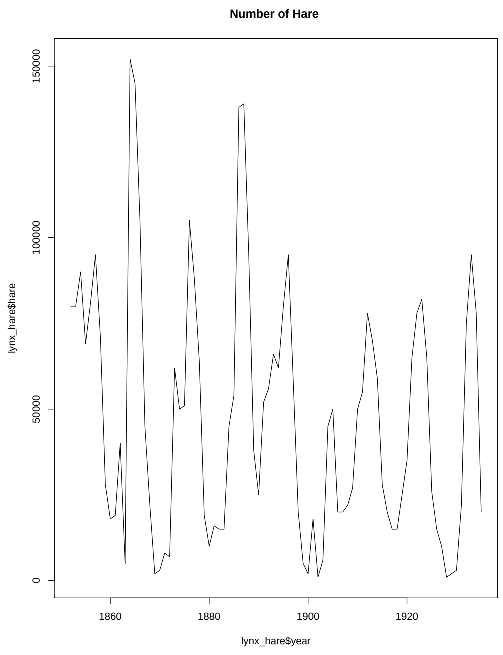
Graphics

- R has built-in functions to automatically plot many standard statistical graphics. Histograms and box plots may be generated with `hist()` and `boxplot()`, respectively.
- Here's an example. To start with let's make a simple plot of the number of hare that you used before. `plot()` function takes to vectors and uses them for the x and y values:

```
> plot(x=lynx_hare$year, y=lynx_hare$hare, main = "Number of Hare ",
     type="l")
# main = "..." sets the title of the plot.
#type="l" allows for plotting with lines. If you want to plot with points, use type
     ="p"
```

  Hopefully, you should be able to see this output:



- Once you've created a graphic that you're happy with, you can copy the entire thing. In the graphic window, click on export. You have three choices: *save as image*, *save as PDF*, and *copy to clipboard*. Image and PDF allow you to save the graph in a portable file. Copy to clipboard allows you to copy and paste the graph, *e.g.* into Ms Word. Simply click on *copy to clipboard* and on *copy plot*. Then paste it into Word and adjust its size.

  Try it: save the plot you created above to a PDF file!
- Often, your data set contains various groups, and you would like to make a box plot for each group. Using **`boxplot`**, that is easy. Here's an example.

Let us load a new data set from Black Board on the CO_2 levels measured in different years and per month. This data set is in the file CO2_scripps.csv. Read the file and inspect what kind of data you have:

```
>co2<-read.table("CO2_scripps.csv", header=TRUE, sep=";", check.names
     = FALSE)
head(co2)
```

Year Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1 1959 315.62 316.38 316.71 317.72 318.29 318.15 316.54 314.80 313.84 313.26 314.80 315.58

2 1960 316.43 316.97 317.58 319.02 320.03 319.59 318.18 315.91 314.16 313.83 315.00 316.19

3 1961 316.93 317.70 318.54 319.48 320.58 319.77 318.57 316.79 314.81 315.38 316.10 317.01

4 1962 317.94 318.55 319.68 320.63 321.01 320.55 319.57 317.40 316.25 315.42 316.69 317.70

5 1963 318.74 319.07 319.86 321.39 322.25 321.48 319.74 317.77 316.21 315.99 317.12 318.31

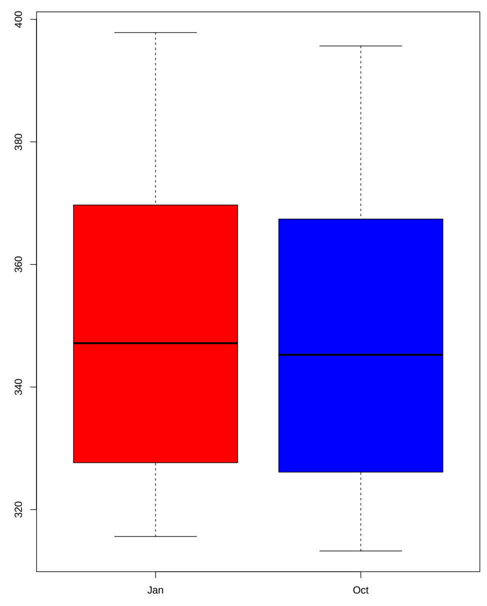
6 1964 319.57 320.00 320.75 321.83 322.25 321.89 320.44 318.70 316.70 316.79 317.79 318.71

Annual Average 1 315.97 2 316.91 3 317.64 4 318.45 5 318.99 6 319.62

Let us make a boxplot to compare the CO_2 levels in four seasons:

```
> boxplot(co2$Jan, co2$Oct, names=c("Jan", "Oct"), col = c("red", "
    blue"))
#names assigns the values to use on the x axis, and the colors give the colors to
    use.
```

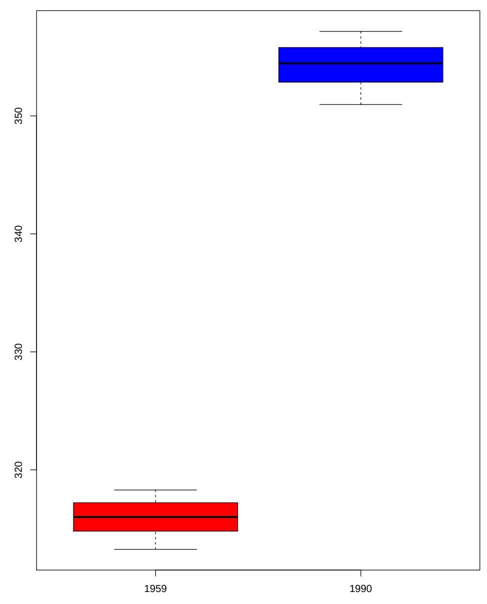
Suprisingly, there is not much of a difference between the two:



Maybe the amount of CO_2 is increasing in time. Let's plot three different years:

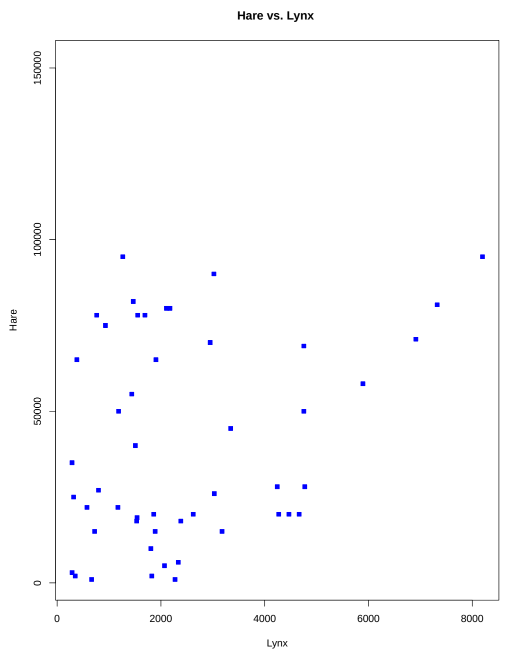
```
> boxplot(unlist(co2[co2$Year=="1959",2:13]), unlist(co2[co2$Year=="
    1990",2:13]), names=c("1959", "1990"), col = c("red", "blue"))
#here unlist makes a vector from one row of co2 dataframe
```

Indeed, there is a large increase in the amount of CO_2 in the years between 1959 and 1990:



- Scatter plots are plots of to numeric variables. For example, using the `lynx_hare` data set, we can make a scatter plot of the number of hare versus number of lynx:

```
> plot(x=lynx_hare$lynx, y=lynx_hare$hare, main = "Hare vs. Lynx ",
    col="blue",
    xlab = "Lynx", ylab = "Hare", bg="blue", pch=22)
#I use pch=22 and bg="blue" to get filled symbols in color blue.
```



R Packages

One of the strengths of R is that the system can easily be extended. The system allows you to write new functions and share those functions with other users through a so-called "R package". Other users can then download the package and install the collection of functions as a "library". The R package may also contain other R objects, for example data sets or documentation. There is a lively R user community and many R packages have been written and made available on CRAN for other users. Just a few examples: there are packages for drawing maps, drawing graphics, exporting objects to HTML, and the list goes on and on. When you download R, a number of packages (around 30) are downloaded by default.

- To use a function from an R package, that package has to be attached to your system. When you start R, not all of the downloaded packages are attached; only seven are by default. To attach (load) another package, use the `library()` function. For instance, to load the package `"ggplot2"`, run:

```
> library(ggplot2) # Attach or "load" a library.
```

- Use the function `search()` to see a list of packages that are currently attached to the system:

```
> search()
```

```
[1] ".GlobalEnv" "package:stats" "package:graphics"
[4] "package:grDevices" "package:datasets" "package:utils"
[7] "package:methods" "Autoloads" "package:base"
```

This list is also called the *search path*. The first element of the output is ".GlobalEnv", which is the current workspace of the user.

- You will occasionally need a package that is not yet installed on your computer. If you have a connection to the internet then a package that is available on CRAN can be installed very easily using the function **install.packages()**. For instance, to install the "binom" package, run the following command:

```
# Install the binom package; the quotes '...' or "..." are required.
> install.packages("binom", dependencies = TRUE)
```

Because we specified **dependencies** = TRUE, if the package **binom** uses ("depends on") yet other packages, these are installed as well. After the installation, you still need to attach the package to your current R session to start using it:

```
> library(binom) # Attach or "load" the binom library.
```

- Another example: The library "MASS" provides an object called shoes. To access this object, run the following commands:

```
> install.packages("MASS", dependencies = TRUE) # Install the package, if
      you don't have it yet
> library(MASS) # Attach/load the library.
> data(shoes) # this makes available a data frame called "shoes"
# on some systems, you have to add quotes, like this:
> data("shoes")
# Print the object "shoes":
> shoes
```

```
$A
[1] 13.2 8.2 10.9 14.3 10.7 6.6 9.5 10.8 8.8 13.3
$B
[1] 14.0 8.8 11.2 14.2 11.8 6.4 9.8 11.3 9.3 13.6
```

We stress that, as with all software, you need to download and install packages only once on a given computer (using **install.packages()**). But you need to load it (using **library()**) in every R session in which you wish to use it.

- The function **library()** can also be used to list all the available libraries on your system with a short description. Run the function without any arguments.