I have found the following commands quite useful during the EDA part of any Data Science project. We will work with the `tidyverse` package where we will actually need the `dplyr` and the `ggplot2` only and with the `iris`dataset.

select_if | rename_if

The `select_if` function belongs to `dply` and is very useful where we want to choose some columns based on some conditions. We can also add a function that applies to column names.

Example: Let's say that I want to choose only the numeric variables and to add the prefix "numeric_" to their column names.

```
library(tidyverse)

iris%>%select_if(is.numeric,  list(~ paste0("numeric_", .)))%>%head()
```

Output:

	numeric_Sepal.Length	numeric_Sepal.Width	numeric_Petal.Length	numeric_Petal.Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2
6	5.4	3.9	1.7	0.4

Notice that we can also use the `rename_if` in the same way. An important note is that the rename_if(), rename_at(), and rename_all() have been superseded by rename_with(). The matching select statements have been superseded by the combination of a select() + rename_with().

These functions were superseded because mutate_if() and friends were superseded by across(). select_if() and rename_if() already use tidy selection so they can't be replaced by across() and instead we need a new function.

---

everything

In many Data Science projects, we want one particular column (usually the dependent variable y) to appear first or last in the dataset. We can achieve this using the `everything()` from `dplyr` package.

Example: Let's say that I want the column Species to appear **first** in my dataset.

```
mydataset<-iris%>%select(Species, everything())
mydataset%>%head()
```

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

Example: Let's say that I want the column Species to appear **last** in my dataset.

This is a little bit tricky. Have a look below at how we can do it. We will work with the `mydataset` where the Species column appears first and we will remove it to the last column.

```
mydataset%>%select(-Species, everything())%>%head()
```

	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

---

relocate

The `relocate()` is a new addition in dplyr 1.0.0. You can specify exactly where to put the columns with **.before** or **.after**

Example: Let's say that I want the Petal.Width column to appear next to Sepal.Width

```
iris%>%relocate(Petal.Width, .after=Sepal.Width)%>%head()
```

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

Notice that we can also set to appear after the last column.

Example: Let's say that I want the Petal.Width to be the last column

```
iris%>%relocate(Petal.Width, .after=last_col())%>%head()
```

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

You can find more info in the tidyverse documentation

---

pull

When we work with data frames and we select a single column, sometimes we the output to be as.vector. We can achieve this with the `pull()` which is part of `dplyr`.

Example: Let's say that I want to run a t.test in the Sepal.Length for setosa versus virginica. Note the the t.test function expects numeric vectors.

```
setosa_sepal_length<-iris%>%filter(Species=='setosa')%>%
```

```
select(Sepal.Length)%>%pull()
virginica_sepal_length<-iris%>%filter(Species=='virginica')%
>%select(Sepal.Length)%>%pull()

t.test(setosa_sepal_length,virginica_sepal_length)
```

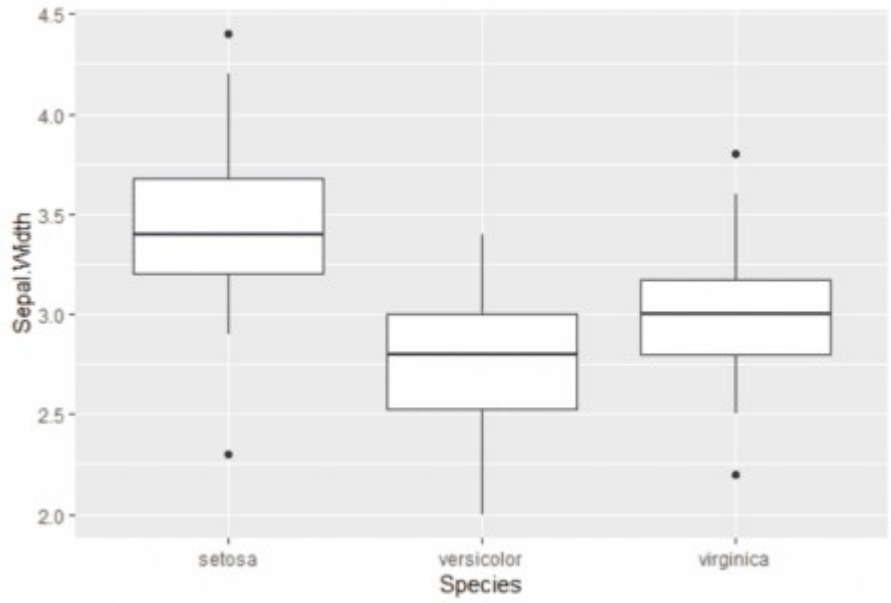
```
        Welch Two Sample t-test

data:  setosa_sepal_length and virginica_sepal_length
t = -15.386, df = 76.516, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.78676 -1.37724
sample estimates:
mean of x mean of y
    5.006     6.588
```

---

reorder

When you work with ggplot2 sometimes is frustrating when you have to reorder the factors based on some conditions. Let's say that we want to show the boxplot of the Sepal.Width by Species.

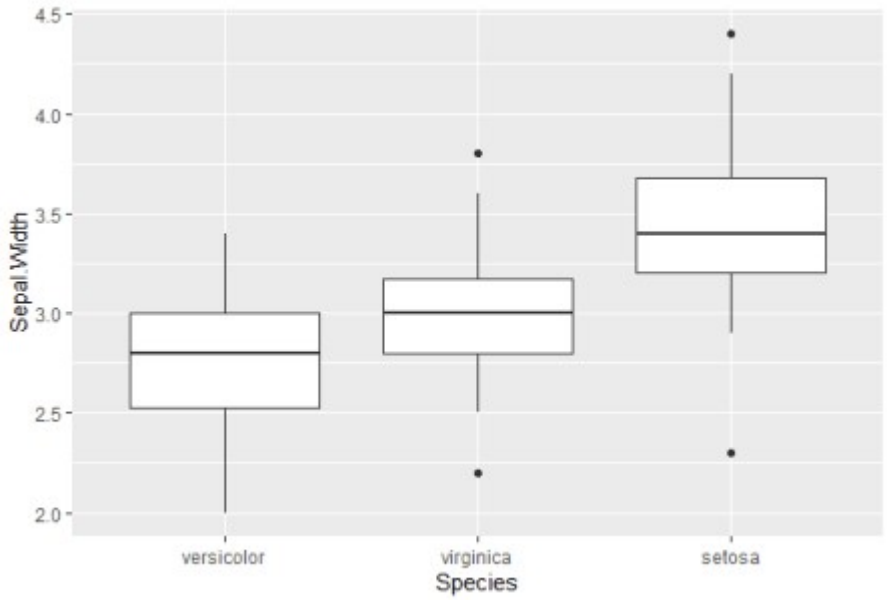
```
iris%>%ggplot(aes(x=Species, y=Sepal.Width))+geom_boxplot()
```



Example: Let's assume that we want to reorder the boxplot based on the Species' median.

We can do that easily with the `reorder()` from the `stats` package.

```
iris%>%ggplot(aes(x=reorder(Species,Sepal.Width, FUN = median),
y=Sepal.Width))+geom_boxplot()+xlab("Species")
```


4.5
4.0
3.5
3.0
2.5
2.0
Sepal.Width
versicolor
virginica
setosa
Species