QDA Lab 1 part 1

Isabel Sassoon & Sarath Dantu

In this lab we be going through the following topics:

1. Introduction to R
2. Exploratory Data Analysis

1. Introduction to R

For CS5701, CS5702, and CS5608 you will be using R for data analysis. So let's start with some basics of R. An elaborate Introduction to R, is available to you from the link. In this session, I will give you a crash course on:

- Objects
- Functions

1.1 Object

Numbers

```
int_a <- 4
int_b <- 2
float_a <- 4.2
float_b <- 6.9
```

```
int_a+int_b
```

```
## [1] 6
```

```
int_a*int_b
```

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

```
int_a/int_b
```

```
## [1] 2
```

```
int_a%%int_b
```

```
## [1] 0
```

```
int_a^int_b
```

```
## [1] 16
```

Characters

```
char_c <- "John"
```

Logical

```
logic_d <- "FALSE" # logic_d <- F
logic_e <- "TRUE" # logic_e <- T
```

Lists/vectors

```
vector_f <- c(2,2,2,1,4,5,3,2,4,4,3)
```

```
int_a * vector_f
```

```
##  [1]  8  8  8  4 16 20 12  8 16 16 12
```

Data frame

```
seq_x <- seq(1:20)
seq_y <- seq(21:40)
data_frame<-as.data.frame(cbind(seq_x,seq_y)) # this is coercion (http://people.brunel.ac.uk/~csstssd/i
data_frame
```

```
##    seq_x seq_y
## 1      1     1
## 2      2     2
## 3      3     3
## 4      4     4
## 5      5     5
## 6      6     6
## 7      7     7
## 8      8     8
## 9      9     9
## 10    10    10
## 11    11    11
## 12    12    12
## 13    13    13
## 14    14    14
## 15    15    15
```

```
## 16    16    16
## 17    17    17
## 18    18    18
## 19    19    19
## 20    20    20
```

```
colnames(data_frame) <- c("X","Y")
data_frame
```

```
##     X  Y
## 1   1  1
## 2   2  2
## 3   3  3
## 4   4  4
## 5   5  5
## 6   6  6
## 7   7  7
## 8   8  8
## 9   9  9
## 10 10 10
## 11 11 11
## 12 12 12
## 13 13 13
## 14 14 14
## 15 15 15
## 16 16 16
## 17 17 17
## 18 18 18
## 19 19 19
## 20 20 20
```

```
data_frame$x_plus_y <- data_frame$X +data_frame$Y
data_frame
```

```
##     X  Y x_plus_y
## 1   1  1        2
## 2   2  2        4
## 3   3  3        6
## 4   4  4        8
## 5   5  5       10
## 6   6  6       12
## 7   7  7       14
## 8   8  8       16
## 9   9  9       18
## 10 10 10       20
## 11 11 11       22
## 12 12 12       24
## 13 13 13       26
## 14 14 14       28
## 15 15 15       30
## 16 16 16       32
## 17 17 17       34
## 18 18 18       36
```

```
## 19 19 19       38
## 20 20 20       40
```

```
# Can you spot the difference between line 73 and 74?
data_frame$iseven <- ifelse(data_frame$X%%2==0,"TRUE","FALSE")
data_frame$isodd <- ifelse(data_frame$X%%3==0,TRUE,FALSE)
data_frame
```

```
##     X  Y x_plus_y iseven isodd
## 1   1  1        2  FALSE FALSE
## 2   2  2        4   TRUE FALSE
## 3   3  3        6  FALSE  TRUE
## 4   4  4        8   TRUE FALSE
## 5   5  5       10  FALSE FALSE
## 6   6  6       12   TRUE  TRUE
## 7   7  7       14  FALSE FALSE
## 8   8  8       16   TRUE FALSE
## 9   9  9       18  FALSE  TRUE
## 10 10 10       20   TRUE FALSE
## 11 11 11       22  FALSE FALSE
## 12 12 12       24   TRUE  TRUE
## 13 13 13       26  FALSE FALSE
## 14 14 14       28   TRUE FALSE
## 15 15 15       30  FALSE  TRUE
## 16 16 16       32   TRUE FALSE
## 17 17 17       34  FALSE FALSE
## 18 18 18       36   TRUE  TRUE
## 19 19 19       38  FALSE FALSE
## 20 20 20       40   TRUE FALSE
```

1.2 Function

```
add_function <- function(a,b){
  a <- a*a
  return(a+b)
}
add_function(90,80)
```

```
## [1] 8180
```

1.3 Essential tips:

- R is very sensitive to cases and ***syntax***.
- Objects are everything in R. Objects with parenthesis are ***functions*** and without are ***data types***.
- `#` in front of a sentence or a word is treated as a comment and is not executed.
- For help in R, `help("command")`

---

2. Exploratory Data Analysis

In this lab session the key objectives are to use R through RStudio to read in a small set of data and to perform exploratory data analysis (EDA). In R, it is possible to read data from different file formats:

- manual input
- plain text
- delimited text - such as .csv/.tsv files
- spreadsheets
- SAS formats
- anything else that you can imagine off...

Now lets see how to do some descriptive data analysis. This will also introduce you to ggplot and its syntax.

2.1 Categorical variables: Numerical summaries

If you recall from the lecture the numerical summary for a categorical variable is a frequency table. These are very easy to produce with R functions.

```
library(ggplot2,gridExtra)
#this line below reads the csv straight into a data frame called worms
worms <- read.csv("data/worms.csv")

#this line below prints the head (or top) of the data frame
head(worms,n=10)
```

```
##         Field.Name Area Slope Vegetation Soil.pH  Damp Worm.density
## 1      Nashs.Field  3.6    11  Grassland     4.1 FALSE            4
## 2   Silwood.Bottom  5.1     2     Arable     5.2 FALSE            7
## 3    Nursery.Field  2.8     3  Grassland     4.3 FALSE            2
## 4      Rush.Meadow  2.4     5     Meadow     4.9  TRUE            5
## 5  Gunness.Thicket  3.8     0      Scrub     4.2 FALSE            6
## 6         Oak.Mead  3.1     2  Grassland     3.9 FALSE            2
## 7     Church.Field  3.5     3  Grassland     4.2 FALSE            3
## 8          Ashurst  2.1     0     Arable     4.8 FALSE            4
## 9      The.Orchard  1.9     0    Orchard     5.7 FALSE            9
## 10   Rookery.Slope  1.5     4  Grassland     5.0  TRUE            7
```

There are many additional packages that offer more sophisticated options. We will explore relevant ones later in the module. It is also possible to do frequency tables for more than one column at the same time (in effect creating a contingency table):

```
table(worms$Damp, worms$Vegetation)
```

```
##
##         Arable Grassland Meadow Orchard Scrub
##   FALSE      3         8      0       1     2
##   TRUE       0         1      3       0     2
```

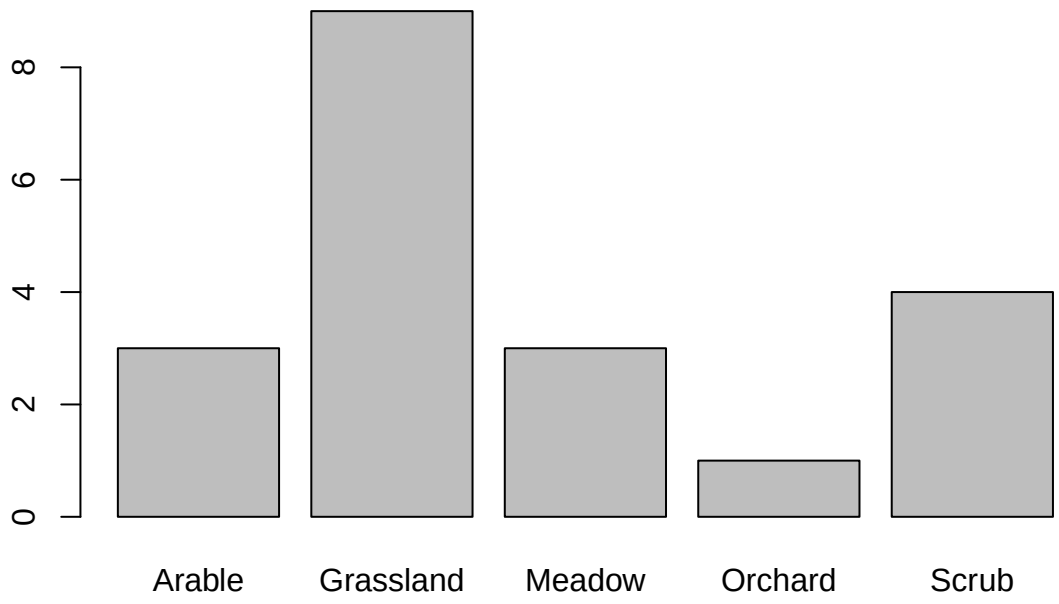
2.2 Categorical variables: Graphical summaries

Now lets look at ways of graphically summarizing a column of categorical data from our data frame. Lets start with the vegetation, using the basic R library

```
table(worms$Vegetation)
```

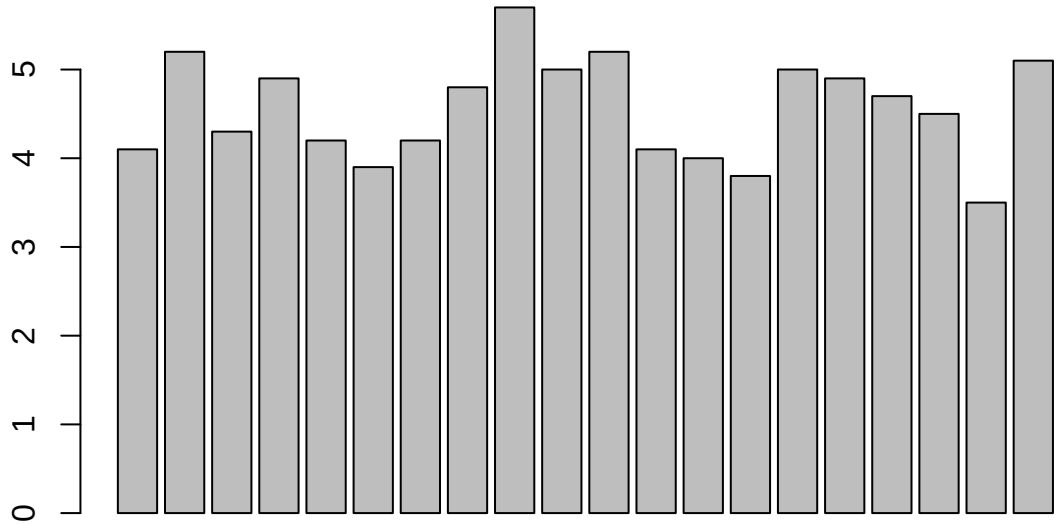
```
##
##    Arable Grassland    Meadow   Orchard     Scrub
##         3         9         3         1         4
```

```
barplot(table(worms$Vegetation))
```

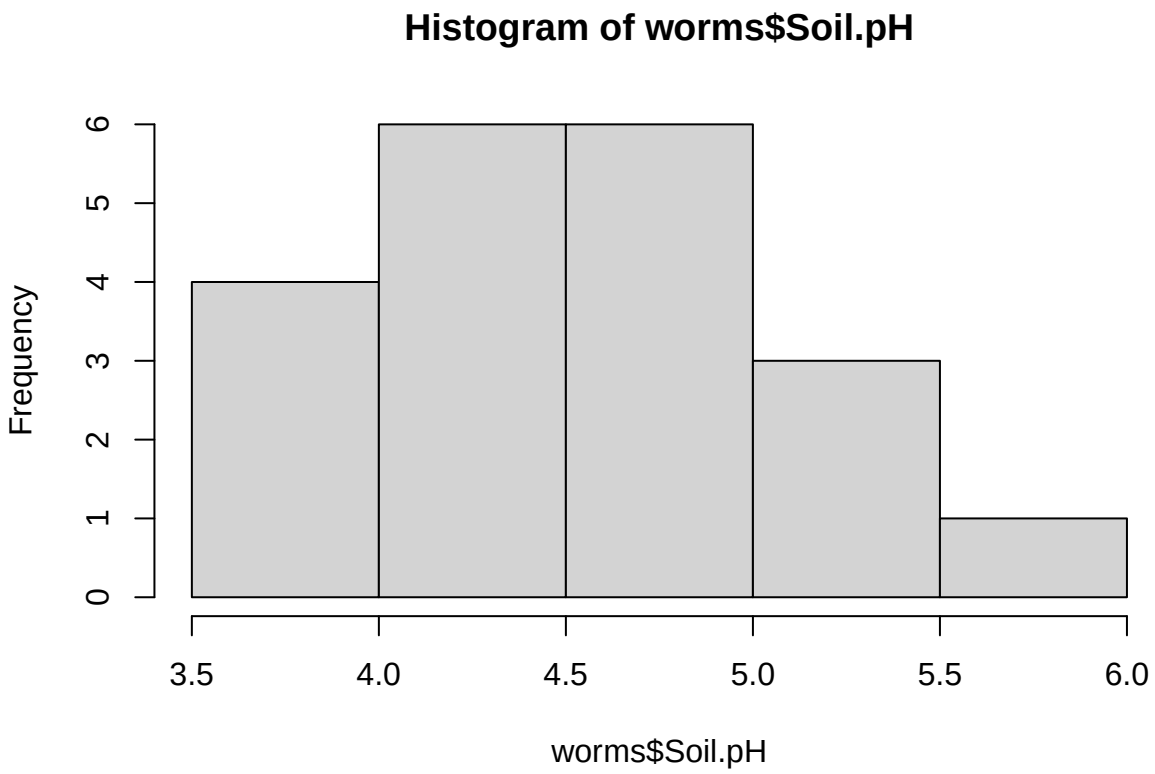


It is also possible to use the same syntax for the two way table (contingency table)

```
barplot(worms$Soil.pH)
```




```
hist(worms$Soil.pH)
```



Using ggplot

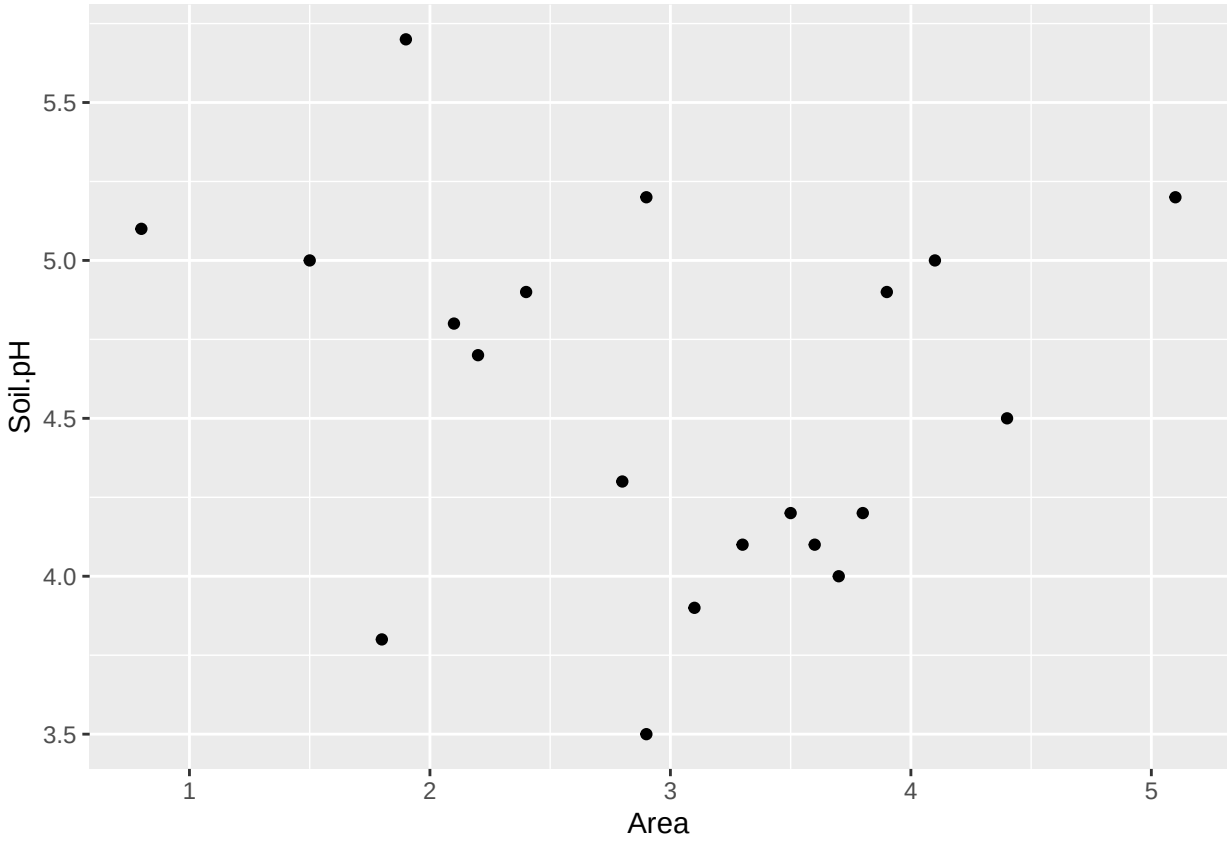
One library that is very handy for graphs is the ggplot2. We have loaded it earlier in this notebook.

There are a huge number of options in ggplot. In most cases you start with `ggplot()`, supply a dataset and aesthetic mapping (with `aes()`). You then add on layers (like `geom_point()` or `geom_histogram()`), scales (like `scale_colour_brewer()`), faceting specifications (like `facet_wrap()`) and coordinate systems (like `coord_flip()`).

A good overview is available in the cheatsheet, you can access from the link

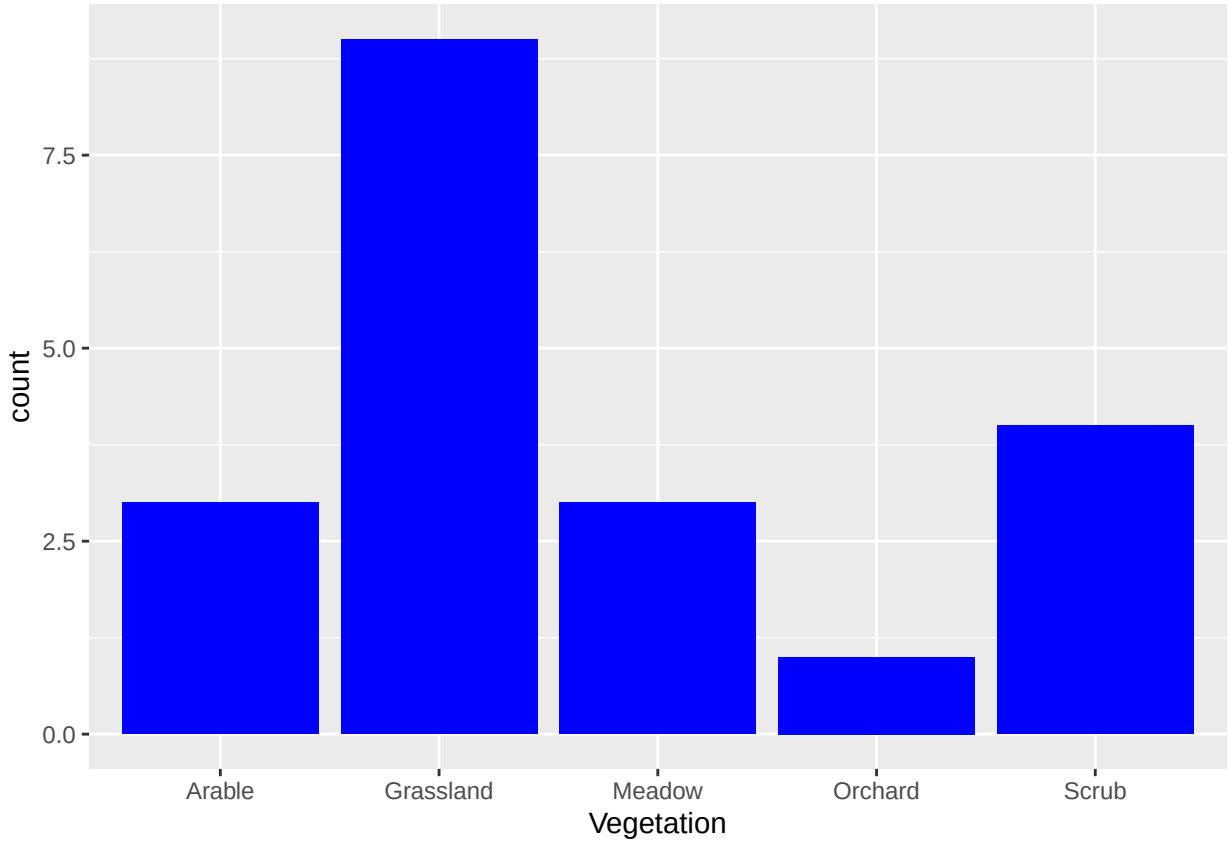
Lets look at replicating the plots above and more using ggplot.

```
#head(worms)
ggplot(data=worms)+geom_point(aes(x=Area,y=Soil.pH))
```



Scatter plot

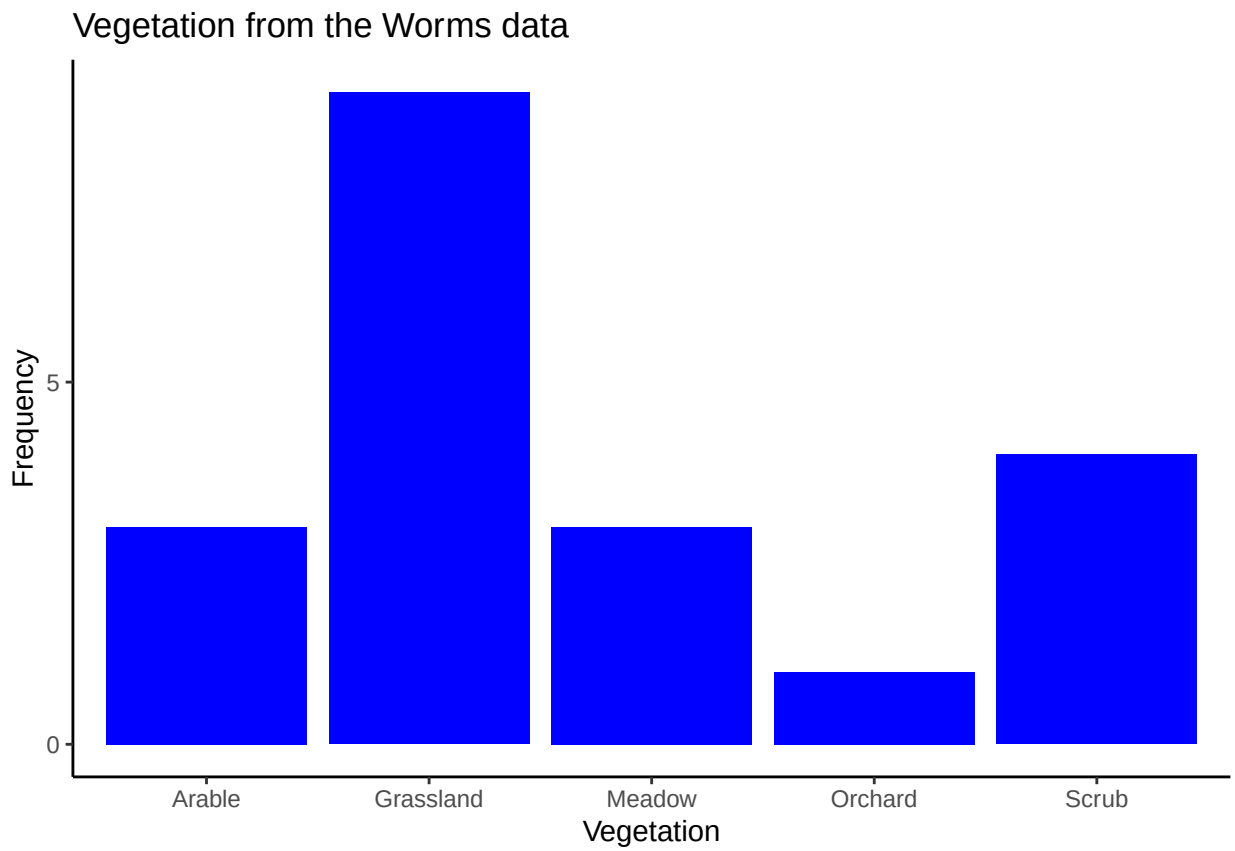
```
ggplot(worms)+ geom_bar(aes(x=Vegetation), fill="blue")
```



Bar plot

There is a lot you can specify in ggplot - see below for another example which now has well labelled titles and axis and a different theme.

```
ggplot(worms)+ geom_bar(aes(x=Vegetation), fill="blue") +
  labs(title = "Vegetation from the Worms data") +
  scale_y_continuous(name="Frequency", breaks=c(0,5,10)) +
  theme_classic()
```



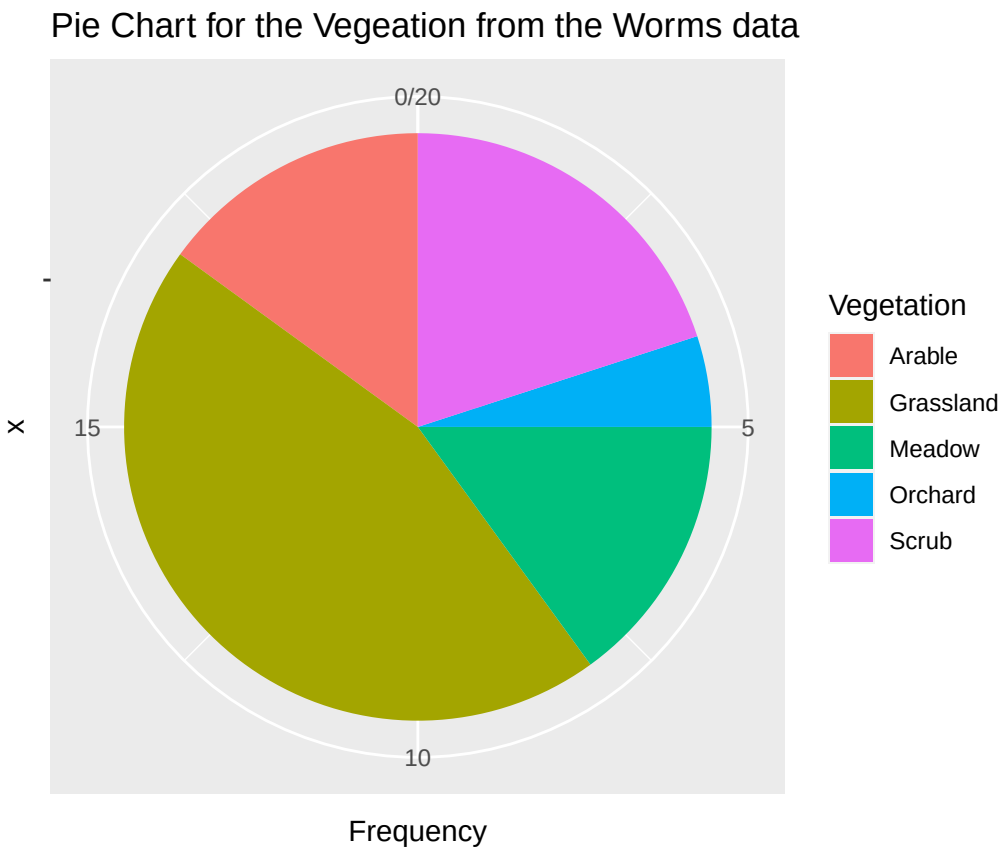
Pie chart

The code chunk below creates a pie chart. In order to do such a plot, the frequency table needs to be computed first, then transformed into a data frame, before finally being plotted using a ggplot option (coord_polar).

Note that giving clear column names makes it easier and quicker to get a better laballed graph.

```
w1<-table(worms$Vegetation)
w2<-as.data.frame(w1)
#w2
colnames(w2)<-c("Vegetation","Frequency")

ggplot(w2, aes(x="", y=Frequency, fill=Vegetation))+geom_bar(width=1, stat = "identity") +coord_polar("
```



2.3. Numerical variables: Descriptve analysis (measures: location & spread)

In the lecture we looked at

- measures of location such as the mean, median and the mode
- measures of spread and variability such as the variance and the IQR
- extreme values

Lets see how to use R to compute the measures of location (mean, median, and mode) and spread (range, inter quartile range, standard deviation and variance):

Measures of location

We are using a small vector of data:

```
data_vector <-c(4,7,2,5,6,2)
```

```
mean(data_vector)
```

Mean

```
## [1] 4.333333
```

```
median(data_vector)
```

Median

```
## [1] 4.5
```

Mode Unlike the mean and the median there is **no built in R function** to compute the mode. This is my approach, there are many ways of achieving the same!

```
freq.table<-table(data_vector)
freq.table<-as.data.frame(freq.table) # what has happened here?
sorted.freq.table<-freq.table[order(-freq.table$Freq),]
head(sorted.freq.table, n=1)
```

DIY Mode

```
##   data_vector Freq
## 1           2    2
```

```
#Uncomment the line below if you do not have the 'modeest' package
#install.packages("modeest")
library(modeest)
mfv(data_vector,method = "mfv")
```

Lazy Mode

```
## [1] 2
```

Weighted Mean How can I calculate my grade?

```
grade.point<-c(13,16,12)
credit<-c(15,15,30)
weighted.mean(grade.point, credit)
```

```
## [1] 13.25
```

Measures of variation

Range Range –> Minimum and the maximum value

```
range(data_vector)
```

```
## [1] 2 7
```

IQR IQR: 25th and 75th percentile values

```
IQR(data_vector)
```

```
## [1] 3.25
```

This is based on the 25th and 75th percentile, which can also be computed in R using the quantile function:

```
quantile(data_vector, c(0.25,0.75))
```

```
##  25%  75%
## 2.50 5.75
```

```
sd(data_vector)
```

Standard Deviation

```
## [1] 2.065591
```

```
var(data_vector)
```

Variance

```
## [1] 4.266667
```

Is this is the easiest/fastest/R'st way to get everything in one go? Glad you asked!!! You are tuning into R with the right approach...

```
summary(data_vector)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.500   4.500   4.333   5.750   7.000
```

This provides us with the **mean**, the **median** and the **1st** and **3rd** quartiles.

Now can you apply what you have learned on the `worms` dataset:

A handy function for summarizing numerical variables is:

```
summary(worms$Worm.density)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    2.00    4.00    4.35    6.25    9.00
```

To find the extreme values - say the highest:

```
head(sort(worms$Worm.density, decreasing = F))
```

```
## [1] 0 1 1 2 2 2
```

How can the code be modified to return the three lowest values?

```
# use this chunk of code to try
```

To compute the others:

- `var()` gives the variance
- `sd()` gives the standard deviation

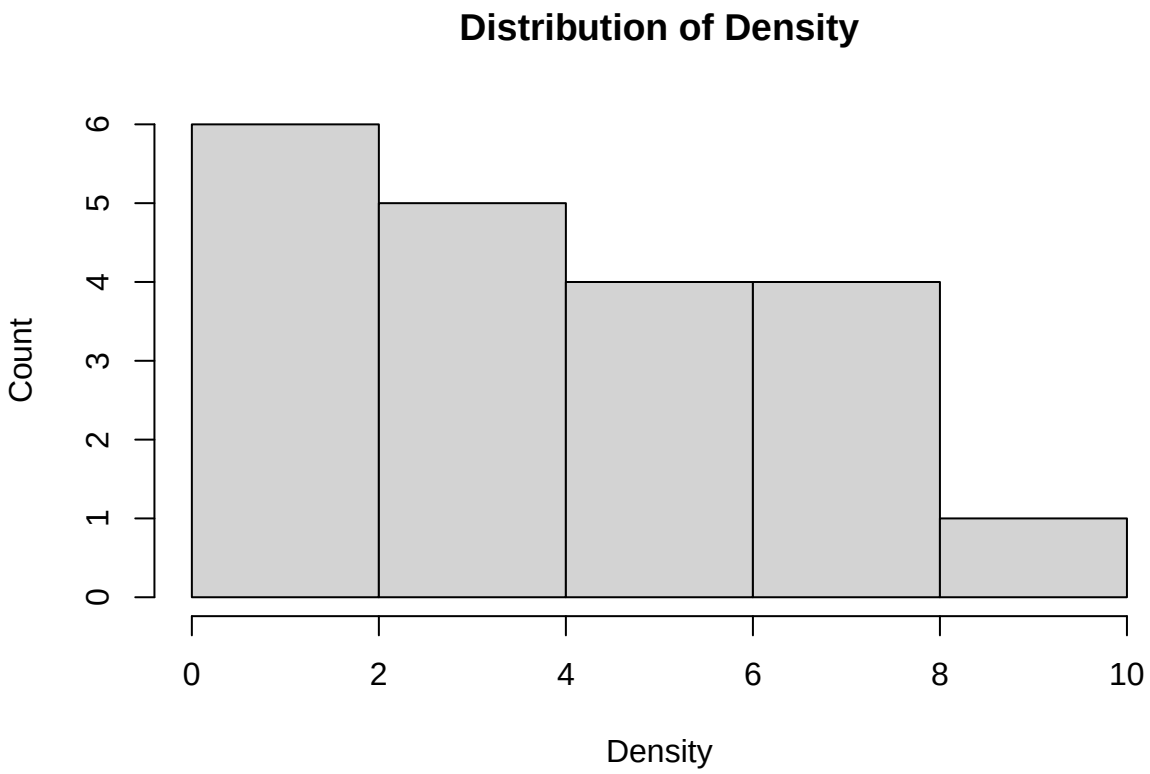
2.4 Numerical variables: Graphical summary

Histogram

A histogram to see the distribution of the values for one of the continuous columns can be created in many ways.

Using the base R code:

```
hist(worms$Worm.density,xlab="Density",ylab="Count",main="Distribution of Density")
```

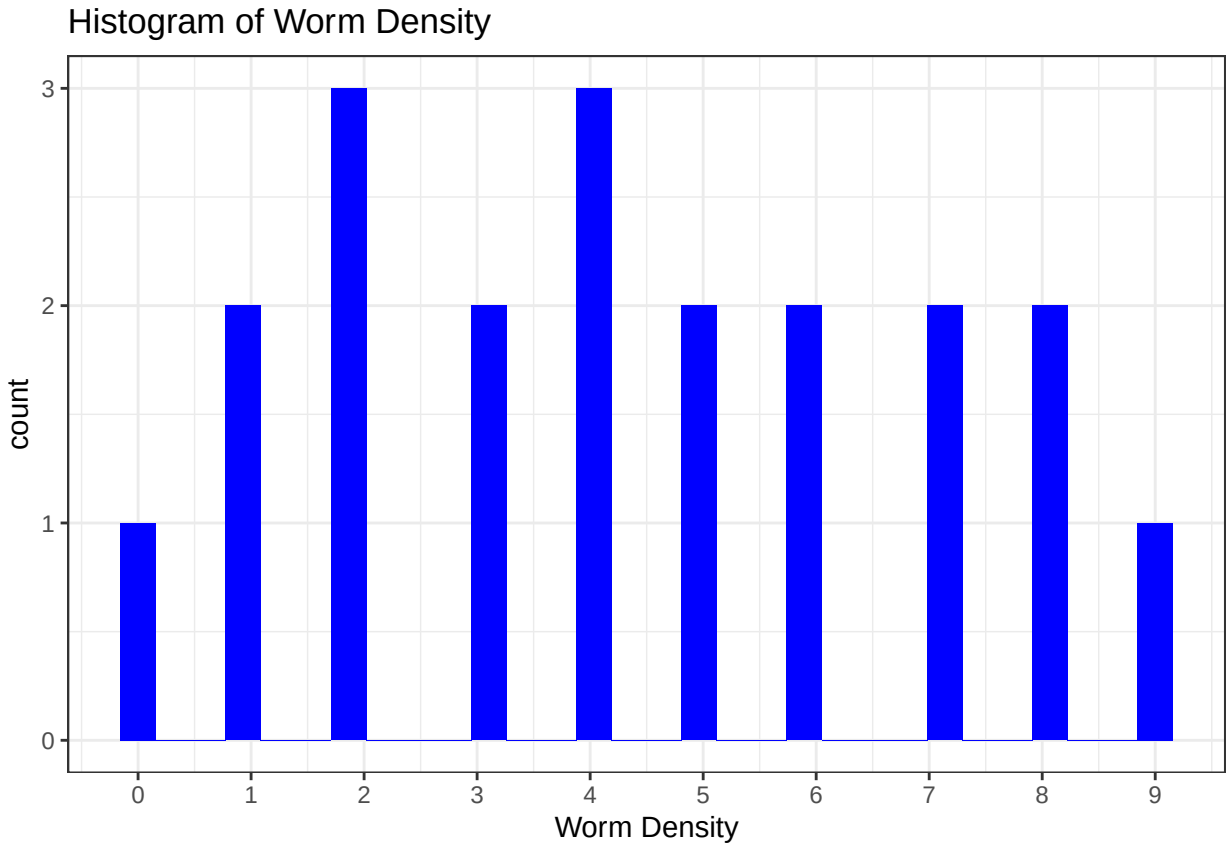


OR ggplot can also be used:

```
# histogram worms

ggplot(worms, aes(x=Worm.density)) +
  geom_histogram(fill="blue") +
  labs(title="Histogram of Worm Density") +
  scale_x_continuous(name="Worm Density", breaks = c(0,1,2,3,4,5,6,7,8,9)) +
  theme_bw()
```

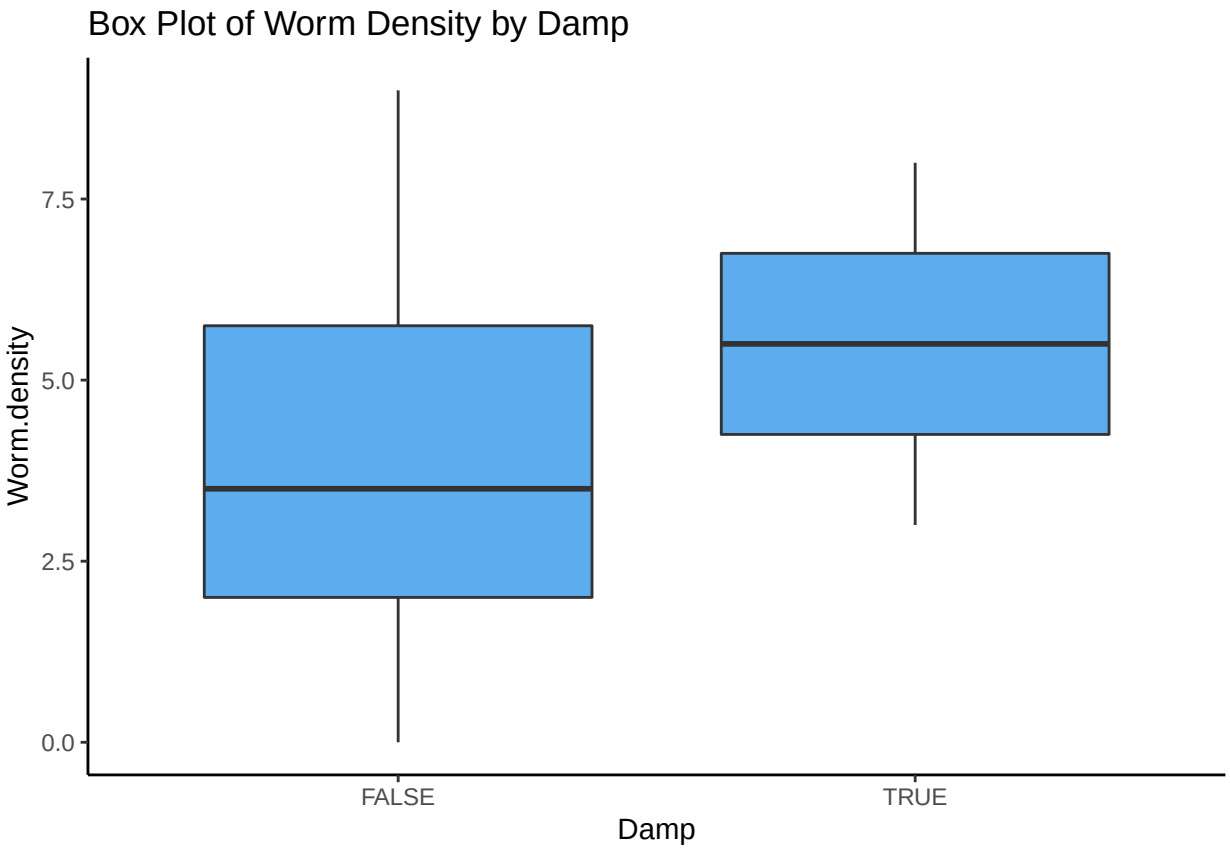
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Box plot

Another interesting perspective on a continuous variable, when it is part of a data set with categorical variable is to look at them both at the same time. A **boxplot** is a useful tool to achieve this.

```
#box plot
ggplot(worms, aes(x=Worm.density, y=Damp)) +
  geom_boxplot(fill="steelblue2") +
  theme_classic() +
  labs(title="Box Plot of Worm Density by Damp") +
  coord_flip()
```


Box Plot of Worm Density by Damp
Worm.density
7.5
5.0
2.5
0.0
FALSE
TRUE
Damp