Data Exploration and Visualization Booklet

Data Science Dojo

Data Exploration Using R

Titanic tragedy data

Reading RAW training data

- Download the data set "Titanic_train.csv" at https://github.com/datasciencedojo/bootcamp/tree/master/Datasets/Titanic_train.csv
- Set working directory of R to the directory of the file using setwd()

```
titanic = read.csv("Titanic_train.csv"); colnames(titanic)
```

```
##  [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"
##  [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
## [11] "Cabin"       "Embarked"
```

```
colnames(titanic)[5] <- "Gender" ## revise the name of a column
```

Look at the first few rows

```
head(titanic[,1:5], 4)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
##                                                 Name Gender
## 1                            Braund, Mr. Owen Harris   male
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female
## 3                             Heikkinen, Miss. Laina female
## 4       Futrelle, Mrs. Jacques Heath (Lily May Peel) female
```

Look at the first few rows

What would be some good features to be considered?

```
head(titanic[,6:ncol(titanic)], 4)
```

```
##   Age SibSp Parch           Ticket    Fare Cabin Embarked
## 1  22     1     0        A/5 21171  7.2500              S
## 2  38     1     0         PC 17599 71.2833   C85        C
## 3  26     0     0 STON/O2. 3101282  7.9250              S
## 4  35     1     0           113803 53.1000  C123        S
```

Summary of the data frame - str(titanic)

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",..: 109 191 35..
##  $ Gender     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : Factor w/ 681 levels "110152","110413",..: 525 596 662 50..
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 1 57 1 1 131..
##  $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

Converting class label to a factor

```
titanic$Survived = factor(titanic$Survived)
titanic$Embarked = factor(titanic$Embarked)
str(titanic$Survived)
```

```
##  Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
```

```
str(titanic$Embarked)
```

```
##  Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 3 4 4 4 2 ...
```

Rename the levels of factors


```
levels(titanic$Survived) <- c("died", "survived")
levels(titanic$Embarked) <- c("Unknown", "Cherbourg", "Queenstown", "Southampton")
str(titanic$Survived)
```

```
##  Factor w/ 2 levels "died","survived": 1 2 2 2 1 1 1 1 2 2 ...
```

```
str(titanic$Embarked)
```

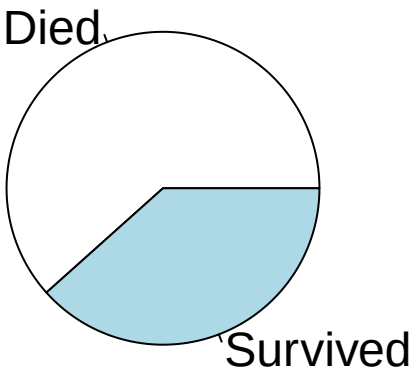
```
##  Factor w/ 4 levels "Unknown","Cherbourg",..: 4 2 4 4 4 3 4 4 4 2 ...
```

Class (survived or died) distribution - PIE Charts

```
survivedTable = table(titanic$Survived); survivedTable
```

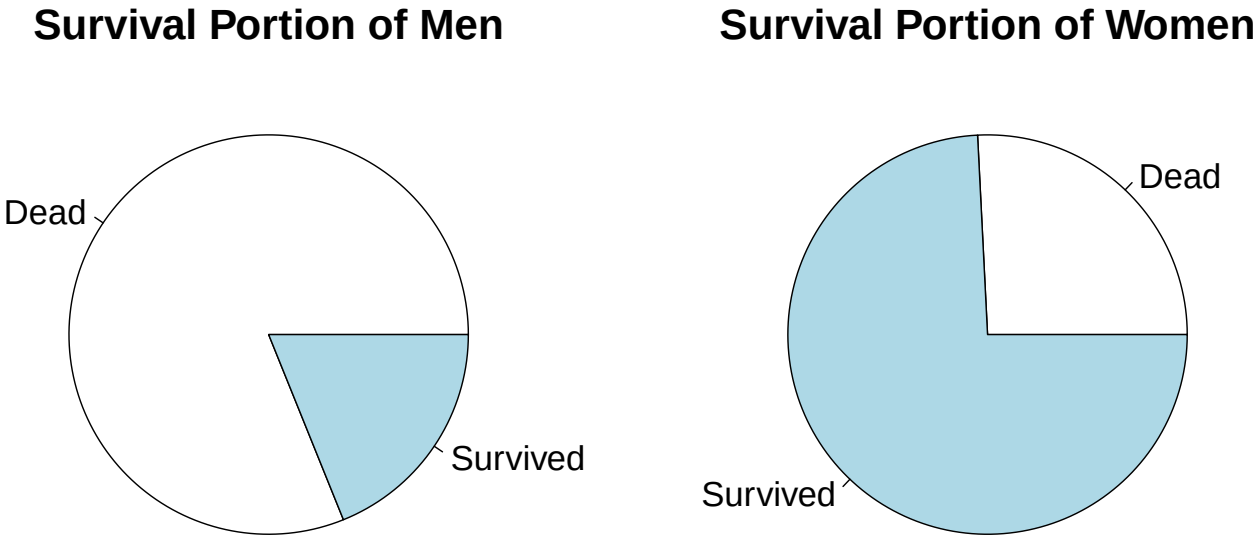
```
##
##     died survived
##      549      342
```

```
# mar, and oma mean margin and outer margin of the plot,
# cex gives the size of texts
par(mar = c(0, 0, 0, 0), oma = c(0, 0, 0, 0), cex=1.5)
pie(survivedTable,labels=c("Died","Survived"))
```



Is Gender a good predictor?

```
par(mfrow = c(1, 2), mar = c(0, 0, 2, 0), oma = c(0, 1, 0, 1), cex=1.5)
pie(table(titanic[titanic$Gender=="male", "Survived"]),
    labels=c("Dead","Survived"),  main="Survival Portion of Men")
pie(table(titanic[titanic$Gender=="female", "Survived"]),
    labels=c("Dead","Survived"), main="Survival Portion of Women")
```

Is Age a good predictor?

```
Age <- titanic$Age; summary(Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.42   20.12   28.00   29.70   38.00   80.00     177
```

How about summary segmented by survival?

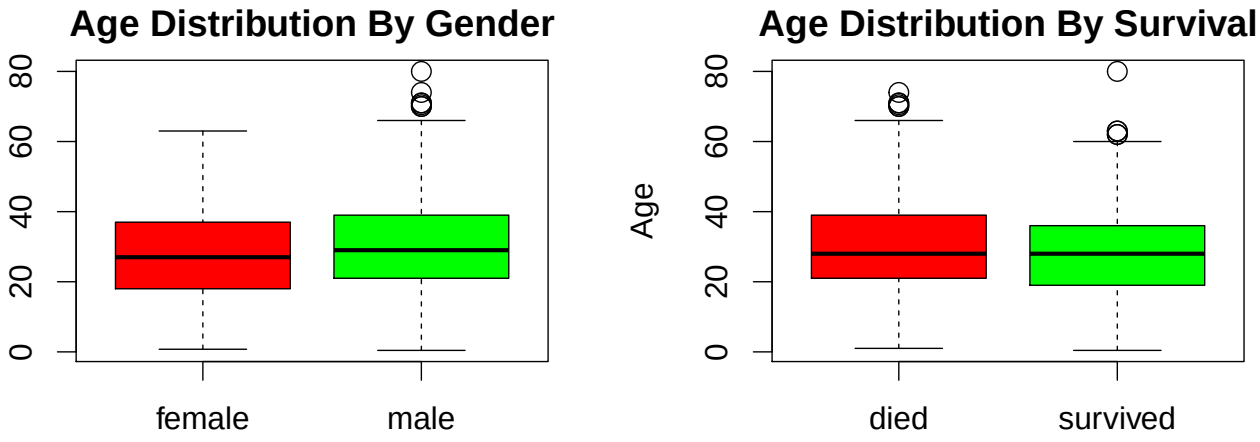
```
summary(titanic[titanic$Survived=="died", "Age"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    1.00   21.00   28.00   30.63   39.00   74.00     125
```

```
summary(titanic[titanic$Survived=="survived", "Age"])
```

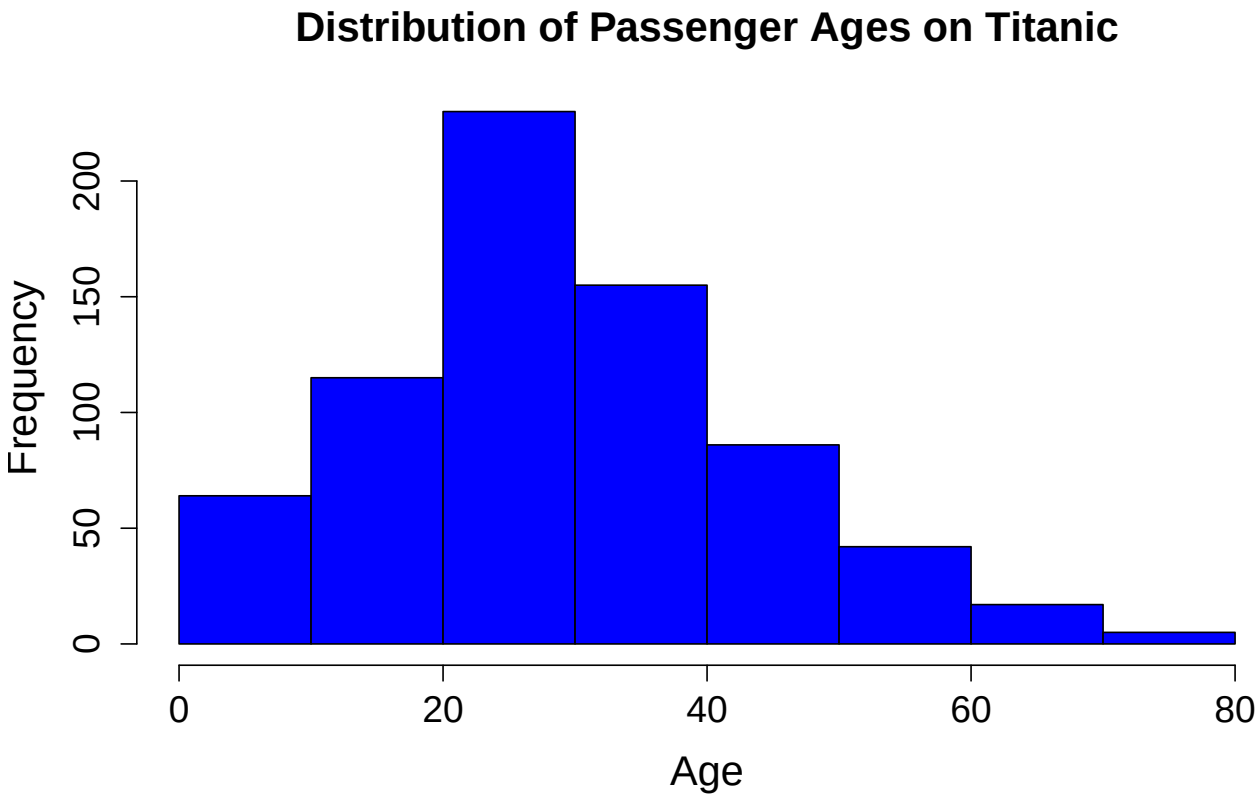
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.42   19.00   28.00   28.34   36.00   80.00      52
```

Age distribution by Survival and Gender



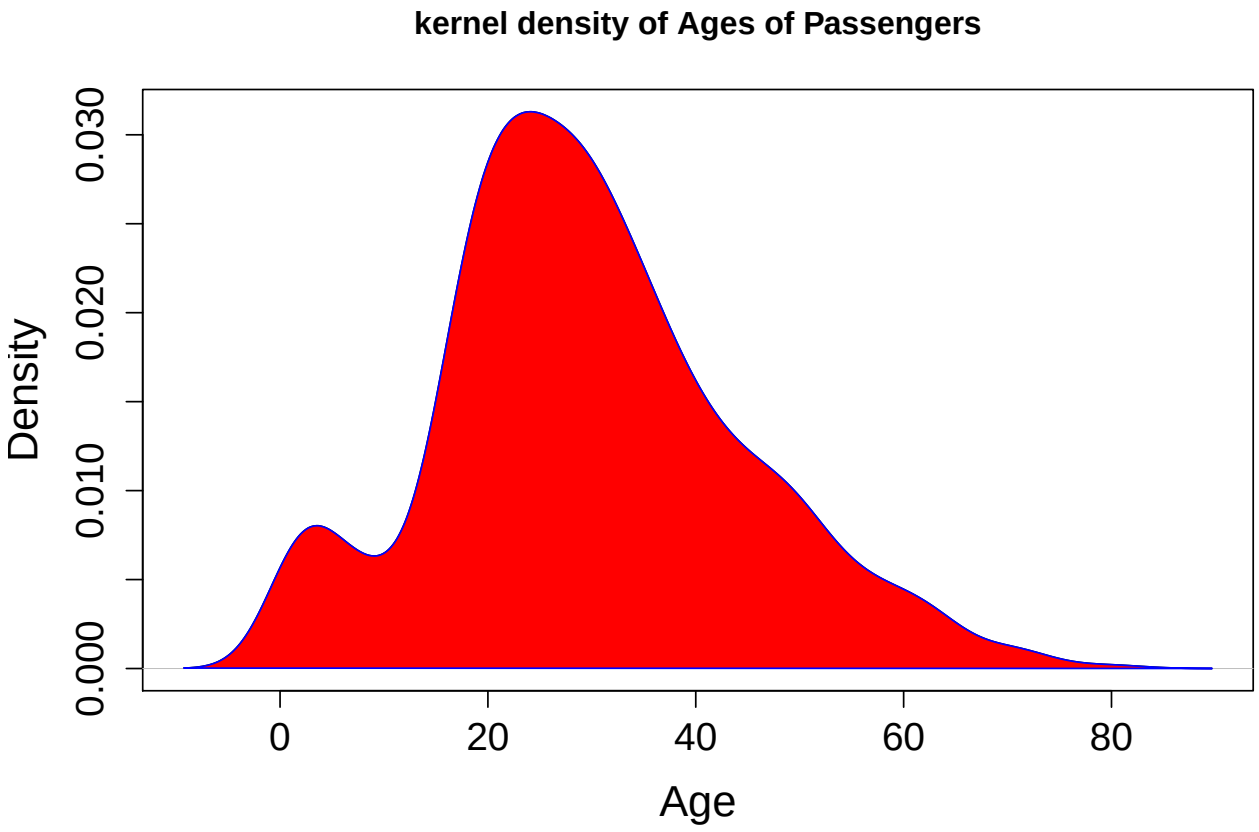
Histogram of Age

```
hist(Age, col="blue", xlab="Age", ylab="Frequency",
     main = "Distribution of Passenger Ages on Titanic",
     cex.lab=1.6, cex.axis=1.4, cex.main=1.6)
```

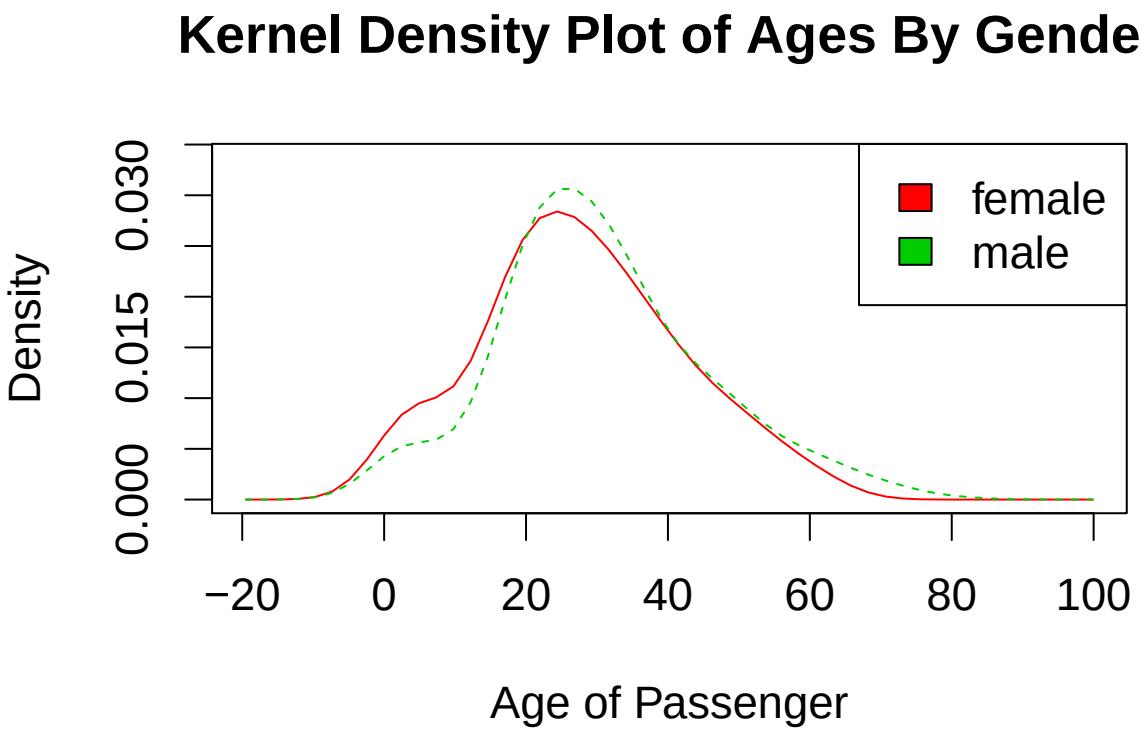


Kernel density plot of Age

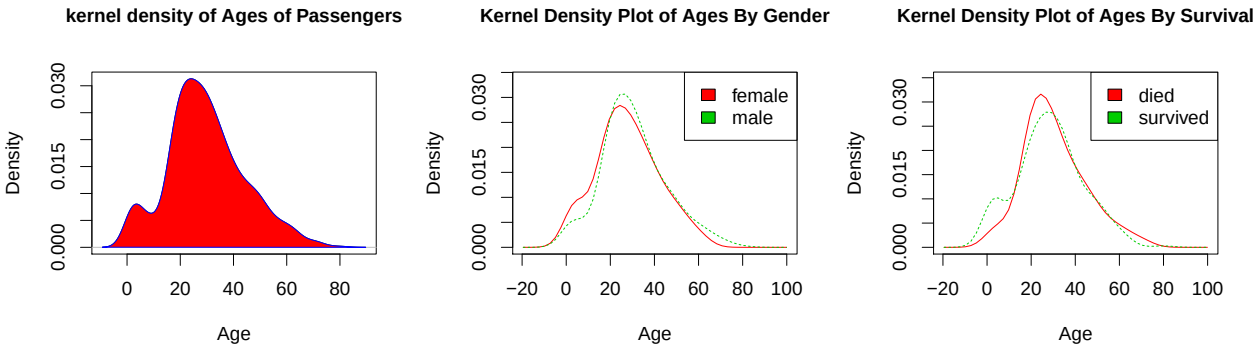
```
d = density(na.omit(Age)) # density() requires all NAs to be removed
plot(d, main = "kernel density of Ages of Passengers",
     xlab="Age", cex.lab=1.6, cex.axis=1.4)
polygon(d, col="red", border="blue")
```



Comparison of density plots of Age with different Gender



Did Age have an impact on survival?



Create categorical groups: Adult vs Child

An example of **feature engineering**!

```
## Multi dimensional comparison
Child <- titanic$Age # Isolating age.
## Now we need to create categories: NA = Unknown, 1 = Child, 2 = Adult
## Every age below 13 (exclusive) is classified into age group 1
```



```
Child[Child<13] <- 1
## Every child 13 or above is classified into age group 2
Child[Child>=13] <- 2
## Use labels instead of 0's and 1's
Child[Child==1] <- "Child"
Child[Child==2] <- "Adult"
```

Create categorical groups: Adult vs Child

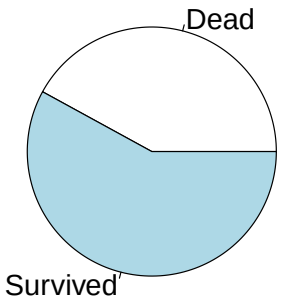
```
# Appends the new column to the titanic dataset
titanic_with_child_column <- cbind(titanic, Child)
# Removes rows where age is NA
titanic_with_child_column <- titanic_with_child_column[!is.na(titanic_with_child_column$Child),]
## Show part of the data frame with new feature: Child
head(titanic_with_child_column[,c(1:3, 13)], 4)
```

```
##   PassengerId Survived Pclass Child
## 1           1     died      3 Adult
## 2           2 survived      1 Adult
## 3           3 survived      3 Adult
## 4           4 survived      1 Adult
```

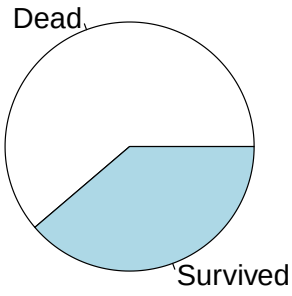
Create categorical groups: Adult vs Child

```
par(mfrow = c(1, 2), mar = c(0, 1, 2, 1), oma = c(0, 1, 0, 1), cex=1.5)
pie(table(titanic_with_child_column[titanic_with_child_column$Child=="Child", "Survived"]),
    labels=c("Dead","Survived"),  main="Survival Portion of Child")
pie(table(titanic_with_child_column[titanic_with_child_column$Child=="Adult", "Survived"]),
    labels=c("Dead","Survived"),  main="Survival Portion of Adult")
```

Survival Portion of Child



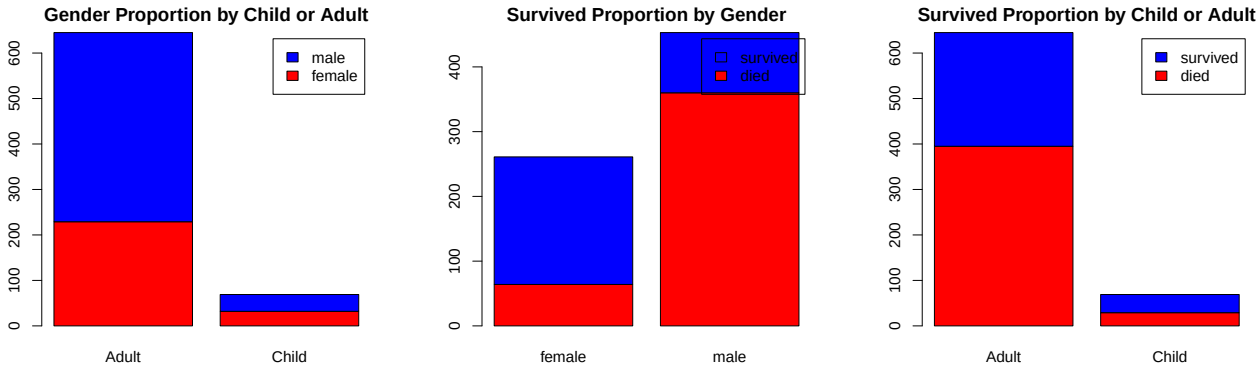
Survival Portion of Adult



Child vs Gender

```
child.gender <- table(titanic_with_child_column$Gender, titanic_with_child_column$Child)
gender.survived <- table(titanic_with_child_column$Survived, titanic_with_child_column$Gender)
child.survived <- table(titanic_with_child_column$Survived, titanic_with_child_column$Child)
barplot(child.gender, main="Gender Proportion by Child or Adult",
        col=c("red","blue"), legend = rownames(child.gender))
barplot(gender.survived, main="Survived Proportion by Gender",
        col=c("red","blue"), legend = rownames(gender.survived))
barplot(child.survived, main="Survived Proportion by Child or Adult",
        col=c("red","blue"), legend = rownames(child.survived))
```

Child vs Gender



Data visualization using R

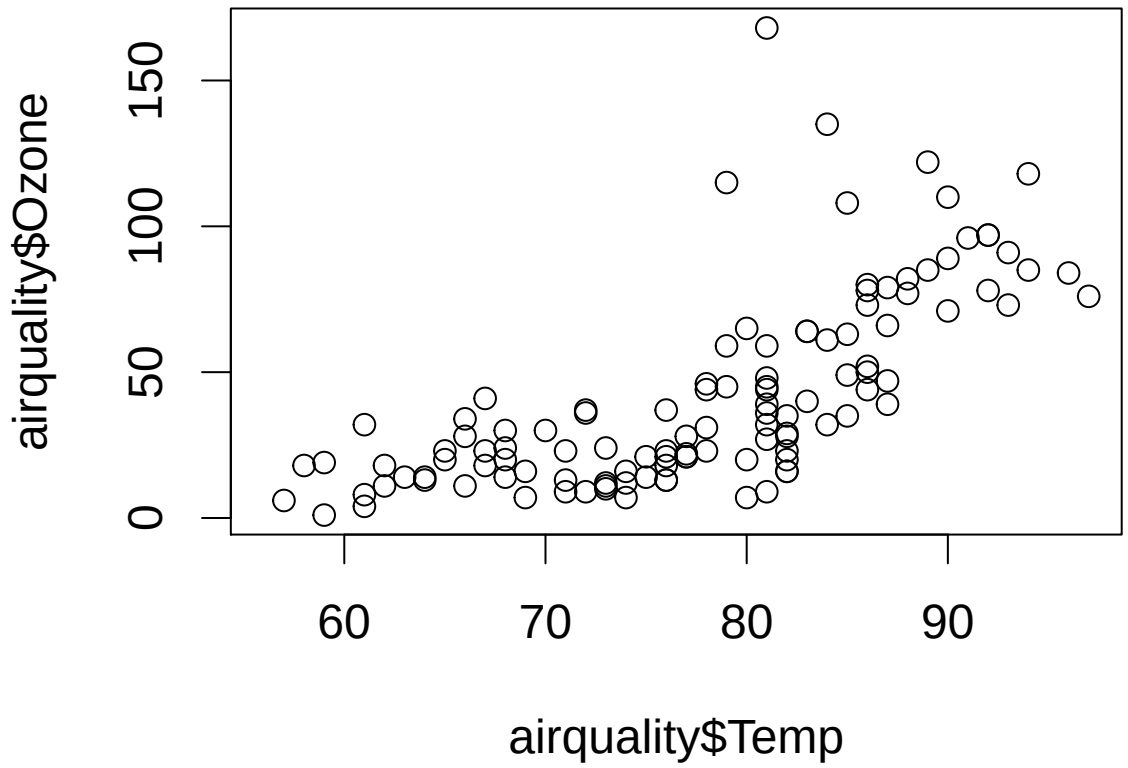
1. **Base graphics**: constructed piecemeal. Conceptually simpler and allows plotting to mirror the thought process.
2. **Lattice graphics**: entire plots created in a simple function call.
3. **ggplot2 graphics**: an implementation of the Grammar of Graphics by Leland Wikinson. Combines concepts from both base and lattice graphics. (Need to install ggplot2 library)
4. Fancier and more telling ones.

A list of interactive visualization in R can be found at:
http://ouzor.github.io/blog/2014/11/21/interactive-visualizations.html

---

Base plotting system

```
par(cex=1.5) ## increase the size of texts
## scatter plot
plot(x = airquality$Temp, y = airquality$Ozone)
```

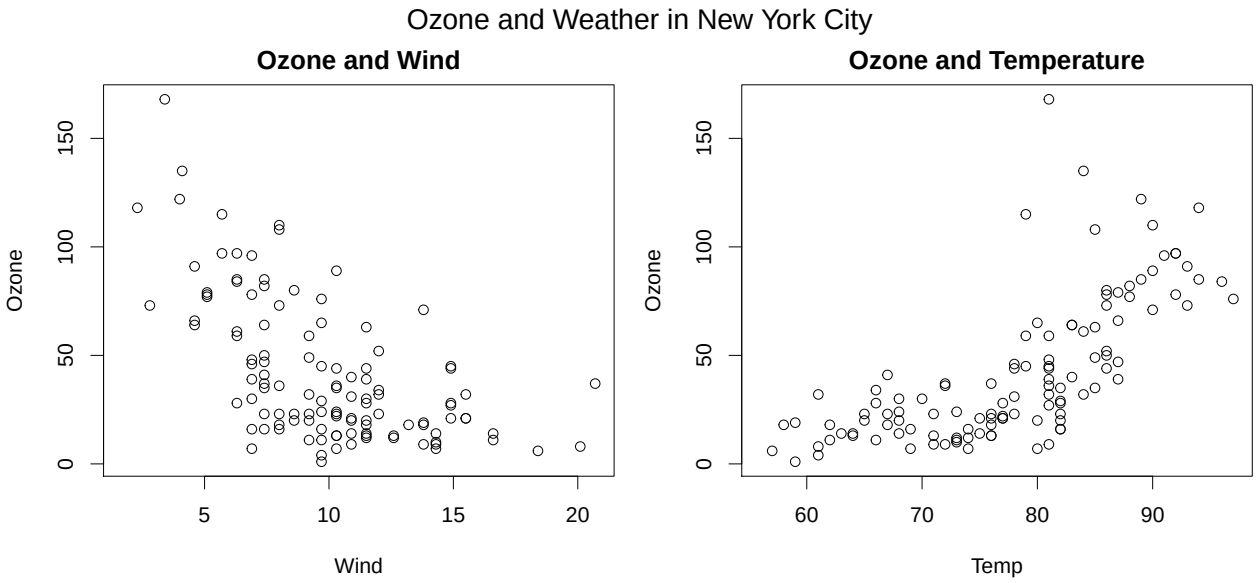



Base plotting system

```
## par() function is used to specify global graphics parameters
## that affect all plots in an R session. Type ?par to see all parameters
par(mfrow=c(1, 2), mar=c(4, 4, 2, 1), oma=c(0, 0, 2, 0), cex=1.5)
## mfrow=c(1, 2): the figures will be drawn in an 1x2 array by row.
## mar/oma: A numerical vector of the form 'c(bottom, left, top, right)',
## which gives the number of lines of margin to be specified on the four sides
## of the plot for mar, and size of the outer margins in lines of text for oma.
with(airquality, {
    plot(Wind, Ozone, main="Ozone and Wind")
    plot(Temp, Ozone, main="Ozone and Temperature")
    mtext("Ozone and Weather in New York City", outer=TRUE)})
```

Difference and relation of mar and oma, check out
http://research.stowers-institute.org/mcm/efg/R/Graphics/Basics/mar-oma/index.htm.

Base plotting system



Plotting functions (high level)

PHASE ONE: Mount a canvas panel on the easel, and draw the draft. (Initialize a plot.)

- **boxplot()**: a boxplot show the distribution of a vector. It is very useful to example the distribution of different variables.
- **plot()**: one of the most frequently used plotting functions in R.
- **barplot()**: create a bar plot with vertical or horizontal bars.
- **hist()**: compute a histogram of the given data values.
- **pie()**: draw a pie chart.

Remember to use **?plot** or **str(plot)**, etc. to check the arguments when you want to make more personalized plots.
A **tutorial** of base plotting system with more details:
http://bcb.dfci.harvard.edu/~aedin/courses/BiocDec2011/2.Plotting.pdf

---

Plotting functions (low level)

PHASE TWO: Add more details on your canvas, and make an artwork. (Add more on an existing plot.)

- **lines**: adds liens to a plot, given a vector of x values and corresponding vector of y values
- **points**: adds a point to the plot
- **text**: add text labels to a plot using specified x,y coordinates
- **title**: add annotations to x,y axis labels, title, subtitles, outer margin

- **mtext**: add arbitrary text to margins (inner or outer) of plot
- **axis**: specify axis ticks

Save your artwork

R can generate graphics (of varying levels of quality) on almost any type of display or printing device. Like:

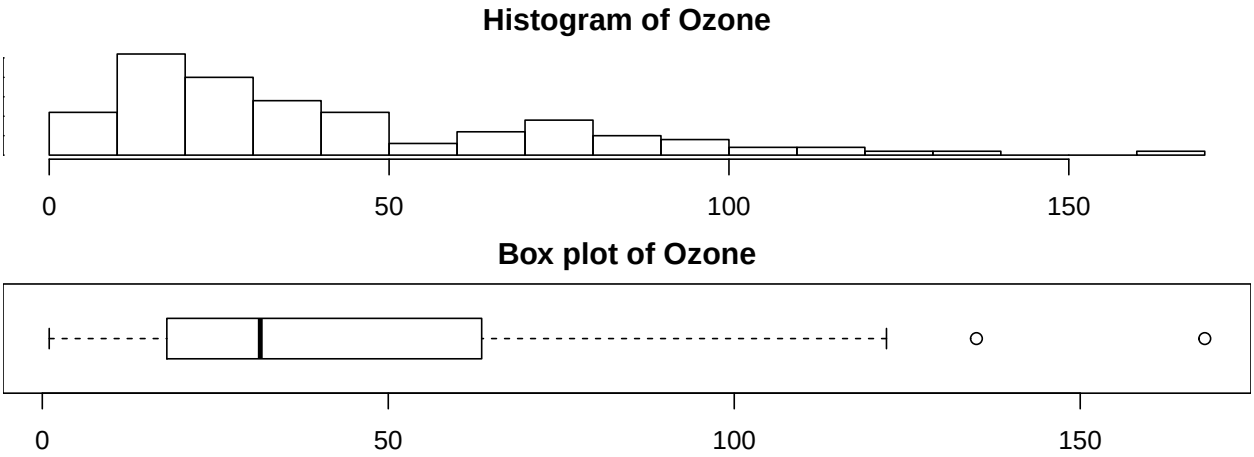
- **postscript()**: for printing on PostScript printers, or creating PostScript graphics files.
- **pdf()**: produces a PDF file, which can also be included into PDF files.
- **jpeg()**: produces a bitmap JPEG file, best used for image plots.

help(Devices) for a list of them all. Simple example:

```
## png(filename = 'plot1.png', width = 480, height = 480, units = 'px')
## plot(x, y)
## dev.off()
```

Example: boxplot and histogram

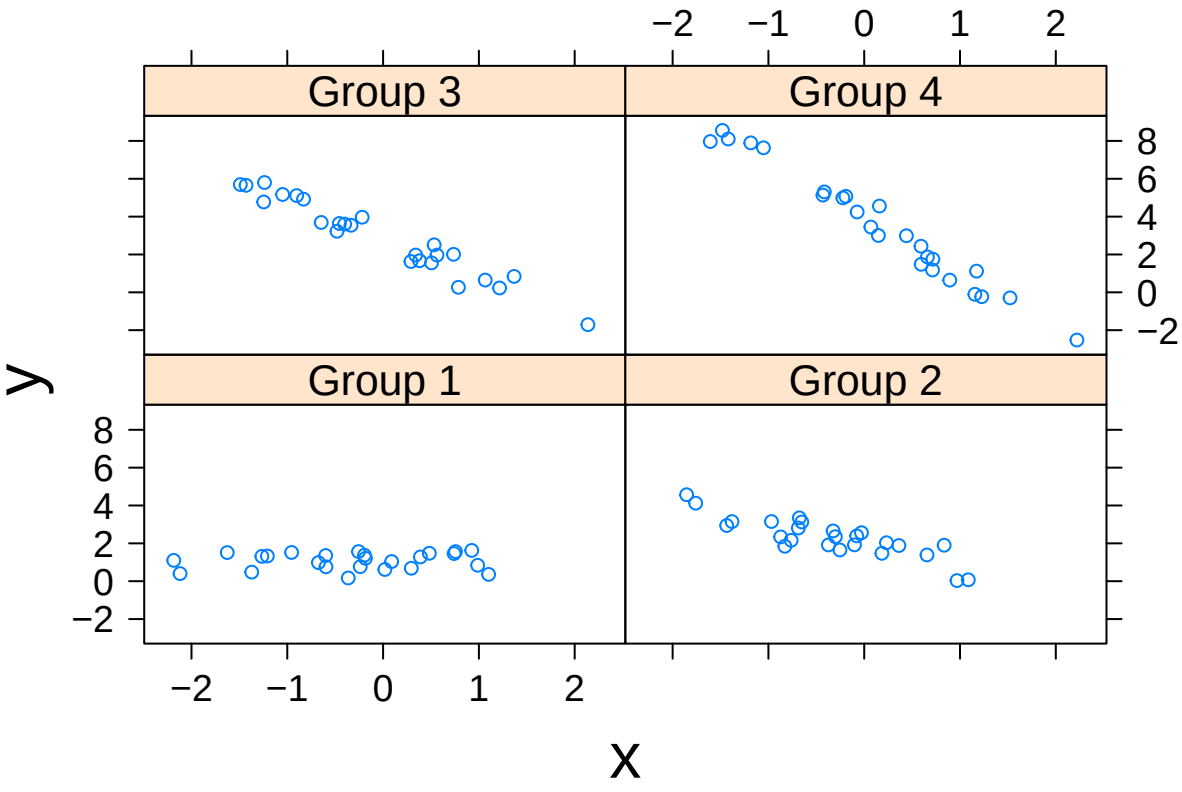
```
## the layout
par(mfrow = c(2, 1), mar = c(2, 0, 2, 0), oma = c(0, 0, 0, 0))
## histogram at the top
hist(airquality$Ozone, breaks=12, main = "Histogram of Ozone")
## box plot below for comparison
boxplot(airquality$Ozone, horizontal=TRUE, main = "Box plot of Ozone")
```



Lattice plotting system


```
## install.packages("lattice") # install the lattice library if you haven't
library(lattice) # load the library
## set the seed so our plots are the same
set.seed(10)
## use rnorm(): generate 100 random numbers
x <- rnorm(100)
## use rep() function to generate a vector, with first 25 elements are 1,
## second 25 elements are 2, ...
f <- rep(1:4, each = 25)
y <- x + f - f * x + rnorm(100, sd = 0.5)
# name the levels of the factor: first 25 elements are in Group 1,
## second 25 elements are in Group 2, ...
f <- factor(f, labels = c("Group 1", "Group 2", "Group 3", "Group 4"))
xyplot(y ~ x | f)
```

Lattice plotting system

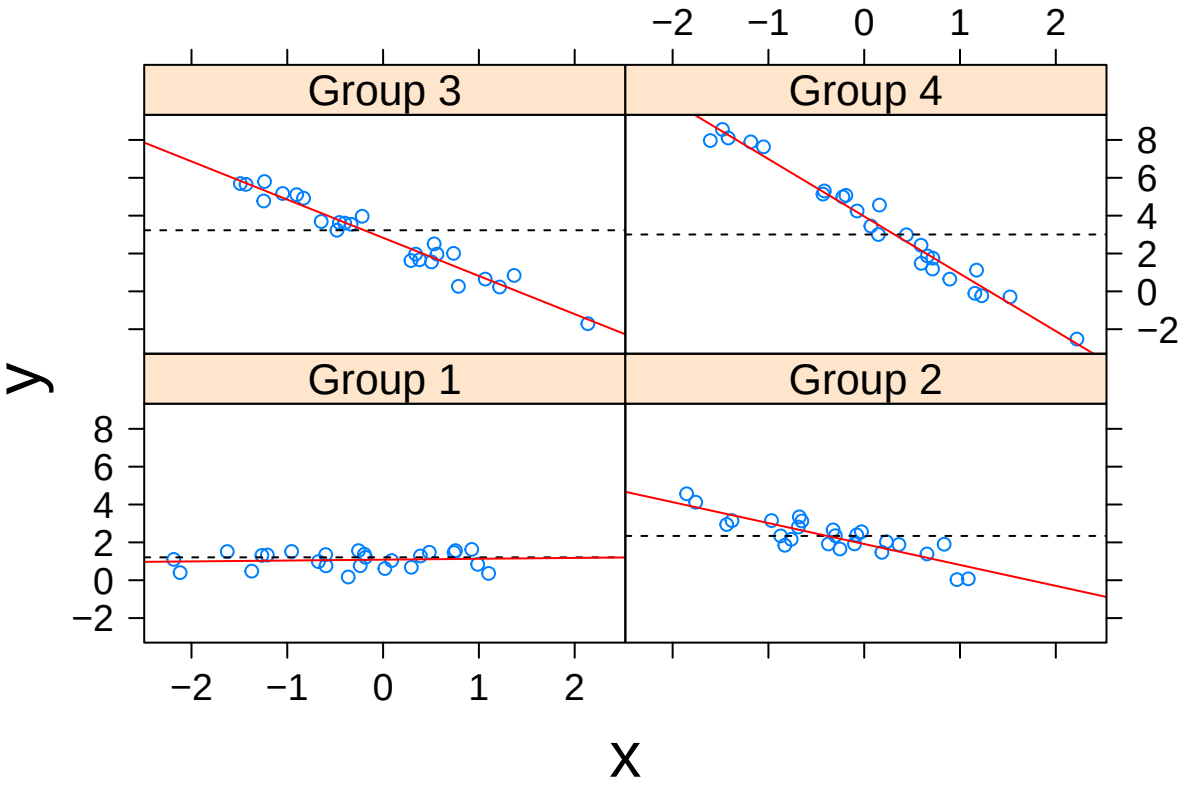


Lattice plotting system

Want more on the plot? Customize the panel function:


```
xyplot(y ~ x | f, panel = function(x, y, ...) {
    # call the default panel function for xyplot
    panel.xyplot(x, y, ...)
    # adds a horizontal line at the median
    panel.abline(h = median(y), lty = 2)
    # overlays a simple linear regression line
    panel.lmline(x, y, col = 2)
})
```

Lattice plotting system



Lattice plotting system

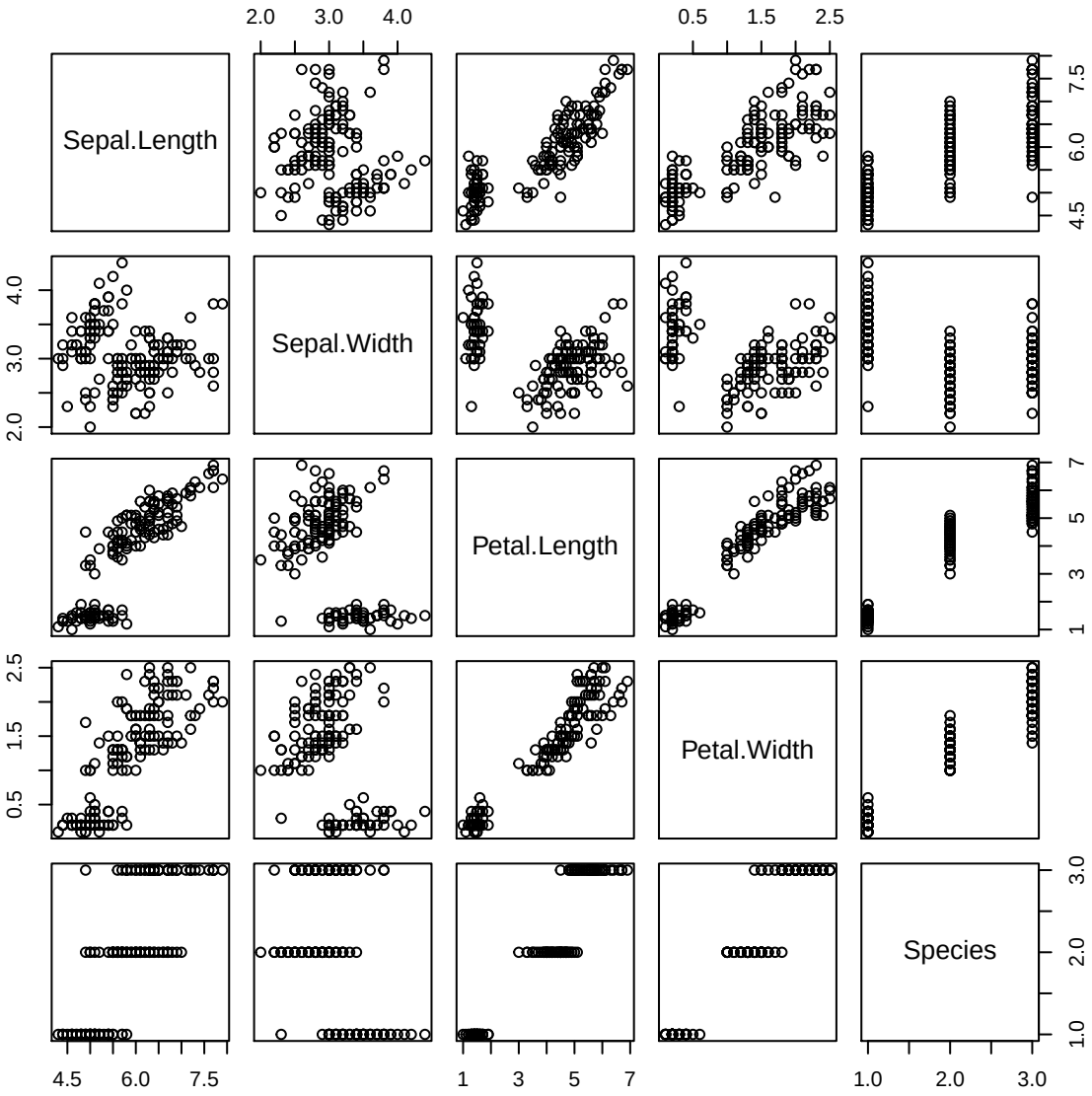
Plotting functions

- **xyplot()**: main function for creating scatterplots
- **bwplot()**: box and whiskers plots (box plots)
- **histogram()**: histograms
- **stripplot()**: box plot with actual points
- **dotplot()**: plot dots on "violin strings"
- **splom()**: scatterplot matrix (like pairs() in base plotting system)

- **levelplot()/contourplot()**: plotting image data

Very useful when we want a lot...

```
pairs(iris) ## iris is a data set in R
```



ggplot2

- An implementation of the **Grammar of Graphics by Leland Wikinson**
- Written by Hadley Wickham (while he was a graduate student as Iowa State)
- A **"third"** graphics system for R (along with base and lattice)
  Available from CRAN via install.packages()
  web site: http://ggplot2.org (better documentation)

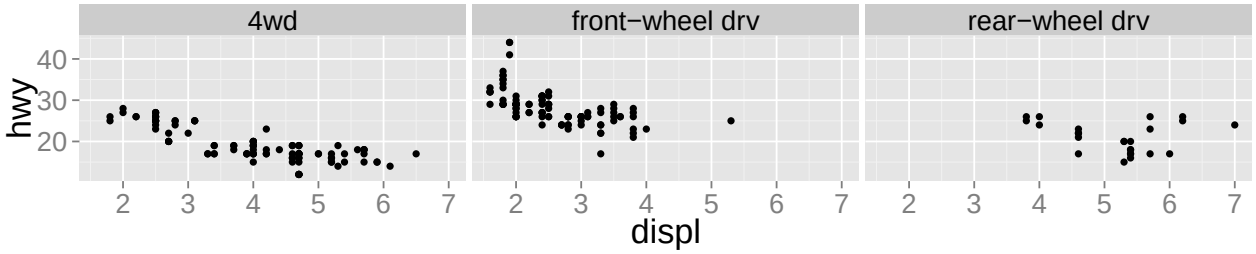
- Grammar of graphics represents the abstraction of graphics ideas/objects
  Think **"verb", "noun", "adjective"** for graphics
  "Shorten" the distance from mind to page
- Two main functions:
  qplot() hides what goes on underneath, which is okay for most operations
  ggplot() is the core function and very flexible for doing this qplot() cannot do

---

qplot function

The qplot() function is the analog to plot() but with many build-in features
Syntax somewhere in between base/lattice
Difficult to be customized (don't bother, use full ggplot2 power in that case)

```
## install.packages("ggplot2") # need to install ggplot2 first
library(ggplot2) # load the library
levels(mpg$drv) <- c("4wd", "front-wheel drv", "rear-wheel drv") # rename the levels
theme_set(theme_gray(base_size = 20)) # change text size
qplot(displ, hwy, data = mpg, facets = .~drv)
```



---

ggplot function

When building plots in ggplot2 (ggplot, rather than using qplot)
The **"artist's palette"** model may be the closest analogy
Plots are built up in layers
Step I: Input the data
noun: the data

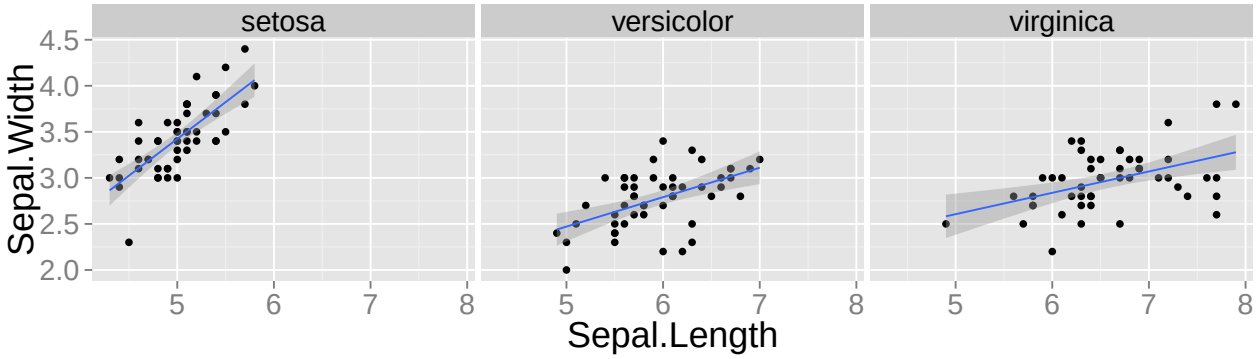
```
library(ggplot2) ## need to install and load this library
g <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) ## this would not show you add plot
```

---

ggplot function

- **Step II: Add layers**
  adjective: describe the type of plot you will produce.
  geom_smooth(method="lm"): add linear regression line;
  geom_grid(. ~ Species): lay out panels in a grid by Species;
  theme(): revision of the appearance. (It is actually the step III)


```
g <- g + geom_point() + geom_smooth(method = "lm") + facet_grid(. ~ Species) +
    theme(text = element_text(size = 20))
g
```

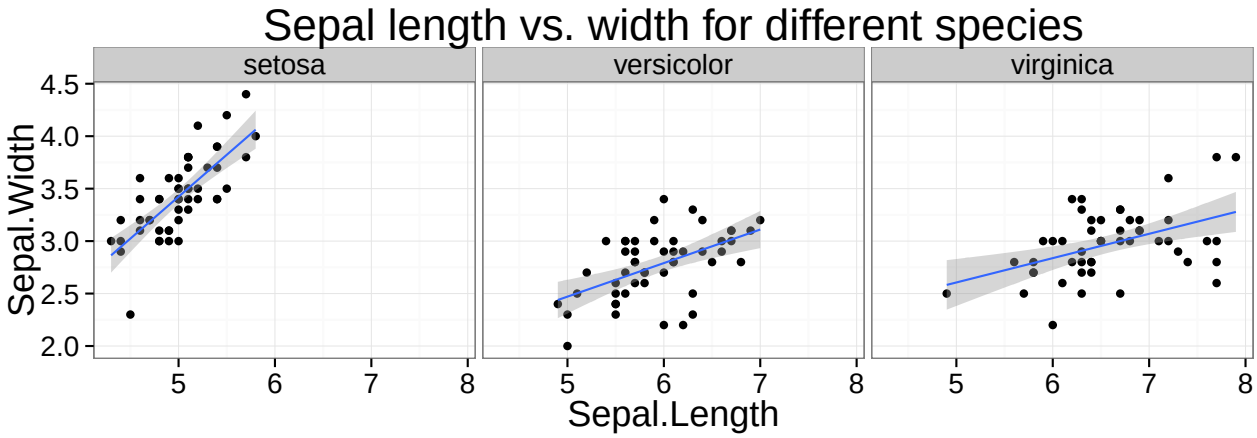


---

ggplot function

- **Step III: Add metadata and annotation**
  adjective: control the mapping between data and aesthetics.
  ggtitle(): add the title;
  theme_bw: choose the classic dark-on-light ggplot2 theme.

```
g + ggtitle("Sepal length vs. width for different species") + theme_bw() +
    theme(text = element_text(size = 18)) # add "verb"
```



Great documentation

Great **documentation** of ggplot with all functions in **step II** and **III** and demos:
http://docs.ggplot2.org/current/

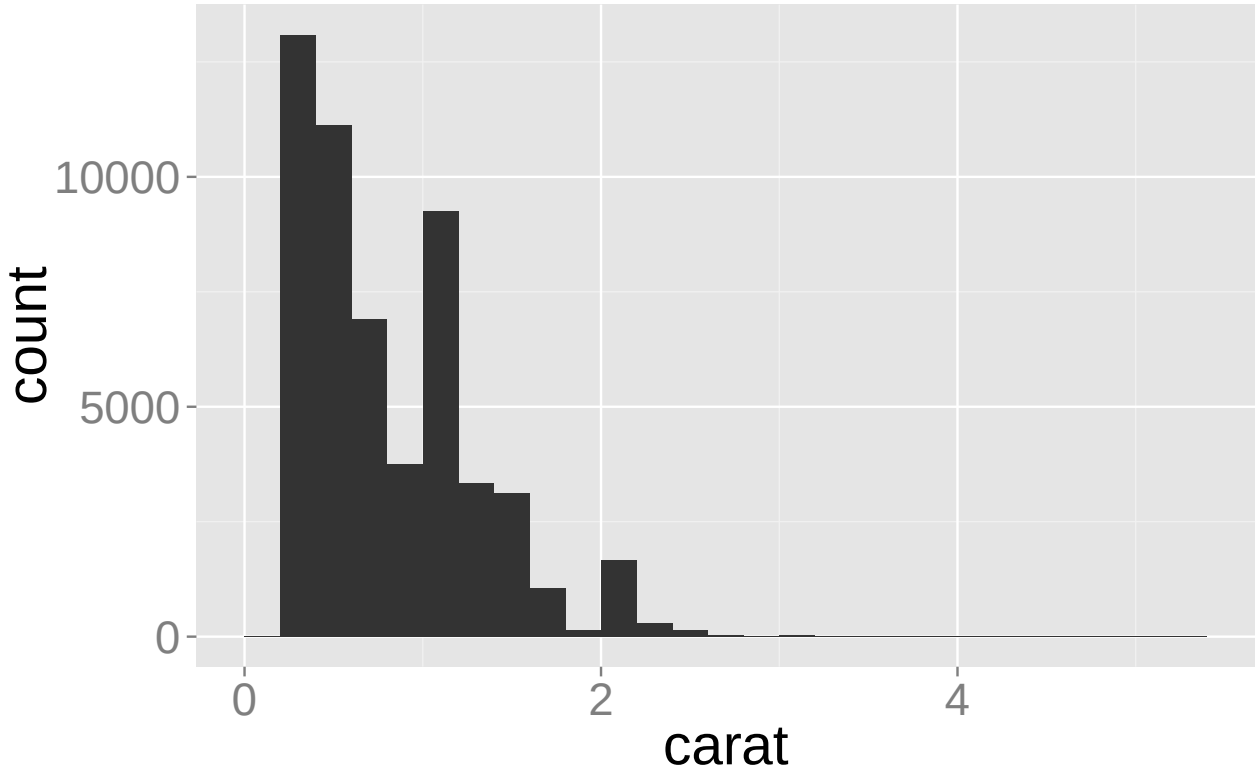
Overview of the diamond data

```
data(diamonds) # loading diamonds data set
head(diamonds, 9) # first few rows of diamond data set
```

```
##   carat       cut color clarity depth table price    x    y    z
## 1  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
## 2  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
## 3  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
## 4  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
## 5  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
## 6  0.24 Very Good     J    VVS2  62.8    57   336 3.94 3.96 2.48
## 7  0.24 Very Good     I    VVS1  62.3    57   336 3.95 3.98 2.47
## 8  0.26 Very Good     H     SI1  61.9    55   337 4.07 4.11 2.53
## 9  0.22      Fair     E     VS2  65.1    61   337 3.87 3.78 2.49
```

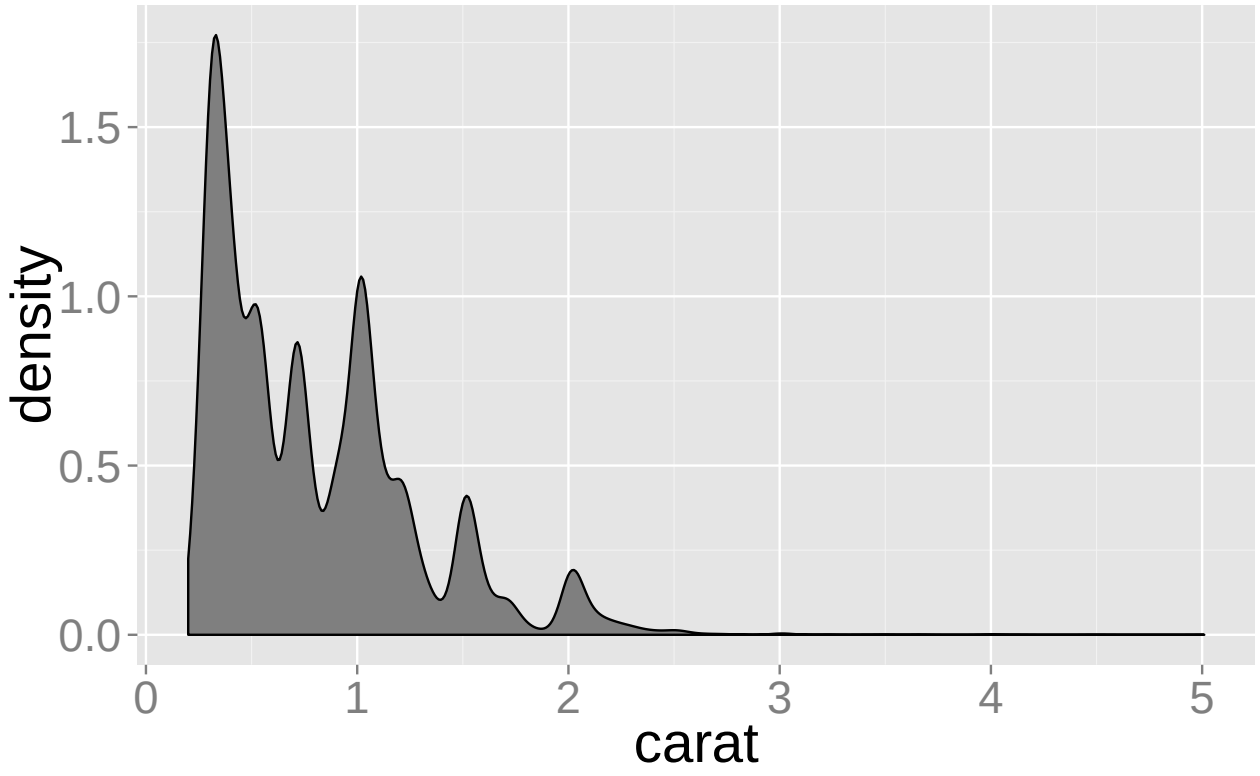
Histogram of carat

```
library(ggplot2)
ggplot(data=diamonds) + geom_histogram(aes(x=carat), binwidth=1/5) +
    theme(text = element_text(size = 25))
```



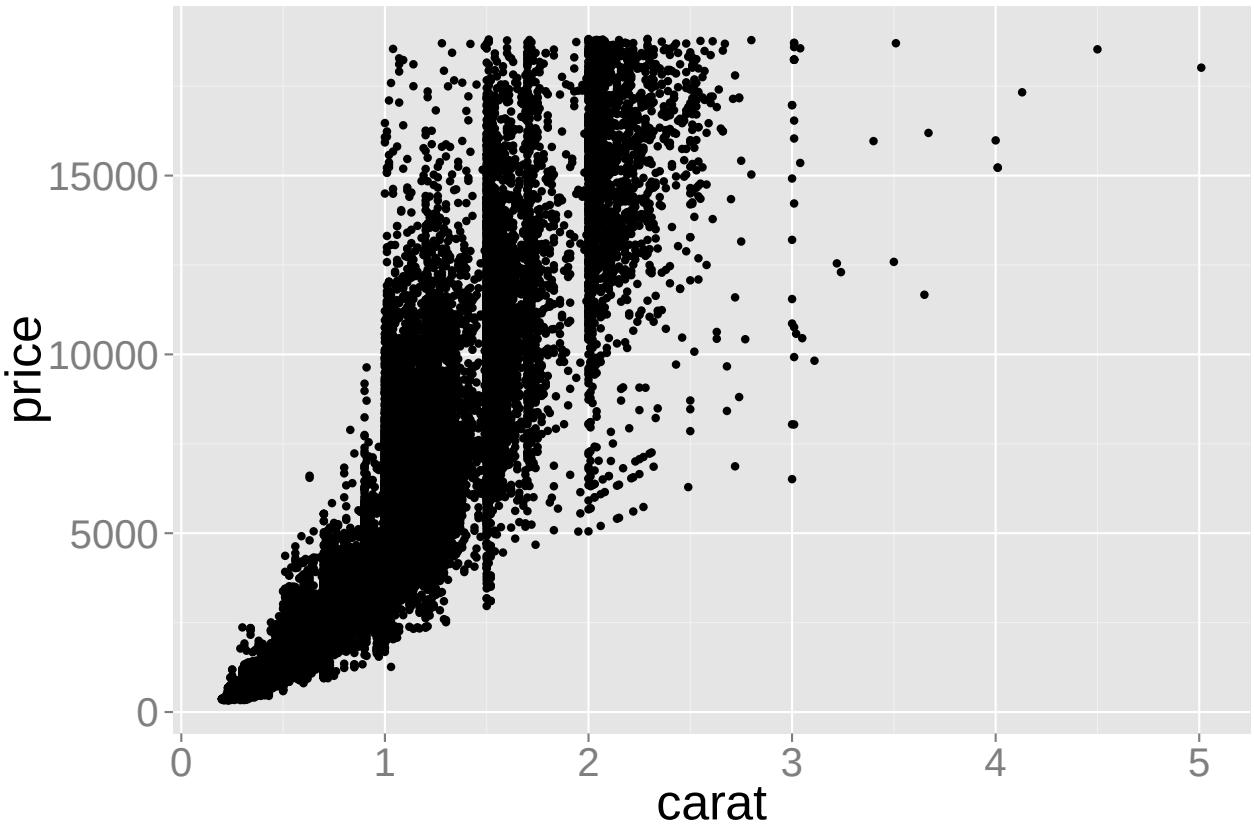
Density plot of carat

```
ggplot(data=diamonds) +
geom_density(aes(x=carat),fill="gray50")+
    theme(text = element_text(size = 25))
```



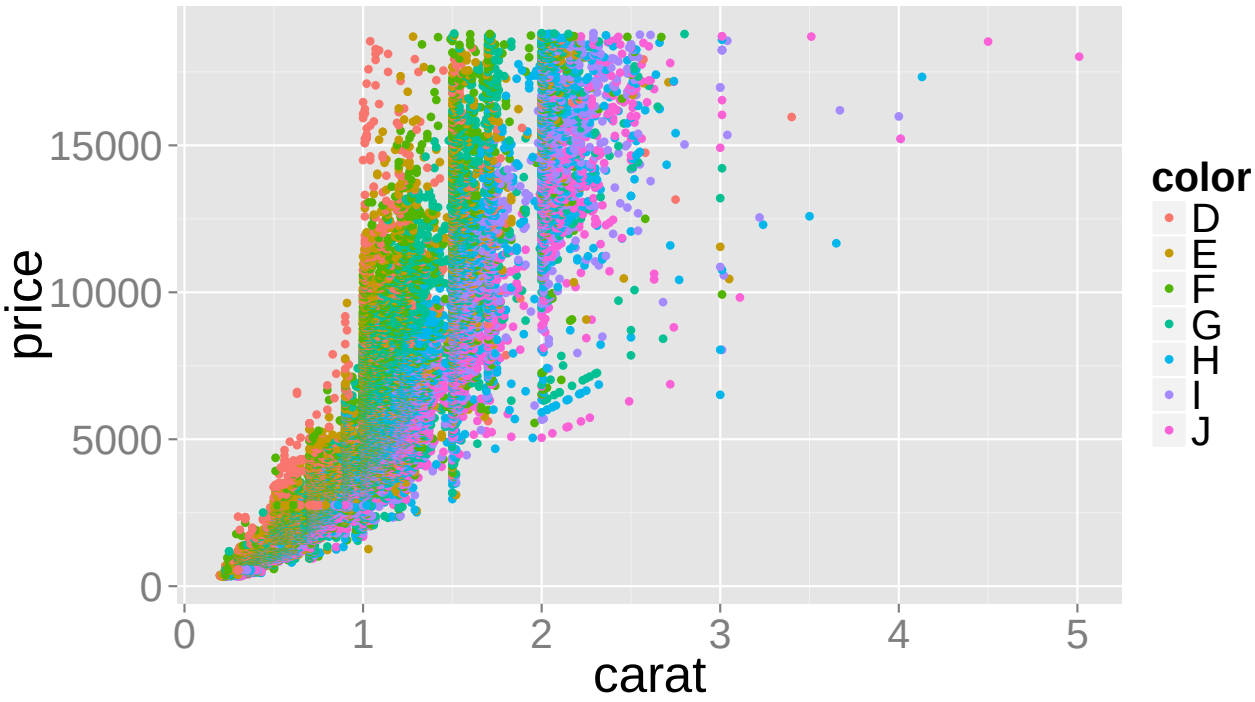
Scatter plots (carat vs. price)

```
ggplot(diamonds, aes(x=carat,y=price)) + geom_point() +
    theme(text = element_text(size = 25))
```

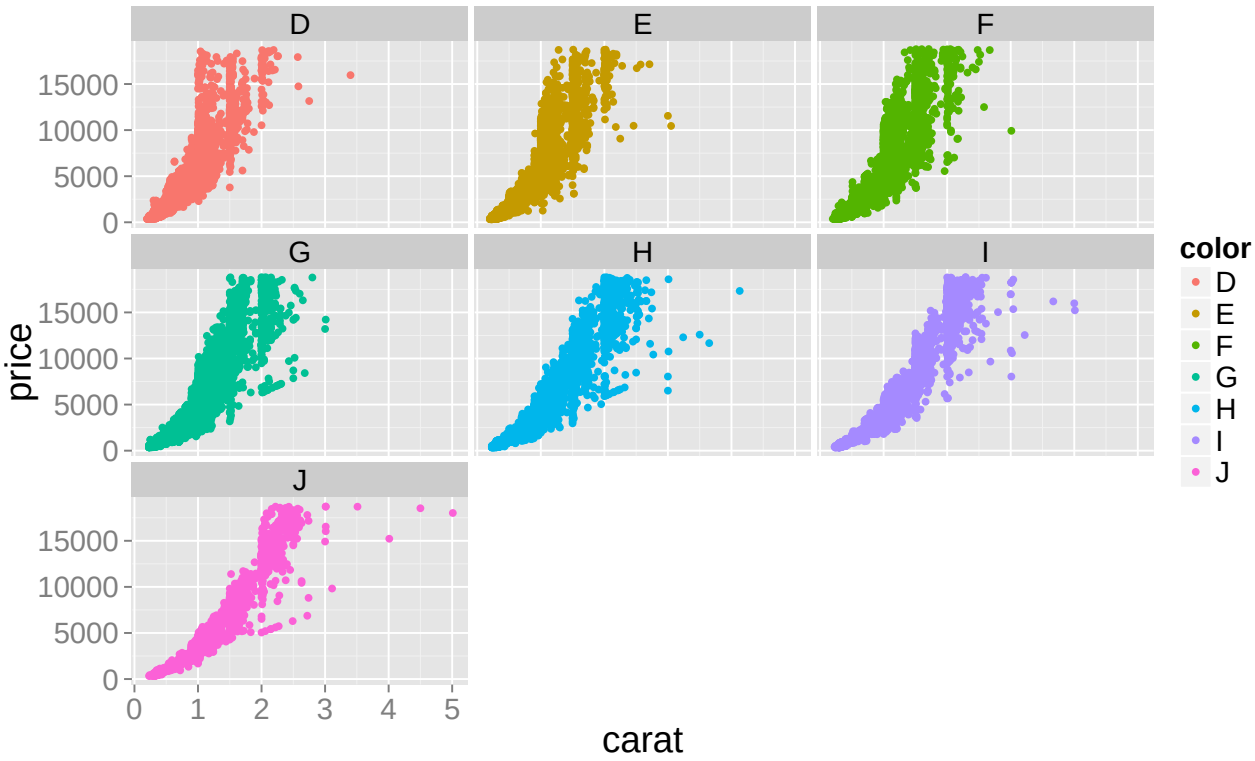
Carat with colors

```
g = ggplot(diamonds, aes(x=carat, y=price)) # saving first layer as variable
# rendering first layer and adding another layer
g + geom_point(aes(color=color)) + theme(text = element_text(size = 25))
```

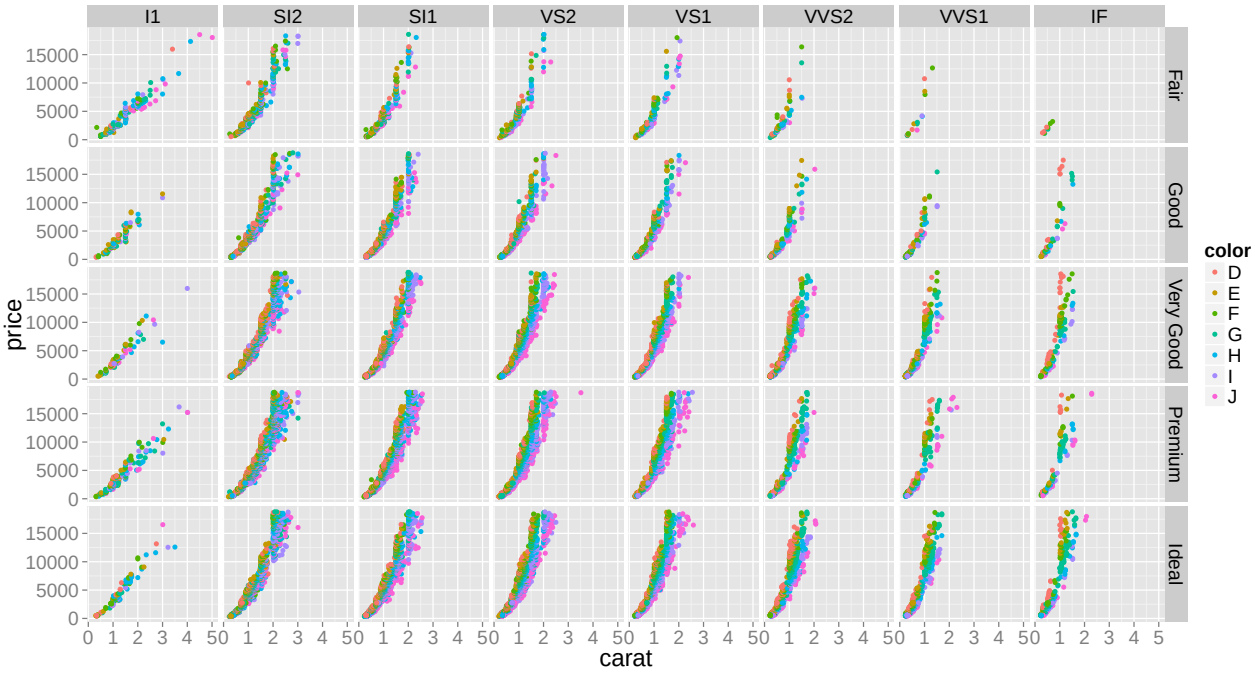



Carat with colors (more details)

```
g + geom_point(aes(color=color)) + facet_wrap(~color) +
    theme(text = element_text(size = 20))
```



Let's consider cut and clarity



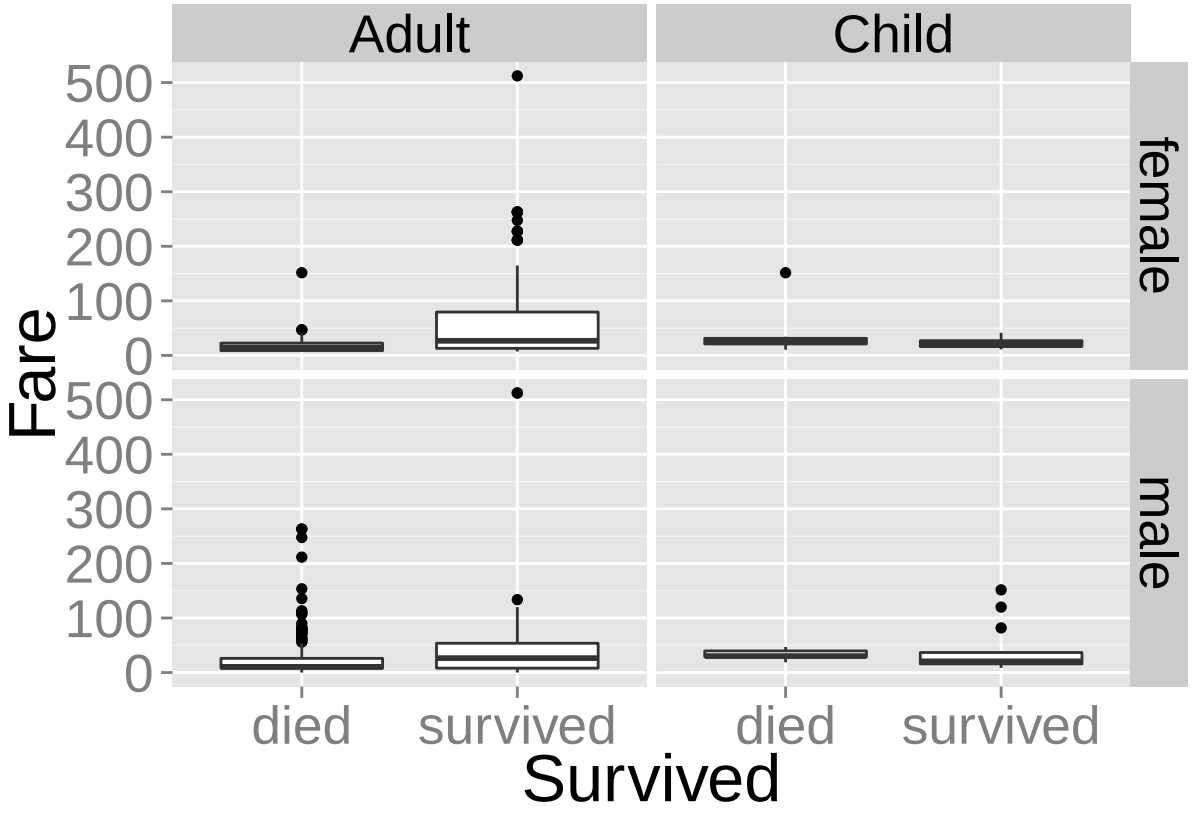
Your trun!

What is your knowledge of diamond's price after exploring this data?

Fare matters?

```
library(ggplot2)
ggplot(titanic_with_child_column, aes(y=Fare, x=Survived)) +
    geom_boxplot() + facet_grid(Gender~Child) +
    theme(text = element_text(size = 25))
```

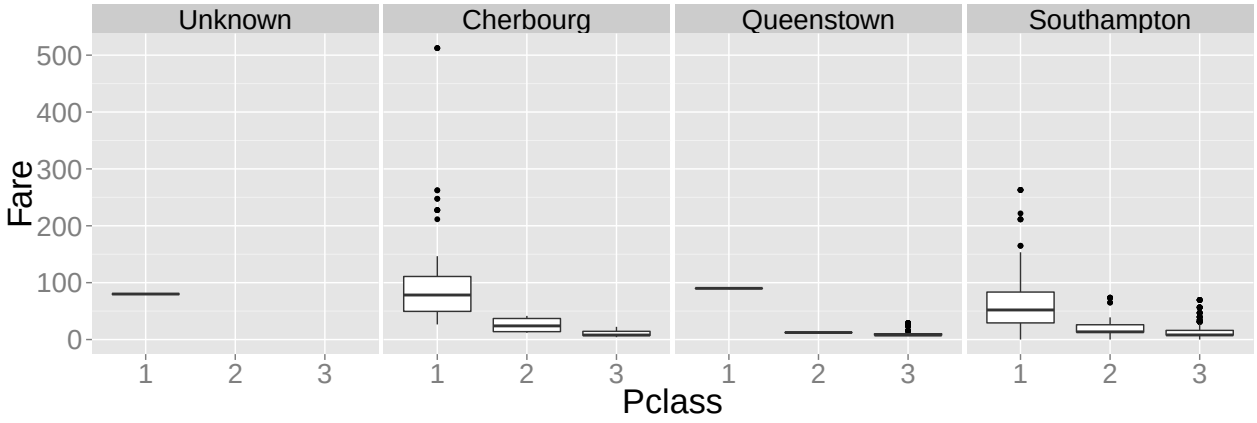

Fare matters?



How about fare, ship class, port embarkation?

```
library(ggplot2)
titanic$Pclass = as.factor(titanic$Pclass)
ggplot(titanic, aes(y=Fare, x=Pclass)) + geom_boxplot() +
    facet_grid(~Embarked) + theme(text = element_text(size = 25))
```

How about fare, ship class, port embarkation?



Question?

