Introduction to R, Exploratory Data Analysis

Chihara-Hesterberg

July 2022

Type commands after the prompt > and then press the <ENTER> key.

Note: anything after the pound symbol # is a comment—explanatory text that is not executed.

```
4*9           # Simple arithmetic
```

```
## [1] 36
```

Create a sequence incrementing by 1:

```
20:30
```

```
##  [1] 20 21 22 23 24 25 26 27 28 29 30
```

We will create an object called dog and assign it the values 1, 2, 3, 4, 5. The symbol <- is the assignment operator.

```
dog <- 1:5
dog
```

```
## [1] 1 2 3 4 5
```

```
dog + 10
```

```
## [1] 11 12 13 14 15
```

```
3*dog
```

```
## [1]  3  6  9 12 15
```

```
sum(dog)         # 1+2+3+4+5
```

```
## [1] 15
```

The object dog is called a *vector*.

If you need to abort a command, press the escape key <**ESC**>. The up arrow key ↑ can be used to recall previous entries.

To obtain help on any of the commands, type the name of the command you wish help on:

```
?sum
```

```
## starting httpd help server ... done
```

Importing data

Data for the third edition of the textbook can be downloaded from the web site [linked phrase] (http://github.com/chihara/MathStatsResamplingR/Edition3/Data) or by installing the R package **resampledata3**.

For instance, let's start with the Flight Delays Case Study (see Chapter 1, **Case Studies**), of the text for a description of this data set). We use the read.csv command to import the data into our R workspace:

```r
library(resampledata)  #temporary - change to resampledata3
```

```
##
## Attaching package: 'resampledata'

## The following object is masked from 'package:datasets':
##
##     Titanic
```

```r
#Alternatively, if the data set has been downloaded to the working directory
#FlightDelays<- read.csv("FlightDelays.csv")
#View(FlightDelays)
```

The str command gives a compact display of the internal structure of the data frame:

```r
str(FlightDelays)
```

```
## 'data.frame':    4029 obs. of  10 variables:
##  $ ID          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Carrier     : Factor w/ 2 levels "AA","UA": 2 2 2 2 2 2 2 2 2 2 ...
##  $ FlightNo    : int  403 405 409 511 667 669 673 677 679 681 ...
##  $ Destination : Factor w/ 7 levels "BNA","DEN","DFW",..: 2 2 2 6 6 6 6 6 6 6 ...
##  $ DepartTime  : Factor w/ 5 levels "4-8am","8-Noon",..: 1 2 4 2 1 1 2 2 3 3 ...
##  $ Day         : Factor w/ 7 levels "Sun","Mon","Tue",..: 6 6 6 6 6 6 6 6 6 6 ...
##  $ Month       : Factor w/ 2 levels "May","June": 1 1 1 1 1 1 1 1 1 1 ...
##  $ FlightLength: int  281 277 279 158 143 150 158 160 160 163 ...
##  $ Delay       : int  -1 102 4 -2 -3 0 -5 0 10 60 ...
##  $ Delayed30   : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 1 2 ...
```

Alternatively, the summary gives numeric summaries of the variables:

```r
summary(FlightDelays)
```

```
##        ID        Carrier       FlightNo       Destination    DepartTime
##  Min.   :   1   AA:2906   Min.   :  71.0   BNA: 172    4-8am   : 699
##  1st Qu.:1008   UA:1123   1st Qu.: 371.0   DEN: 264    8-Noon  :1053
##  Median :2015             Median : 691.0   DFW: 918    Noon-4pm:1048
##  Mean   :2015             Mean   : 827.1   IAD:  55    4-8pm   : 972
##  3rd Qu.:3022             3rd Qu.: 787.0   MIA: 610    8-Mid   : 257
##  Max.   :4029             Max.   :2255.0   ORD:1785
##                                            STL: 225
##   Day       Month       FlightLength       Delay         Delayed30
##  Sun:551   May :1999   Min.   : 68.0   Min.   :-19.00   No :3432
##  Mon:630   June:2030   1st Qu.:155.0   1st Qu.: -6.00   Yes: 597
##  Tue:628               Median :163.0   Median : -3.00
##  Wed:564               Mean   :185.3   Mean   : 11.74
##  Thu:566               3rd Qu.:228.0   3rd Qu.:  5.00
##  Fri:637               Max.   :295.0   Max.   :693.00
##  Sat:453
```

Other commands to get basic information about the data frame include names, to view the names of the variables in FlightDelays:

```r
names(FlightDelays)
```

```
##  [1] "ID"           "Carrier"      "FlightNo"     "Destination"  "DepartTime"
##  [6] "Day"          "Month"        "FlightLength" "Delay"        "Delayed30"
```

To view the first 6 rows of the data, use the head command:

```r
head(FlightDelays)      #default, 6 rows
```

```
##   ID Carrier FlightNo Destination DepartTime Day Month FlightLength Delay
## 1  1      UA      403         DEN      4-8am Fri   May          281    -1
## 2  2      UA      405         DEN     8-Noon Fri   May          277   102
## 3  3      UA      409         DEN      4-8pm Fri   May          279     4
## 4  4      UA      511         ORD     8-Noon Fri   May          158    -2
## 5  5      UA      667         ORD      4-8am Fri   May          143    -3
## 6  6      UA      669         ORD      4-8am Fri   May          150     0
##   Delayed30
## 1        No
## 2       Yes
## 3        No
## 4        No
## 5        No
## 6        No
```

```r
head(FlightDelays, 4)   #first 10 rows
```

```
##   ID Carrier FlightNo Destination DepartTime Day Month FlightLength Delay
## 1  1      UA      403         DEN      4-8am Fri   May          281    -1
## 2  2      UA      405         DEN     8-Noon Fri   May          277   102
## 3  3      UA      409         DEN      4-8pm Fri   May          279     4
## 4  4      UA      511         ORD     8-Noon Fri   May          158    -2
##   Delayed30
## 1        No
## 2       Yes
## 3        No
## 4        No
```

(What do you suppose the tail command does?)

The columns are the *variables*. There are two types of variables: *numeric*, for example, FlightLength and Delay and *factorI (also called* categorical*) (for example Carrier and DepartTime). The rows are called* observations* or *cases*.

To check the size (number of rows and columns) of the data frame, type

```r
dim(FlightDelays)    # dim = dimension
```

```
## [1] 4029   10
```

This tells us that there are 4029 observations and 10 columns.

Tables, bar charts and histograms

The factor variable Carrier in the FlightDelays data set assigns each flight to one of two *levels*: UA or AA. To obtain the summary of this variable, we use the $ operator to extract this variable from the FlightDelays data frame:

```r
table(FlightDelays$Carrier)
```

```
##
##   AA   UA
## 2906 1123
```

Remark R is **case-sensitive**! Carrier and carrier are considered different.

To create a contingency table summarizing the relationship between carrier (Carrier) and whether or not a flight was delayed by more than 30 minutes (Delayed30), type:

```
table(FlightDelays$Carrier, FlightDelays$Delayed30)
```

```
##
##        No  Yes
##   AA 2513  393
##   UA  919  204
```

To perform additional operations, we will first store the table object:

```
out <- table(FlightDelays$Carrier, FlightDelays$Delayed30)
out
```

```
##
##        No  Yes
##   AA 2513  393
##   UA  919  204
```

```
addmargins(out)  # table with row/column sums
```

```
##
##          No  Yes  Sum
##   AA   2513  393 2906
##   UA    919  204 1123
##   Sum 3432  597 4029
```

The proportions command gives joint or conditional distributions:

```
proportions(out)           #joint distribution (sum of all cells = 1)
```

```
##
##              No        Yes
##   AA 0.62372797 0.09754281
##   UA 0.22809630 0.05063291
```

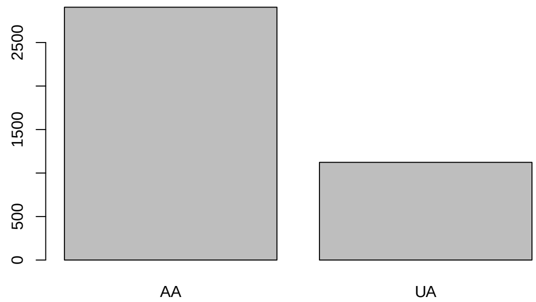
```
proportions(out, 1)        #conditional distributions (row sums = 1)
```

```
##
##             No       Yes
##   AA 0.8647626 0.1352374
##   UA 0.8183437 0.1816563
```

Thus, 9.8% of flights were American Airlines and delayed by more than 30 minutes, whereas of all American Airline flights, 13.5% were delayed by more than 30 minutes

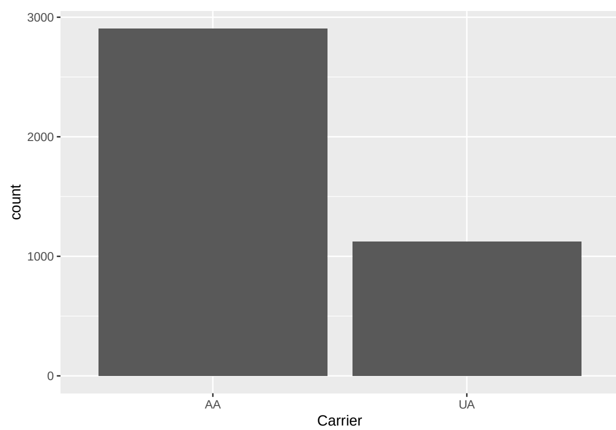
To visualize the distribution of a factor variable, create a *bar chart*:

```
barplot(table(FlightDelays$Carrier))
```



The barplot command is part of base R, that is, it comes with the default installation of R. Researchers have developed *packages* that enhance many of the basic R commands. One such package is the **ggplot2** package.

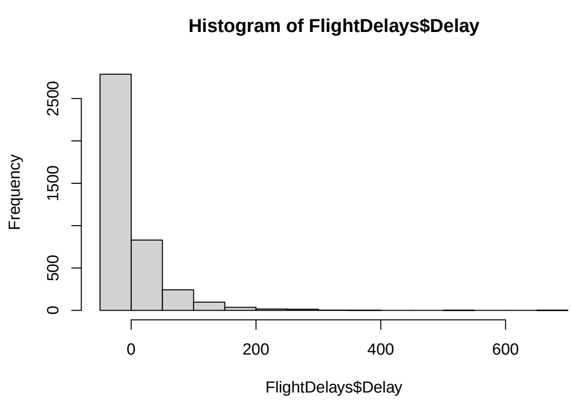
```
library(ggplot2)

ggplot(FlightDelays, aes(x = Carrier)) + geom_bar()
```



Note that the syntax is different from the base R command for a bar plot: in particular, we do not first have to create a table.

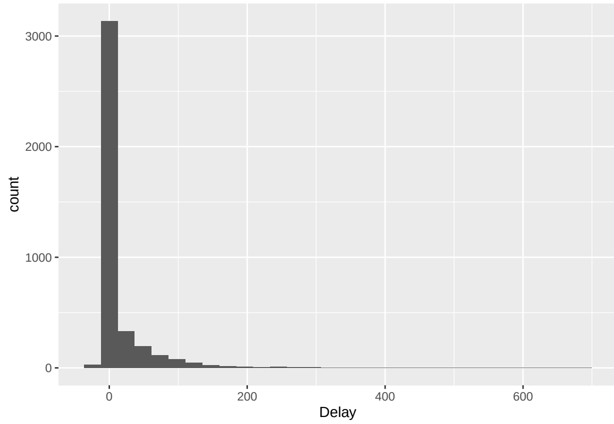
To visualize the distribution of a numeric variable, create a histogram.

```
hist(FlightDelays$Delay)     # base R
```



```
ggplot(FlightDelays, aes(x = Delay)) + geom_histogram()   # ggplot2
```

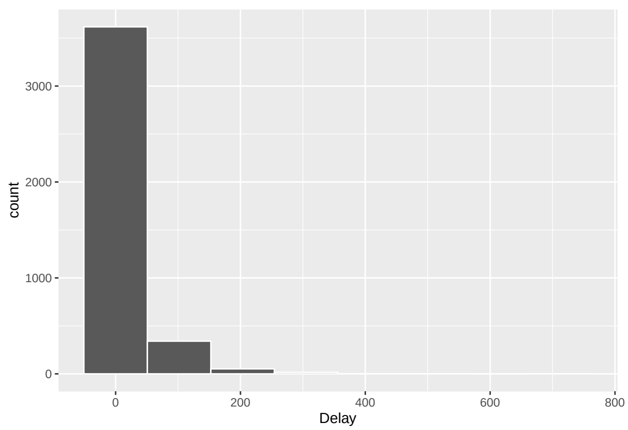
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The shape of the distribution of this variable is *right-skewed*.

The default number of bins for the **ggplot2** histogram is 30. We'll change this here:

```
ggplot(FlightDelays, aes(x = Delay)) + geom_histogram(bins = 8, color = "white")
```



Numeric Summaries

Because it is a bit cumbersome to use the syntax FlightDelays$Delay each time we want to work with the Delay variable, we will extract this variable and store it in the (vector) object delay:

```
delay <- FlightDelays$Delay
head(delay)
```

```
## [1]  -1 102   4  -2  -3   0
```

```
mean(delay)
```

```
## [1] 11.7379
```

```
mean(delay, trim = .05)   #trimmed mean
```

```
## [1] 5.038875
```

```
median(delay)
```

```
## [1] -3
```

Other basic statistics:

```
max(delay)
```

```
## [1] 693
```

```
min(delay)
```

```
## [1] -19
```

```
range(delay)
```

```
## [1] -19 693
```

```
var(delay)          #variance
```

```
## [1] 1733.098
```

```
sd(delay)           #standard deviation
```

```
## [1] 41.6305
```

```
quantile(delay)     #quartiles
```

```
##   0%  25%  50%  75% 100%
##  -19   -6   -3    5  693
```

If you need the population variance (that is, denominator of 1/n instead of 1/(n-1)),

```
n <- length(delay)
(n-1)/n*var(delay)
```

```
## [1] 1732.668
```

Statistics with grouping by a categorical variable

To find statistics for a numeric variable grouped by a categorical variable, one option is to use the tapply command

```
tapply(FlightDelays$Delay, FlightDelays$Carrier, mean)
```

```
##       AA       UA
## 10.09738 15.98308
```

```
tapply(FlightDelays$Delay, FlightDelays$Carrier, sd)
```

```
##       AA       UA
## 40.08063 45.13895
```

Thus, the mean length of an American Airlines flight delay was 10.1 minutes compared to 15.98 minutes for United Airlines. The standard deviations for AA and UA were 40.08 m and 45.1 m, respectively.

Another approach is to use the commands in the **dplyr** package

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
FlightDelays %>% group_by(Carrier) %>% summarize(mean(Delay), sd(Delay))
```

```
## # A tibble: 2 x 3
##   Carrier `mean(Delay)` `sd(Delay)`
##   <fct>           <dbl>       <dbl>
## 1 AA               10.1        40.1
## 2 UA               16.0        45.1
```

The %>% is the piping operator in the **dplyr** package: it takes the data set FlightDelays and "pipes" it to the command group by. The group_by command groups the observations by Carrier, that is by AA or UA. Then we pipe this information to the summarize command and compute the mean and standard deviation of the Delay variable for each of AA and UA flights.

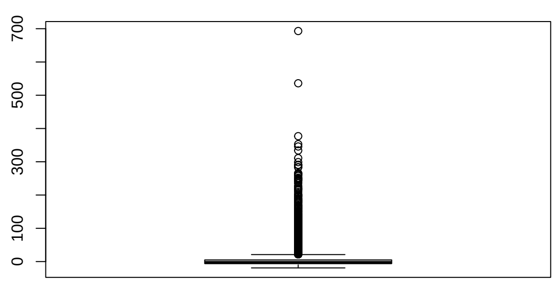
Boxplots

Boxplots give a visualization of the 5-number summary of a variable.

```r
summary(delay)          #numeric summary
```

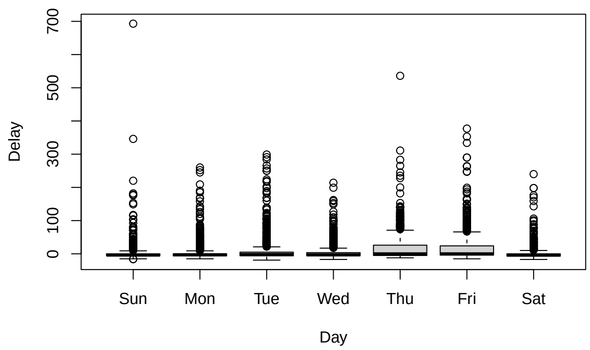
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -19.00   -6.00   -3.00   11.74    5.00  693.00
```

```r
boxplot(delay)
```

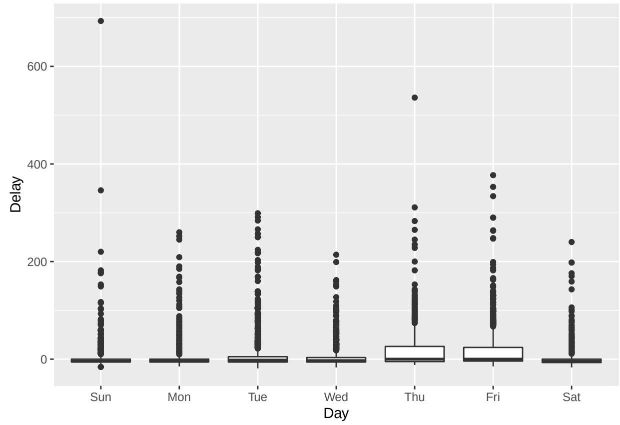


To compare the distribution of a numeric variable across the levels of a categorical (factor) variable:

```r
boxplot(Delay ~ Day, data = FlightDelays)          # base R
```



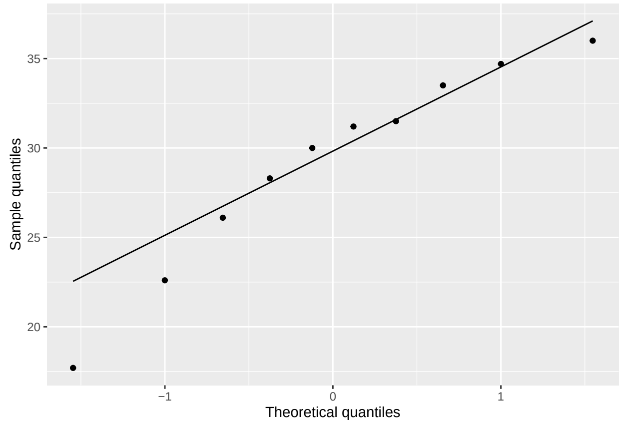

```
ggplot(FlightDelays, aes(x = Day, y = Delay)) + geom_boxplot()   # ggplot2
```



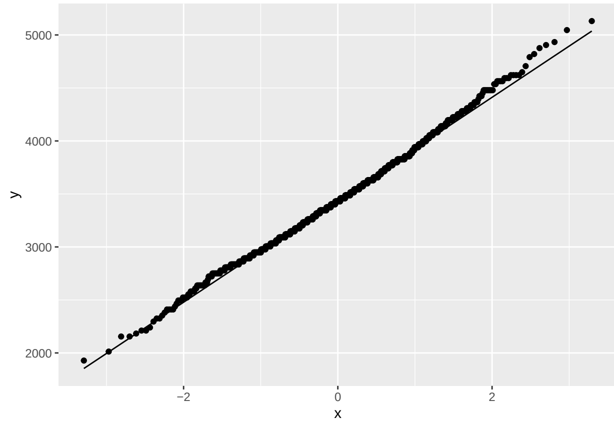
Normal quantile plots

We create normal quantile plots to see if it is plausible that data come from a normal distribution. The ggplot command requires a data frame for it's first argument.

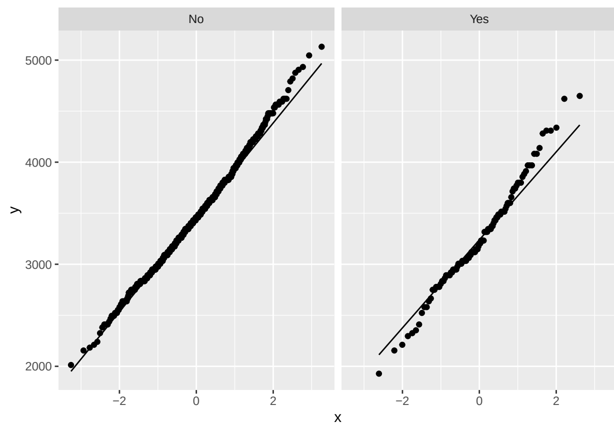
```
x <- c(17.7, 22.6, 26.1, 28.3, 30, 31.2, 31.5, 33.5, 34.7, 36)
df <- data.frame(x = x)
ggplot(df, aes(sample = x)) + geom_qq() + geom_qq_line() +
  labs(x = "Theoretical quantiles", y = "Sample quantiles")
```



```
ggplot(NCBirths2004, aes(sample = Weight))+ geom_qq() +
  geom_qq_line()
```

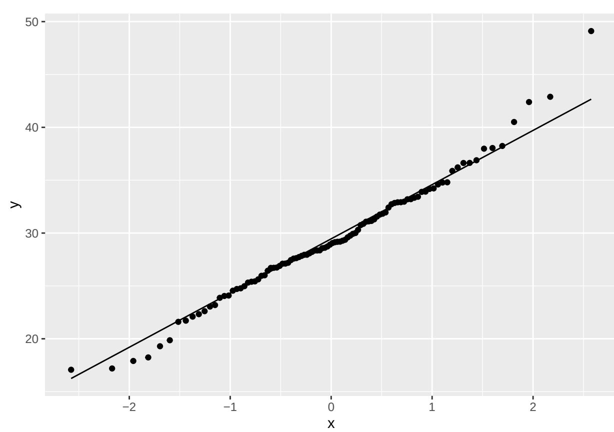


```
ggplot(NCBirths2004, aes(sample = Weight)) + geom_qq() +
  geom_qq_line() + facet_grid( ~ Smoker)
```



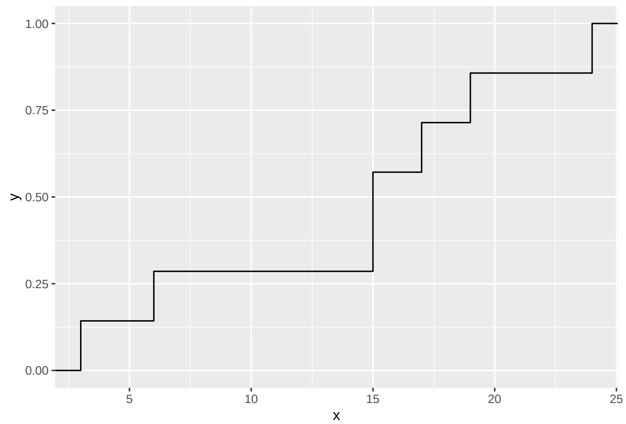
Let's generate a random sample of size 100 from the normal distribution N(30, 6) and create the normal-quantile plot.

```
my.sample <- rnorm(100, 30, 6)
df <- data.frame(my.sample)      #create data frame
ggplot(df, aes(sample = my.sample)) + geom_qq() + geom_qq_line()
```

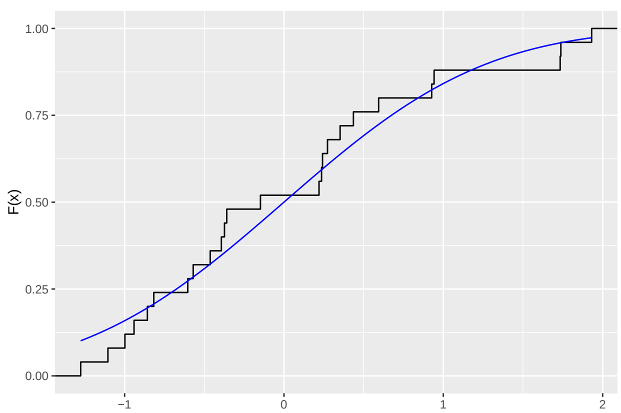


Empirical Cumulative Distribution Function (ECDF)

```
x <- c(3, 6, 15, 15, 17, 19, 24)
df <- data.frame(x = x)
ggplot(df, aes(x = x)) + stat_ecdf()
```

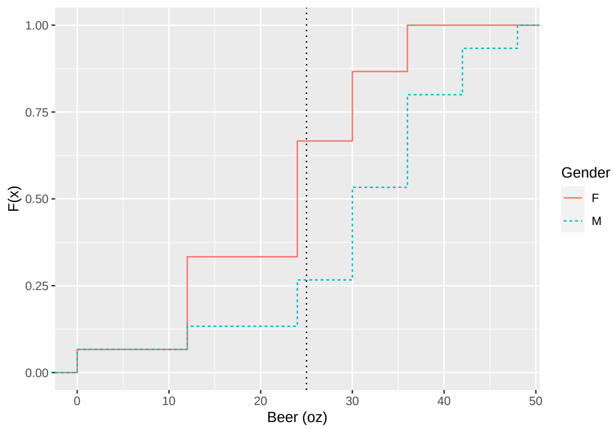



```
df <- data.frame(x = rnorm(25))      # random sample from N(0,1)
ggplot(df, aes(x = x)) + stat_ecdf() +
  stat_function(fun = pnorm, color = "blue") +
  labs(x="", y = "F(x)")
```



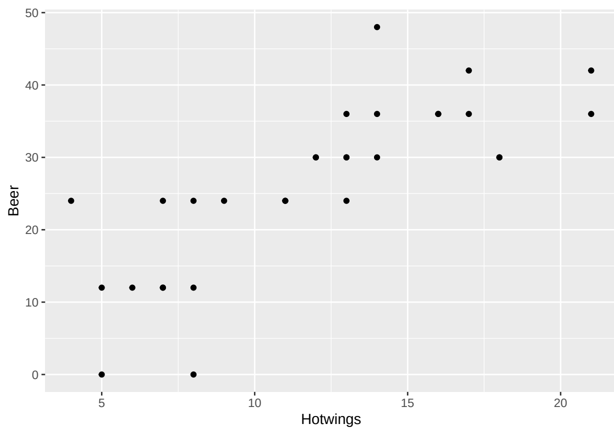
Using the data from the Beer and Hotwings case study:

```
ggplot(Beerwings, aes(x = Beer, linetype = Gender, color = Gender)) +
  stat_ecdf() +  labs(x = "Beer (oz)", y = "F(x)") +
  geom_vline(xintercept = 25, lty = 3)
```



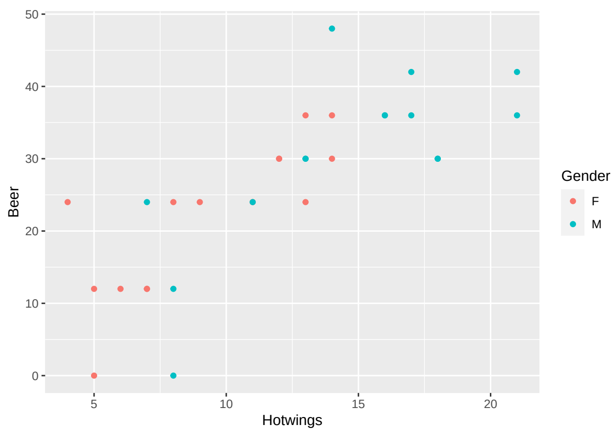
Scatter plots

```
ggplot(Beerwings, aes(x = Hotwings, y = Beer)) +
  geom_point()
```



We can also distinguish the two genders by adding the color aesthetic:

```
ggplot(Beerwings, aes(x = Hotwings, y = Beer,
       color = Gender)) + geom_point()
```



Misc. Remarks

- Functions in R are called by typing their name followed by arguments surrounded by *parentheses*: ex. hist(delay). Typing a function name without parentheses will give the code for the function.

```
sd
```

```
## function (x, na.rm = FALSE)
## sqrt(var(if (is.vector(x) || is.factor(x)) x else as.double(x),
##     na.rm = na.rm))
## <bytecode: 0x000001687647a828>
## <environment: namespace:stats>
```

- We saw earlier that we can assign names to data (we created a vector called dog.) Names can be any length, must start with a letter, and may contain letters or numbers:

```
fish25 <- 10:35
fish25
```

```
##  [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
## [26] 35
```

Certain names are **reserved** so be careful to not use them: cat, c, t, T, F,…

To be safe, before making an assignment, type the name:

```
whale
```

If you get the output Problem: Object "whale" not found, then it is safe to use whale!

```
whale <- 200
objects()
```

```
## [1] "delay"     "df"        "dog"       "fish25"    "my.sample" "n"
## [7] "out"       "whale"     "x"
```

```
rm(whale)
objects()
```

```
## [1] "delay"     "df"        "dog"       "fish25"    "my.sample" "n"
## [7] "out"       "x"
```

- In general, R is space-insensitive.

```
3 +4
```

```
## [1] 7
```

```
3+   4
```

```
## [1] 7
```

```
mean(3+5)
```

```
## [1] 8
```

```
mean ( 3 + 5 )
```

```
## [1] 8
```

BUT, the assignment operator must not have spaces! <- is different from < , -

- To quit, type q() at the prompt.

You will be given an option to save the workspace.

If you select **Yes**: all objects created in this session are saved to your working directory so that the next time you start up R, if you load this working directory, these objects will still be available. You will not have to re-import FlightDelays, for instance, nor recreate delay.

You can, for back-up purposes, save data to an external file/disk by using, for instance, the write.csv command. See the help file for more information.