Iris: Data Manipulation + Visualization

Edgar Anderson’s Iris Data Description This famous (Fisher’s or Anderson’s) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

str(iris)

## 'data.frame': 150 obs. of 5 variables:  
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

#libraries:  
library(MASS)  
library(dplyr)

## Warning: package 'dplyr' was built under R version 4.0.3

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':  
##   
## select

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

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

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 4.0.3

library(magrittr) #%>% operator

## Warning: package 'magrittr' was built under R version 4.0.3

library(tidyr)

## Warning: package 'tidyr' was built under R version 4.0.3

##   
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':  
##   
## extract

iris is a data frame with 150 cases (rows) and 5 variables (columns) named Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species. We rearrange columns for better aesthetics of the data, and it would be helpful when we tidy data.

str(iris)

## 'data.frame': 150 obs. of 5 variables:  
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...  
## $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

#rearranging columns  
iris1<-iris[c(5,1:4)]  
str(iris1)

## 'data.frame': 150 obs. of 5 variables:  
## $ Species : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...  
## $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...  
## $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...  
## $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...

The principles of tidy data provide a standard way to organise data values within a dataset. A standard makes initial data cleaning easier because you don’t need to start from scratch and reinvent the wheel every time. The tidy data standard has been designed to facilitate initial exploration and analysis of the data, and to simplify the development of data analysis tools that work well together. In R, we can use the library: tidyr for data cleaning,

library(tidyr)  
#tidying data using tidyr  
iris.tidy <- iris1 %>%  
 gather(key, Value, -Species) %>%  
 separate(key, c("Part", "Measure"), "\\.")  
str(iris.tidy)

## 'data.frame': 600 obs. of 4 variables:  
## $ Species: Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ Part : chr "Sepal" "Sepal" "Sepal" "Sepal" ...  
## $ Measure: chr "Length" "Length" "Length" "Length" ...  
## $ Value : num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...

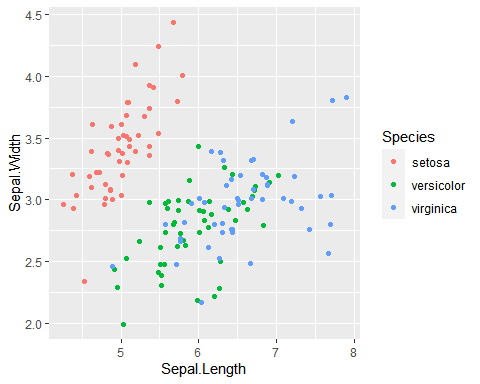
This is just a small instance of data tidying that I’ve used to skip the manual steps it takes to tidy data using data manipulation functions provided in the library: dplyr. For beginners, I have shown how data manipulation can be done manually in the following code chunk.

library(dplyr)  
library(magrittr)  
  
#import data  
iris<-read.csv("datasets/iris.csv")  
#Separating sepal data  
iris1a<-iris %>%  
 mutate(Part="Sepal") %>%  
 select(Species,Part,Sepal.Length,Sepal.Width) %>%  
 rename(Length=Sepal.Length,Width=Sepal.Width)  
#Separating petal data  
iris1b<-iris %>%  
 mutate(Part="Petal") %>%  
 select(Species,Part,Petal.Length,Petal.Width) %>%  
 rename(Length=Petal.Length,Width=Petal.Width)  
#Combining sepal and petal data  
iris2<-rbind(iris1a,iris1b)  
iris2a<-iris2 %>%  
 mutate(Measure="Length") %>%  
 select(Species,Part,Measure,Length) %>%  
 rename(Value=Length)  
iris2b<-iris2 %>%  
 mutate(Measure="Width") %>%  
 select(Species,Part,Measure,Width) %>%  
 rename(Value=Width)  
iris3<-rbind(iris2a,iris2b)

Now, we are at the step where we visualize the data we cleaned.

First, we are going to visualize the Sepal data of all three species.

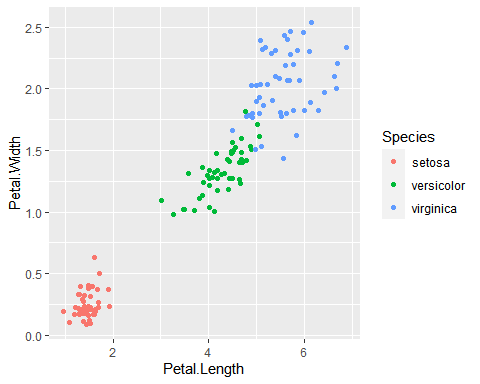
ggplot(data = iris,aes(x=Sepal.Length,y=Sepal.Width,col=Species))+  
 geom\_jitter()



The data interpretation can be difficult in the following graph as we can’t really interpret anything useful.

Now we are going to plot Petal Length and Petal Width.

ggplot(data = iris,aes(x=Petal.Length,y=Petal.Width,col=Species))+  
 geom\_jitter()



The second graph is more visually pleasing as there isn’t a mix up of different species and we can even conclude a strong correlation for measure of Petals depending on the species.

The plots in the graphs aren’t accurate because we used the function geom\_jitter Funtion geom\_point() maps the data as it is, but geom\_jitter() adds a little variance to data for better representation in several cases and avoid overlapping of same values.

In the next graph we are going to use a function facet\_grid() which would make the visualization experience more aesthetic and easier to interpret.

ggplot(data=iris.tidy,aes(x=Measure,y=Value,col=Species))+  
 geom\_jitter()+  
 facet\_grid(.~Part)

