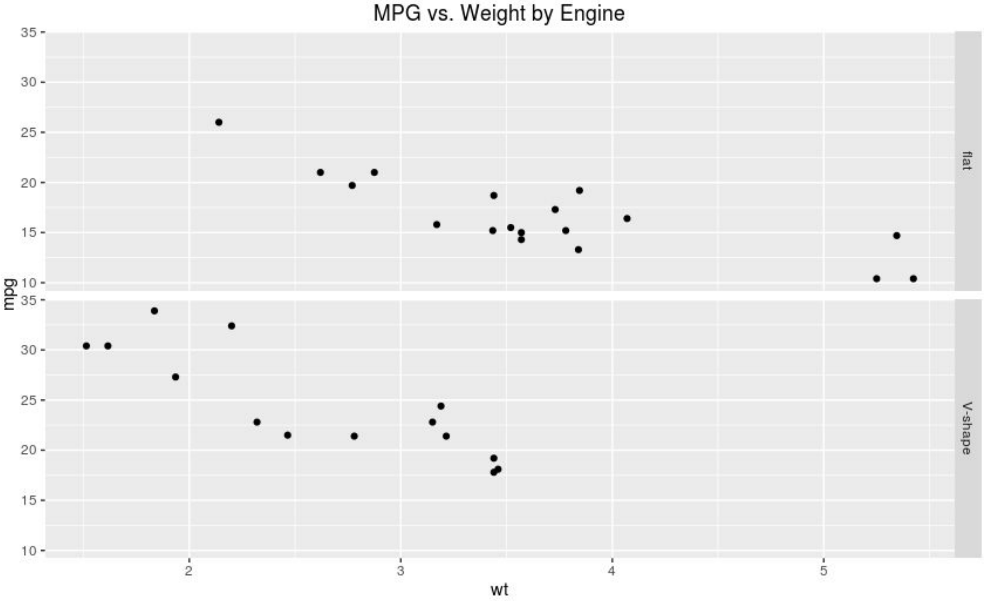
```
#modify mtcars to add two new facets: amf (transimssion type) and vsf (shape of the engine)
mtcars$amf[mtcars$am==0] = 'automatic'
mtcars$amf[mtcars$am==1] = 'manual'
mtcars$vsf[mtcars$vs==0] = 'flat'
mtcars$vsf[mtcars$vs==1] = 'V-shape'

qplot(x=wt, y=mpg,
    facets=.~amf,
    data=mtcars, main="MPG vs. Weight by Transmission")
```



- The argument we need to pass to create the facet is a formula with a tilde, with arguments on either side
- This formula tells qplot to only have one row due to the period to the left of the tilde.
- note that the dot simply means 'blank', as in 'only one pair of facets'

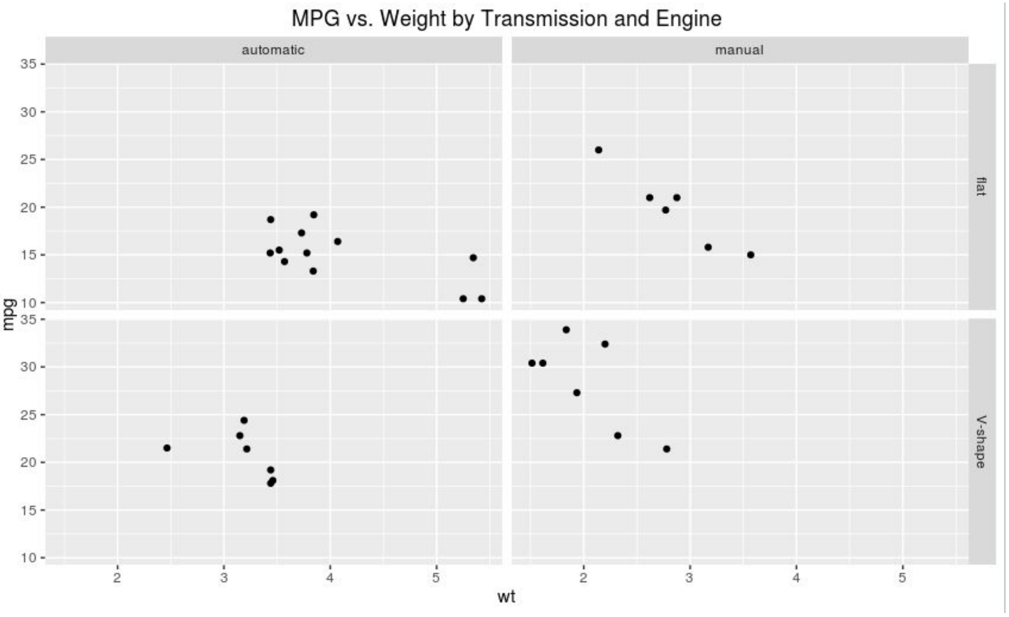
Code – One pair of stacked facets

```
qplot(x=wt, y=mpg,
    facets=vsf~.,#similar to above, but instead of the graphs side by side
    #they are stacked on each other
    data=mtcars, main="MPG vs. Weight by Engine")
```



Code – Two pairs of facets

```
qplot(x=wt, y=mpg,
    facets=vsf~amf,
    data=mtcars, main="MPG vs. Weight by Transmission and Engine")
```

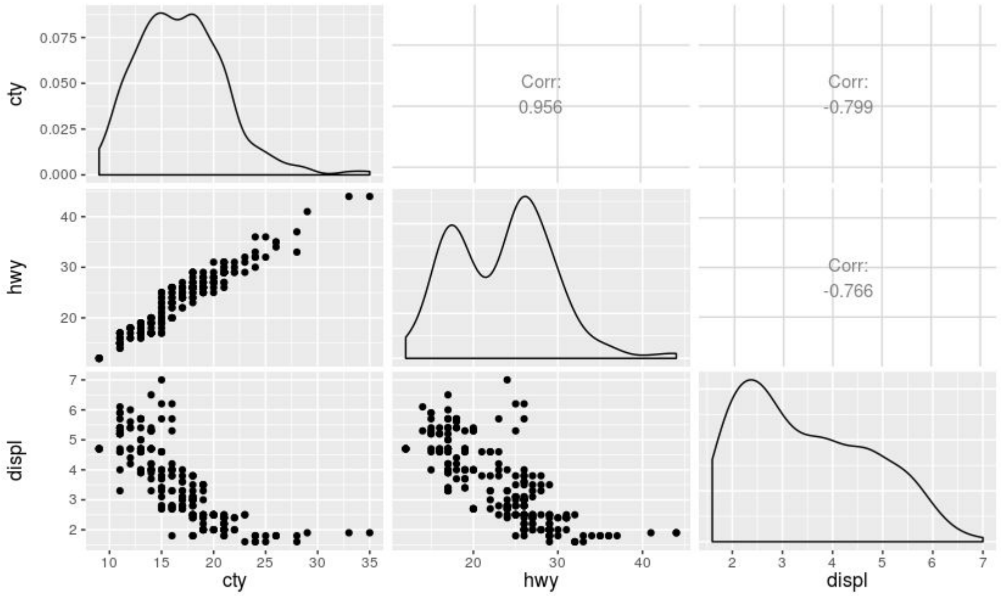


- Similar to above. This prints out 4 graphs / facets (rows correspond to shapes and columns correspond to transmission)
- This actually shows the relationship between 4 variables
- If we didnt use these panels, we would have to create multiple scatterplots, and it would be harder to interpret the relationship between these 4 variables
- Putting these panels side by side allows us to analyze 4 variables efficiently

Code – Illustrate facets among multiple variables

```
library(GGally)
DF <- mpg[,c('cty','hwy','displ')]
```

ggpairs(DF)



- Now use the function ggpairs() to illustrate relationships between multiple variables using an array of scatterplots, smooth histograms, and correlation values
- In the first line, create a dataframe that has 3 columns (cty, hwy, displ)
  - These are extracted from MPG
- Note this is not limited to 3 columns – you can have more
- ggpairs() appears to be very useful for getting a quick overview of the relationships between the features in the data

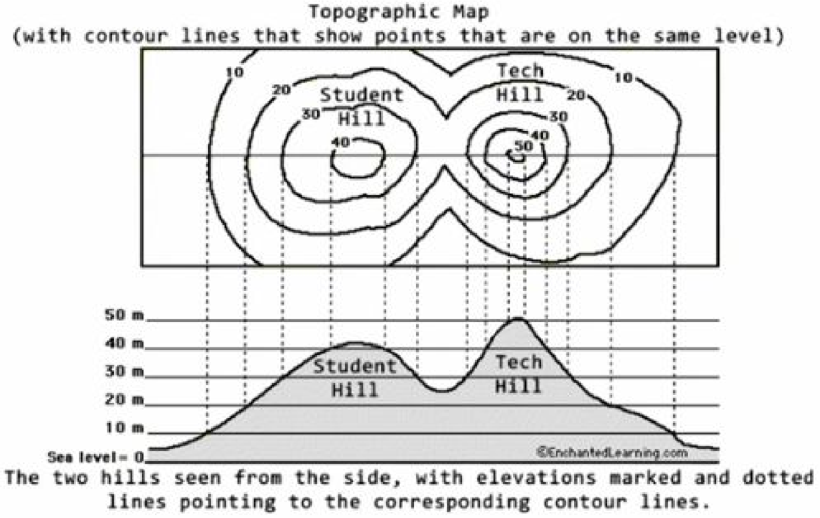
Contour Plots

Contour plots graph relationships between 3 numeric variables

- Typically, ‘z’ is a function of x and y

An example of a contour plot is a topographic map

Topographic Map
(with contour lines that show points that are on the same level)



The two hills seen from the side, with elevations marked and dotted lines pointing to the corresponding contour lines.

- For example, a hill’s location and height
- The different contours map out areas that have equal elevation

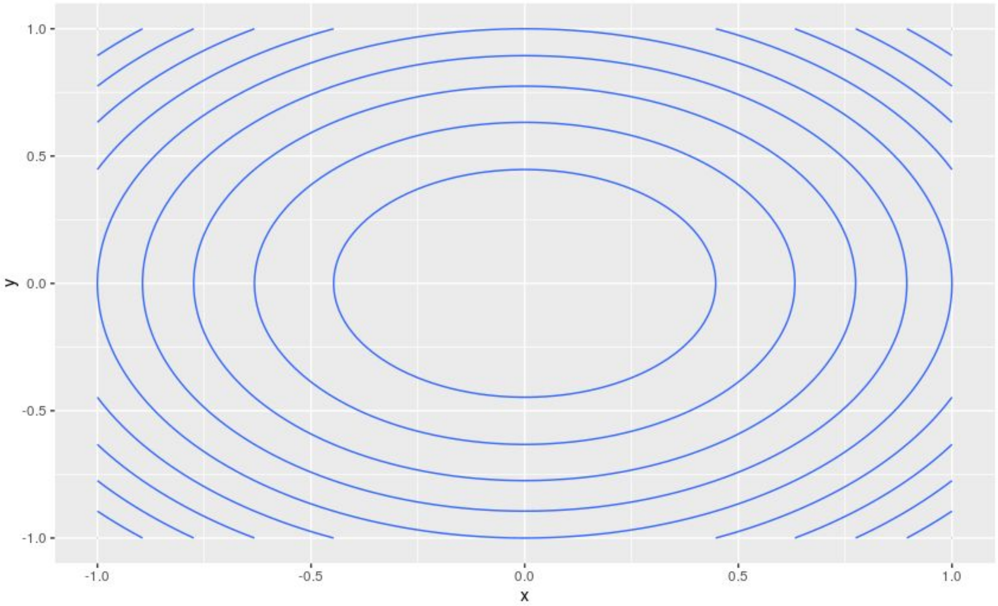
How to create a contour plot

- Create a grid for x values
- Create a grid of y values
- Create an expanded x by y grid

- Compute values of z on the expanded grid
- Graph the data

Code

```
#create a grid of x values
#create 100 points equally spaced between -1 and 1
xGrid <- seq(-1,1,length.out=100)
#create the y grid - since its identical to x, just copy x
yGrid <- xGrid
#create an expanded grid
R<-expand.grid(xGrid,yGrid)
#label the axis
names(R) = c('x','y')
#compute the z values; since they should be circular, the z function / elevation is
#the square of the x-axis plus the square of the y-axis
R$z <- R$x^2 + R$y^2
ggplot(R,aes(x=x,y=y,z=z)) + stat_contour()
```



Box Plots

Box plots are alternatives to histograms that are usually more lossy, in the sense that they lose more data

- As a plus, they emphasize quantiles and outliers in a way that a histogram cannot

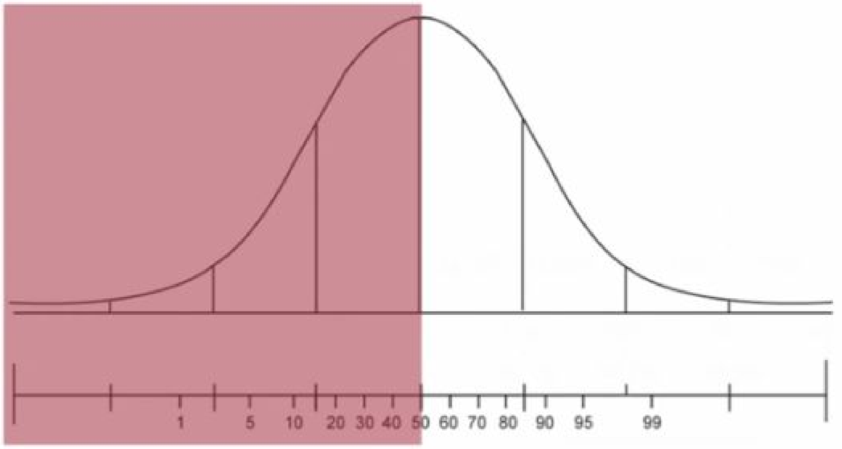
Histograms are good for showing the shape of the data (the modes etc), but a box plot is better for showing the median, IQR, whiskers, and outliers

- Box plots emphasize percentile information

The **R percentile** of a numeric dataset is the point at which approximately R percent of the data lies underneath and approximately 100-R percent if the data lie above

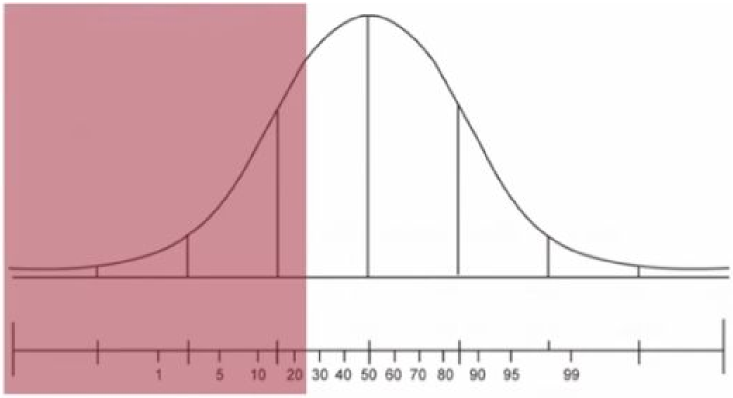
- Another name for the R percentile is the **zero point R quantile**

The **median** (also: **2nd Quartile**) is the point which 50% of the data lies under

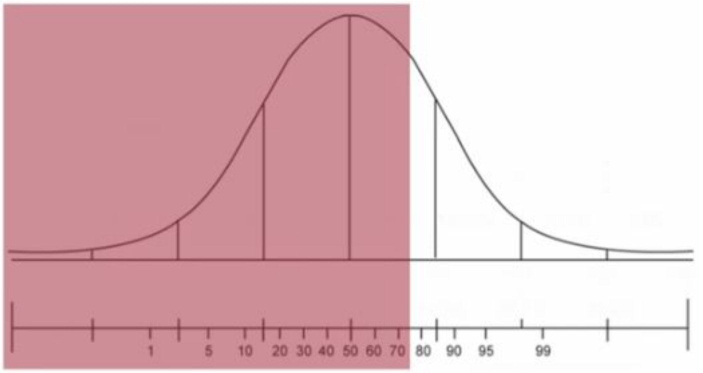


  - For Gaussians the median is in the middle of the box plot, but if one tail is heavier than the other the median will be off to the side (and thus non-Gaussian)

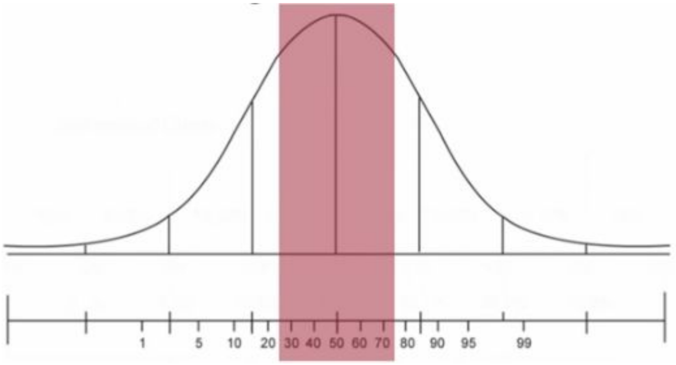
The **25th Percentile** (also: **1st Quartile**) is the point that 25% of the data lies to the left of



The **75th Percentile** (also: **3rd Quartile**) is the point that 75% of the data lies to the left of



The IQR (Inter-Quartile Range) is the range between the 25th percentile and the 75th percentile

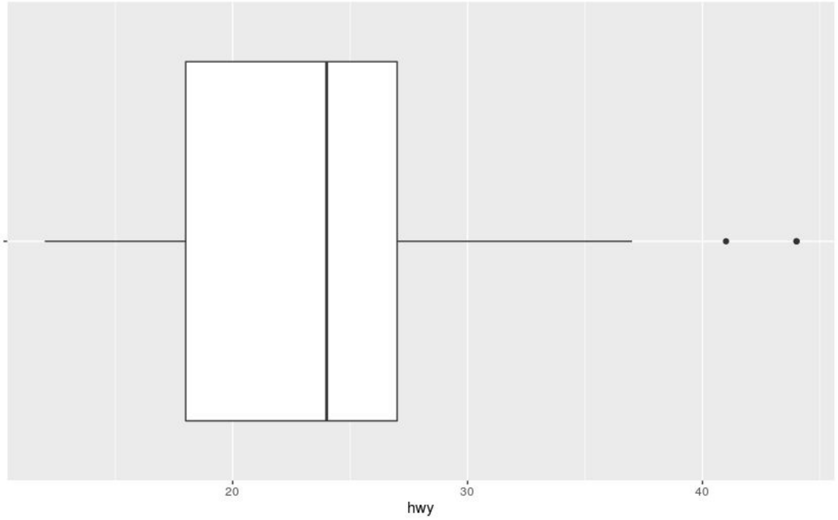


  - This is also 50% of the data (the central part of the data)

Code

```
ggplot(mpg,aes("",hwy)) +
  geom_boxplot() + #adds the boxplot geometry
  coord_flip() + #this makes the box plot lie on its side (as opposed to the default, which is
standing up)
  scale_x_discrete("") #this is required if we want to make the box lying down
```

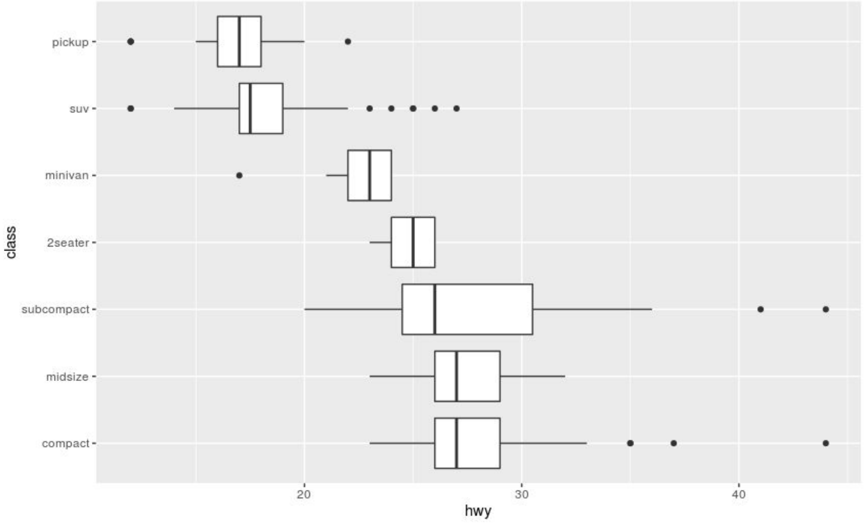
- this box plot creates a plot with the following properties:
    - a box denoting the IQR
    - An inner line bisecting the box denoting the median
    - Whiskers extending to the most extreme point (outliers) no further than 1.5 times IRQ length away from the edges of the box
    - Points outside the box and whiskers marked as outliers



Create multiple box plots - one for the highway mpg per each class of car in the mpg data

- having the data in different rows allows us to compare the box plots to each other
- Code

```
ggplot(mpg,aes(
  reorder(class,-hwy,median),hwy)) + #reorder function reorders the classes in
     #order of increasing highway mpg (using the median)
  geom_boxplot() + #adds the boxplot geometry
  coord_flip() + #this makes the box plot lie on its side
     #(as opposed to the default, which is standing up)
  scale_x_discrete("class") #this is required if we want to make the box lying down;
     #class describes the y axis properly
```



    - The classes are ordered by fuel efficiency (according to the median, which is represented by the line in the box)
    - Subcompacts have the highest spread of categories
    - Almost all 2-seaters have a higher highway mpg than SUVs
    - SUVs and 2-seaters have almost disjoint values (meaning: no overlap)

Comparing box plots to smoothed histograms

- The box plot does not convey the multimodal nature of the distribution that the histogram shows.
- It is easier to read the median and the IQR in the box plot, which show the center and central 50% range from the box plot.

QQPlots

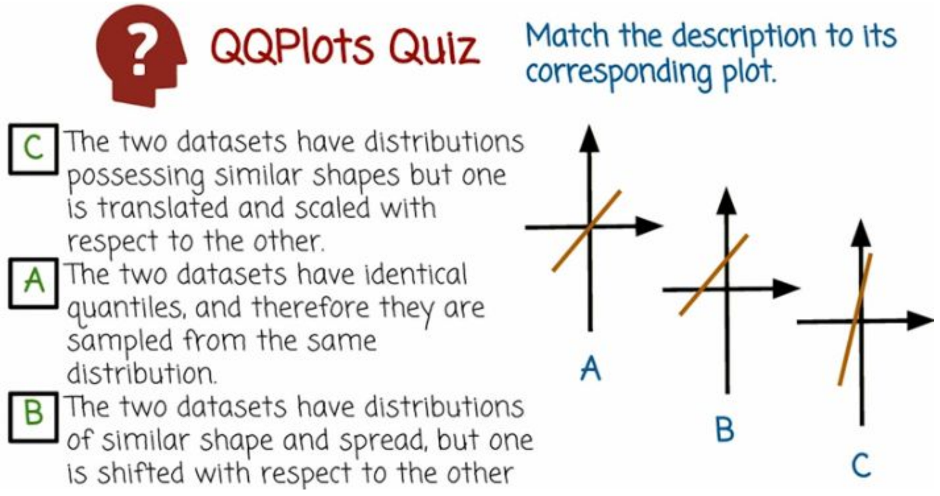
Quantile-quantile plots, or **QQPlots**, are scatter plots of quantiles of one dataset on the x-axis vs the quantiles of another dataset on the y-axis

- In some cases, either the x or y describes quantiles coming from a theoretical distribution rather than a specific dataset

Since this is usually the quantiles of one dataset vs the quantiles of another, the shape of the scatter plot implies the following conclusions

- A straight line with slope 1 that passes through the origin implies that the two datasets have identical quantiles, and therefore that they are sampled from the same distribution
- A straight line with slope 1 that does not pass through the origin implies that the two datasets have distributions of similar shape and spread, but that the one is shifted with respect to the other
- A straight line with slope different from 1 that does not pass through the origin implies that the two datasets have distributions possessing similar shapes but that one is translated and scaled with respect to the other.
- A non-linear S shape implies that the dataset corresponding to the x-axis is sampled from a distribution with heavier tails than the other dataset.
- A non-linear reflected S shape implies that the dataset whose quantiles correspond to the y-axis is drawn from a distribution having heavier tails than the other dataset.

QQPlot Quiz 1



QQPlots Quiz

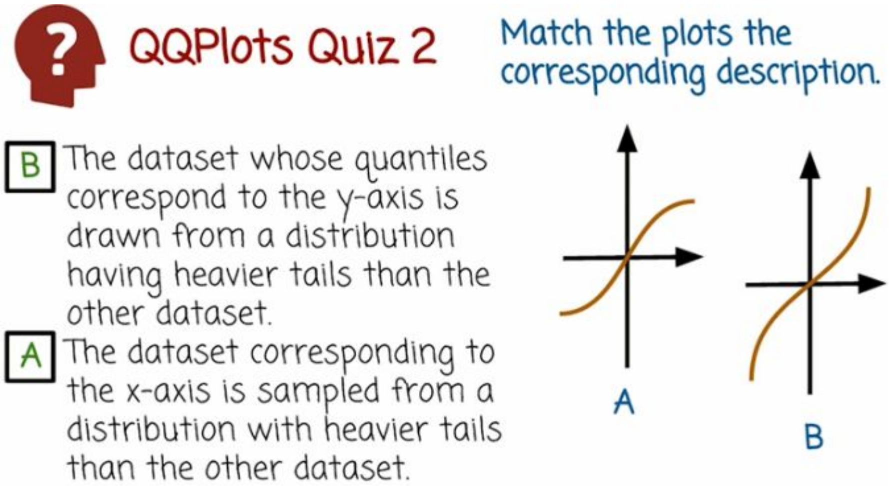
Match the description to its corresponding plot.

C The two datasets have distributions possessing similar shapes but one is translated and scaled with respect to the other.

A The two datasets have identical quantiles, and therefore they are sampled from the same distribution.

B The two datasets have distributions of similar shape and spread, but one is shifted with respect to the other

QQPlot Quiz 2



QQPlots Quiz 2

Match the plots the corresponding description.

B The dataset whose quantiles correspond to the y-axis is drawn from a distribution having heavier tails than the other dataset.

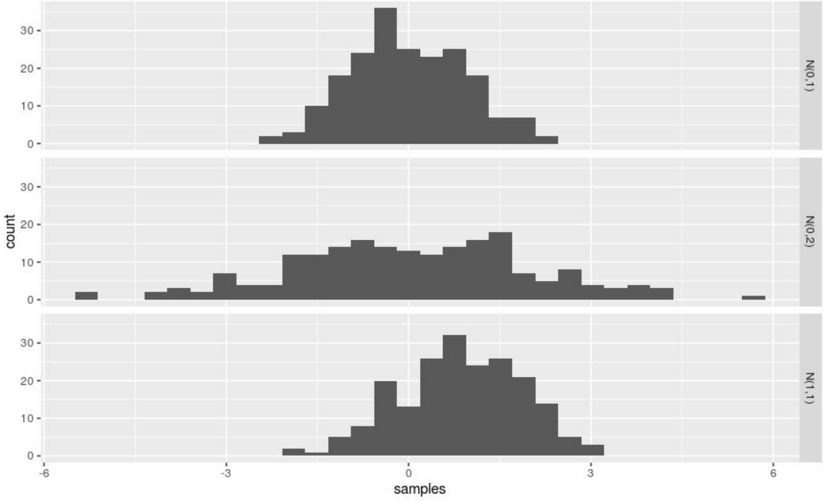
A The dataset corresponding to the x-axis is sampled from a distribution with heavier tails than the other dataset.

- The first one is B because a specific value of the x coordinate is being mapped to a higher value on the y coordinate because of the shape of the graph
- For the second, the specific value on the x axis will be mapped to a lower value on the y axis
  - The quantiles on the y axis are shrunk corresponding to the quantiles on the x axis

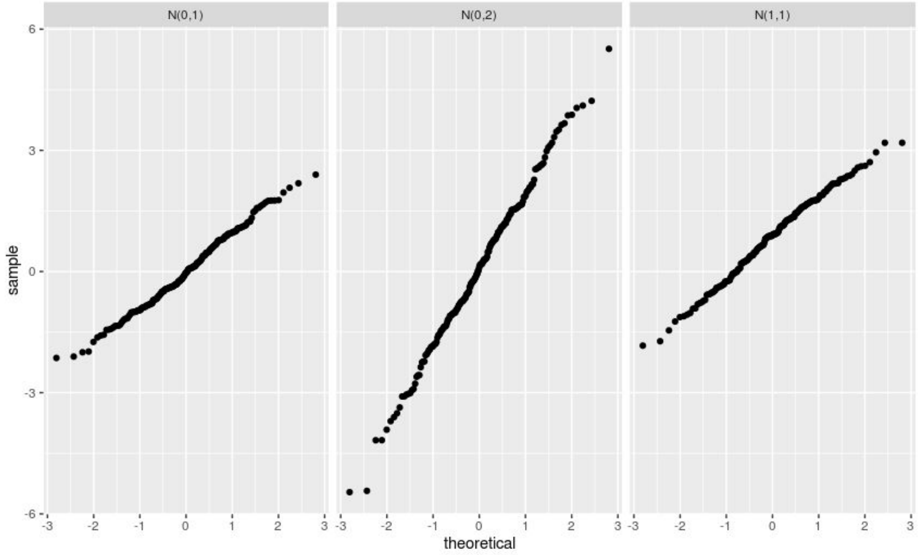
QQPlot Example Code 1

- Code

```
#create a data frame with samples from 3 different normal distributions
#*each sample has 200 points with 3 different sets of parameters
#**one has a mean 1 and STD 1
#**one has a mean 0 and STD 1
#**one has a mean 0 and STD 2
D <- data.frame(samples = c(rnorm(200,1,1), rnorm(200,0,1),rnorm(200,0,2)))
#name the samples for the graph
D$parameter[1:200] <- 'N(1,1)';
D$parameter[201:400] <- 'N(0,1)';
D$parameter[401:600] <- 'N(0,2)';
#show the graphs as a histogram.  This isnt the QQPlot but is meant to show the
#   distributions visually
qplot(samples,facets=parameter~.,
    geom='histogram',
    data=D)
```



```
#now for the QQPlot
ggplot(D, aes(sample = samples)) +
  stat_qq() +
  facet_grid(.~parameter)
```

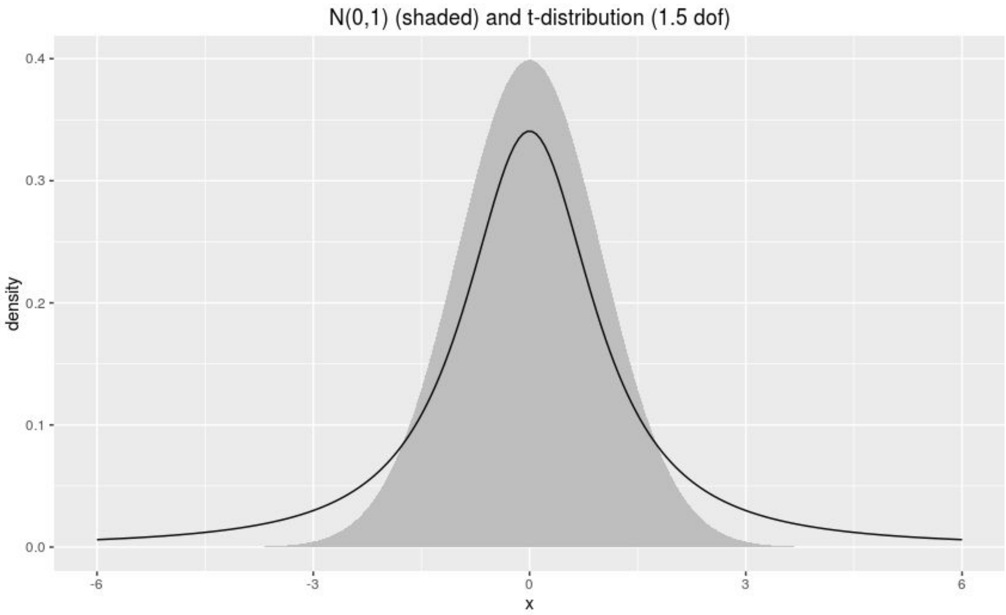


- Looking at these graphs you can tell that the top histogram has a center of 0 and a STD of 1, the middle has a center of 0 and a STD of 2, and the bottom has a center at 1 and a STD of 1
- This creates QQPlots of the given samples against the theoretical Gaussian distribution with the parameters 0 and 1
- Put the results in a facet graph, where the value of the parameter corresponding to thedifferent dataset
- This will give us a facet graph with three panels arranged along columns
- In each panel we will have a QQPlot of each of the datasets against the theoretical normal distribution with parameters 0 and 1
- All three graphs show a linear shape (and are explained in Quiz 1 above)

QQPlot Example 2

- Code

```
xGrid <- seq(-6,6, length.out = 200)#create grid between -6 and 6 having 200 equally
spaced points
R <- data.frame(density = dnorm(xGrid, 0, 1))#compute the gaussian density / density
function of the normal distribusions
  #with parameters 0 and 1 at these grid points
R$tdensity <- dt(xGrid, 1.5)
R$x <- xGrid
ggplot(R, aes(x=x, y=density)) +
  geom_area(fill = I('grey')) + #for the Gaussian distribution, we want it to be displayed using
    #filled area - so call the fill gemotery
  geom_line(aes(x=x, y=tdensity)) + #for the t-density use a line graph - so use the line
geometry
  labs(title = "N(0,1) (shaded) and t-distribution (1.5 dof)")
```

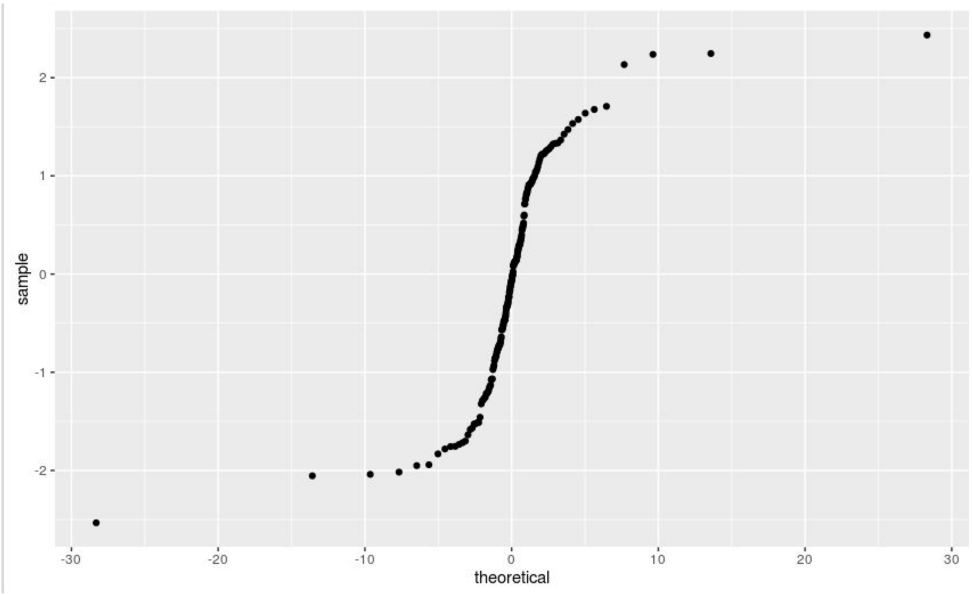

N(0,1) (shaded) and t-distribution (1.5 dof)

- Another example of the QQPlot where we will show the quantiles of the Gaussian distribution on one axis and the t-distribution (theoretical distribution?) on another axis
    - The t-distribution has much heavier tails than the normal distribution and it has a fundamentally different shape in that the tails decay polynomially rather than exponentially; this will give us a non-linear QQPlot shape
- Looking at the graph, we get two distributions, the t-distribution using the solid line and the Gaussian/normal distribution using the shaded area
- Both seem similar at first, having a bell shaped curve.
    - However, the tails of the t-distribution decay much slower than the tails of the Gaussian distribution no matter the variance of the Gaussian

QQPlot Example 3

- Code

```
xGrid <- seq(-6,6, length.out = 200)#create grid between -6 and 6 having 200 equally
spaced points
R <- data.frame(density = dnorm(xGrid, 0, 1))#compute the gaussian density / density
function of
#   the normal distribusions
R$samples <- rnorm(200,0,1)
pm <- list(df = 1.5)
ggplot(R, aes(sample=samples)) +
  stat_qq(distribution = qt, dparams=pm) #draw the quantiles of the dataset against the
quantiles
#   from the distribution, denoted
  #here by 'qt' which is the t-distribution with the specific parameter (df corresponds to the
degrees of
#   freedom 1.5)
```



- This displays the QQPlot of samples from the normal/Gaussian distribution verses the theoretical quantiles of the t-distribution
- As for the plot this generates, the x-axis describes the quantiles of the theoretical t-distribution
    - The y-axis corresponds to the quantiles of the dataset drawn from a normal distribution
    - Refer to the earlier quiz - the presence of the 'S' shape is consistent with the case of the t-distribution and the Gaussian distribution having fundamentally different shape
    - with the t-distribution having much heavier tails.

Devices

By default R overwrites plot areas

- To open a new plot area, the R code is

```
dev.new()
```

- To save a plot:

```
ggsave(file=”myPlot.pdf”)
```

    - R picks up on the type and saves it according to that type(pdf, postscript, jpeg, etc)
- To send all future graphs to a file, use one of these commands: postscript, pdf, xfig, bmp, png, jpeg, tiff
    - out of these, pdf, postscript, and xfig allows the user to zoom in after the fact
        PDF is most popular / preferred
    - Do NOT forget to close the stream with: dev.off()
    - Example code

```
pdf(file="MyFile.pdf", height = 5, width = 5, pointsize = 11)
ggplot(D, aes(sample = samples)) +  stat_qq() + facet_grid(.~parameter)
qplot(samples, facets = parameter~., geom = 'histogram', data = D)
dev.off()
```

Data Preparation

Lesson Summary

We should now be comfortable with

- Graphs and plots
- Some of the native datasets in R

We learned a variety of ways to visualize data