Swiss Paraplegic Research

Workshop – Introduction into R

Data Visualization

Yannick Rothacher
Data Scientist

Data visualization in R

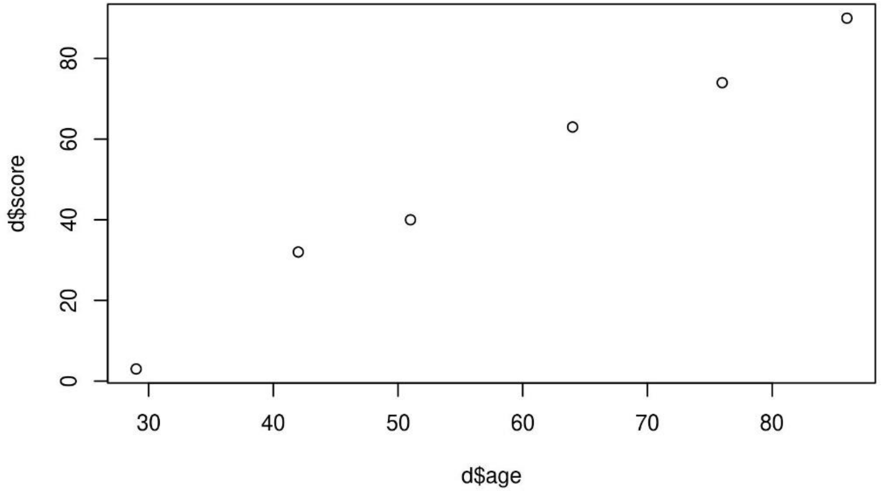
- The **visualization of data** is a fundamental part of data analysis
- Data visualizations are often the best way to **convey complex information**
- There are many functions in R to create different plots
  - We will first focus on the basic **`plot`** function…
  - …illustrated on this simple data set:

```
> d
     id    SCIstatus    group  score  age
1 pat1  paraplegic     cntrl      3   29
2 pat2  tetraplegic  treatm     32   42
3 pat3  tetraplegic   cntrl     40   51
4 pat4  paraplegic   treatm     63   64
5 pat5  paraplegic     cntrl     74   76
6 pat6  tetraplegic  treatm     90   86
```

The `plot` function

- Using **`plot`** to create a scatter plot:

```
> d
    id   SCIstatus    group  score  age
1 pat1  paraplegic    cntrl      3   29
2 pat2 tetraplegic   treatm     32   42
3 pat3 tetraplegic    cntrl     40   51
4 pat4  paraplegic   treatm     63   64
5 pat5  paraplegic    cntrl     74   76
6 pat6 tetraplegic   treatm     90   86

> plot(x = d$age, y = d$score)
```



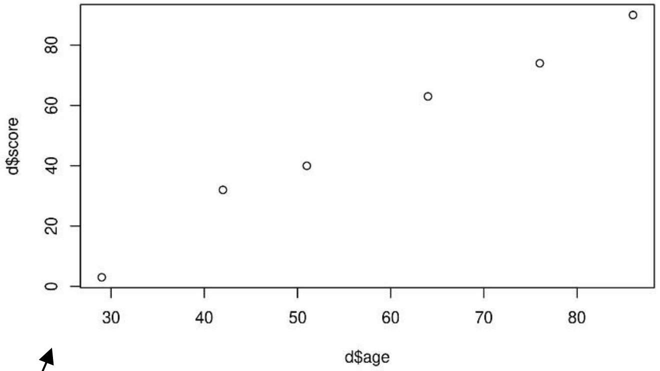
The `plot` function

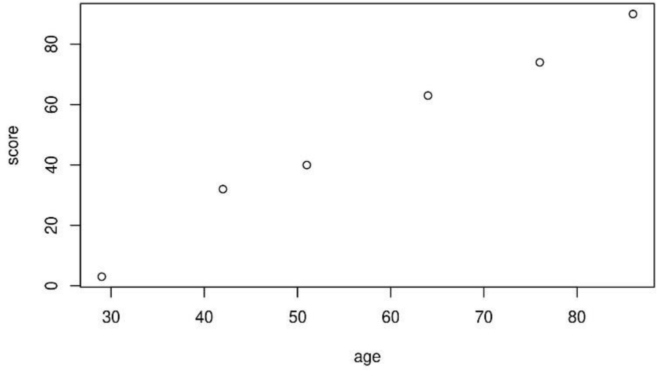
- Using **`plot`** to create a scatter plot:

```
> d
     id   SCIstatus    group  score  age
1 pat1  paraplegic    cntrl      3   29
2 pat2  tetraplegic  treatm     32   42
3 pat3  tetraplegic   cntrl     40   51
4 pat4  paraplegic   treatm     63   64
5 pat5  paraplegic    cntrl     74   76
6 pat6  tetraplegic  treatm     90   86

> plot(x = d$age, y = d$score)
```

Using a formula:

```
> plot(score ~ age, data = d)
```





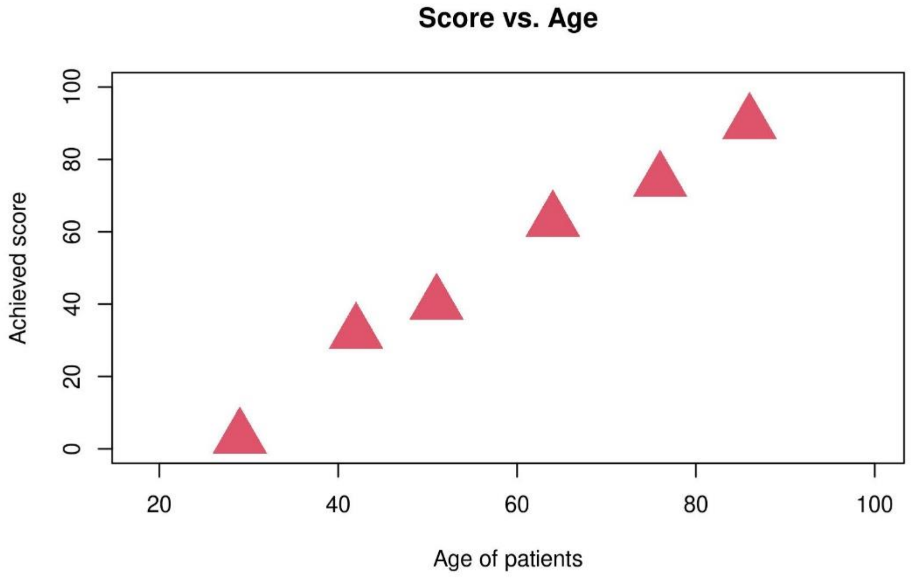
The `plot` function

-> How the plot function processes the data:

```
 x   y pch col cex
29   3  17   2   4
42  32  17   2   4
51  40  17   2   4
64  63  17   2   4
76  74  17   2   4
86  90  17   2   4
```

- `plot` can take many additional arguments:

```
> d
    id   SCIstatus    group  score  age
1 pat1  paraplegic    cntrl      3   29
2 pat2  tetraplegic  treatm     32   42
3 pat3  tetraplegic   cntrl     40   51
4 pat4  paraplegic   treatm     63   64
5 pat5  paraplegic    cntrl     74   76
6 pat6  tetraplegic  treatm     90   86

> plot(x = d$age, y = d$score,
       main = "Score vs. Age",
       xlab = "Age of patients",
       ylab = "Achieved score",
       xlim = c(18, 100),
       ylim = c(0, 100),
       pch = 17,
       col = 2,
       cex = 4)
```

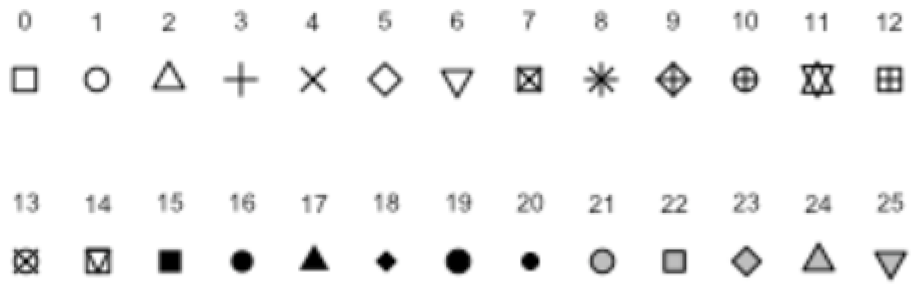


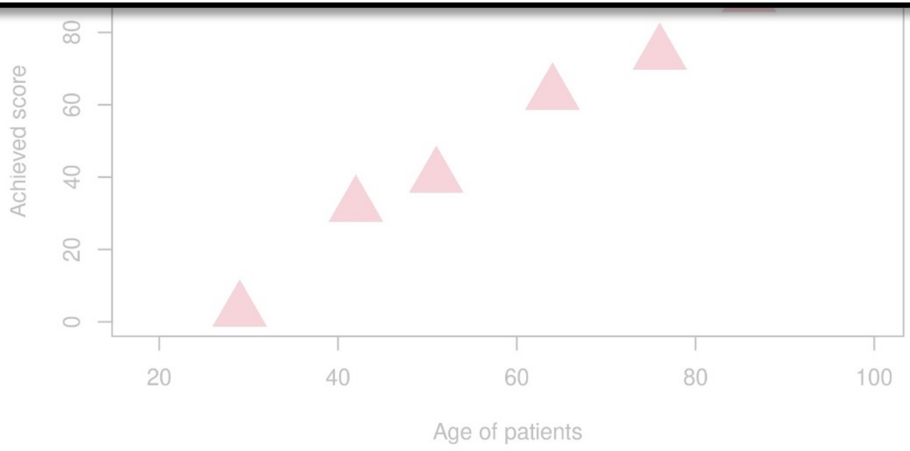
The `plot` function

- `plot` can take many additional ar

```
> d
     id    SCIstatus    group  score
1 pat1  paraplegic    cntrl      3
2 pat2  tetraplegic  treatm     32
3 pat3  tetraplegic   cntrl     40
4 pat4  paraplegic   treatm     63
5 pat5  paraplegic    cntrl     74
6 pat6  tetraplegic  treatm     90

> plot(x = d$age, y = d$score,
       main = "Score vs. Age",
       xlab = "Age of patients",
       ylab = "Achieved score",
       xlim = c(18, 100),
       ylim = c(0, 100),
       pch = 17,
       col = 2,
       cex = 4)
```

The **pch** argument defines the **shape** of the points:





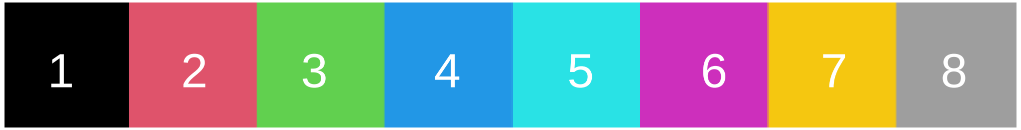
The `plot` function

- `plot` can take many additional ar

```
> d
   id   SCIstatus   group  score
1 pat1  paraplegic   cntrl      3
2 pat2  tetraplegic treatm     32
3 pat3  tetraplegic  cntrl     40
4 pat4  paraplegic  treatm     63
5 pat5  paraplegic   cntrl     74
6 pat6  tetraplegic treatm     90

> plot(x = d$age, y = d$score,
       main = "Score vs. Age",
       xlab = "Age of patients",
       ylab = "Achieved score",
       xlim = c(18, 100),
       ylim = c(0, 100),
       pch = 17,
       col = 2,
       cex = 4)
```

The **col** argument defines the **color** of the points

- Can define color through number:



- …through name:
  `col = "green"`

- …through Hex code:
  `col = "#00FF00"`

- …through rgb values:
  `col = rgb(0,1,0)`

Age of patients

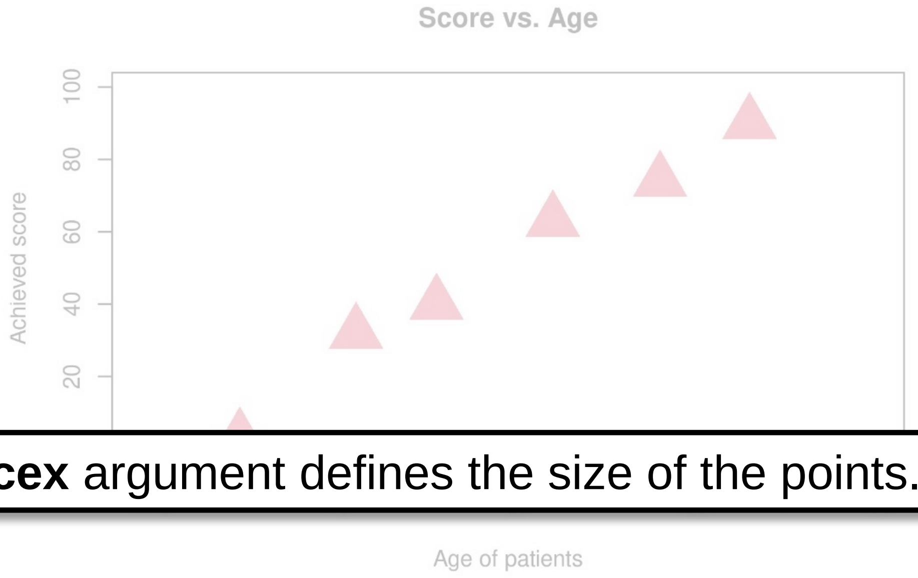
The `plot` function

-> How the plot function processes the data:

```
 x  y pch col cex
29  3  17   2   4
42 32  17   2   4
51 40  17   2   4
64 63  17   2   4
76 74  17   2   4
86 90  17   2   4
```

- `plot` can take many additional arguments:

```
> d
    id    SCIstatus    group  score   age
1 pat1  paraplegic     cntrl      3    29
2 pat2  tetraplegic   treatm     32    42
3 pat3  tetraplegic    cntrl     40    51
4 pat4  paraplegic    treatm     63    64
5 pat5  paraplegic     cntrl     74    76
6 pat6  tetraplegic   treatm     90    86

> plot(x = d$age, y = d$score,
       main = "Score vs. Age",
       xlab = "Age of patients",
       ylab = "Achieved score",
       xlim = c(18, 100),
       ylim = c(0, 100),
       pch = 17,
       col = 2,
       cex = 4)
```



The **cex** argument defines the size of the points.

The `plot` function

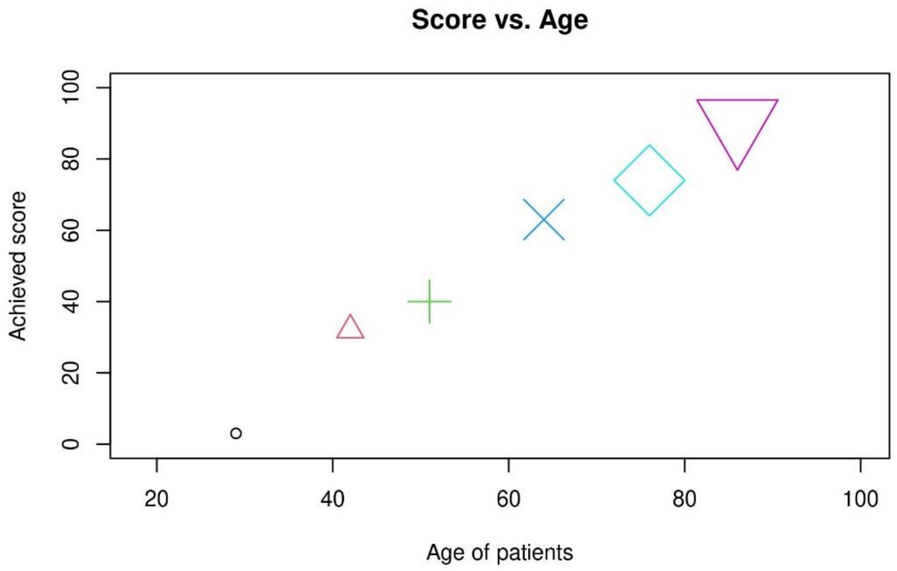
- `plot` can take many additional arguments:

```
> d
    id   SCIstatus    group  score  age
1 pat1  paraplegic    cntrl      3   29
2 pat2  tetraplegic  treatm     32   42
3 pat3  tetraplegic   cntrl     40   51
4 pat4  paraplegic   treatm     63   64
5 pat5  paraplegic    cntrl     74   76
6 pat6  tetraplegic  treatm     90   86

> plot(x = d$age, y = d$score,
     main = "Score vs. Age",
     xlab = "Age of patients",
     ylab = "Achieved score",
     xlim = c(18, 100),
     ylim = c(0, 100),
     pch = 1:nrow(d),
     col = 1:nrow(d),
     cex = 1:nrow(d))
```

Supplied as vectors

-> How the plot function processes the data:

```
 x   y pch col cex
29   3   1   1   1
42  32   2   2   2
51  40   3   3   3
64  63   4   4   4
76  74   5   5   5
86  90   6   6   6
```



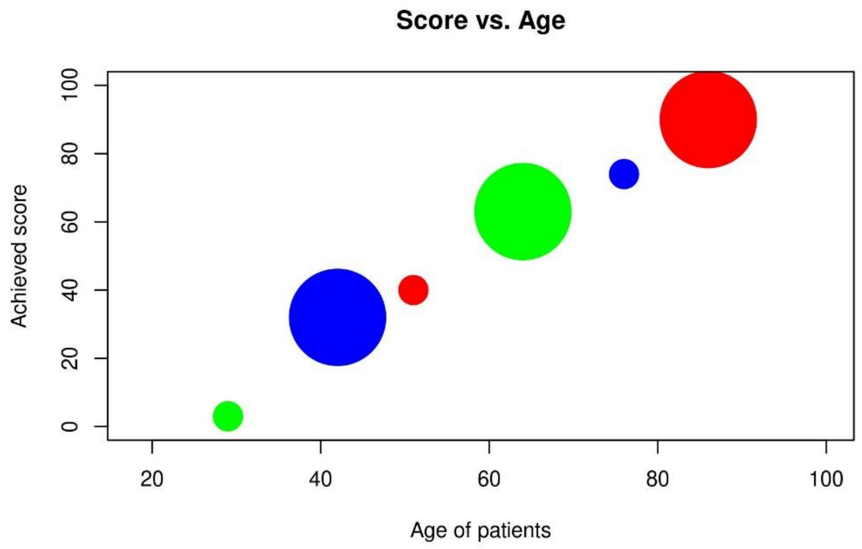
The `plot` function

- `plot` can take many additional arguments:

```
> d
    id   SCIstatus    group  score  age
1 pat1  paraplegic    cntrl      3   29
2 pat2  tetraplegic  treatm     32   42
3 pat3  tetraplegic   cntrl     40   51
4 pat4  paraplegic   treatm     63   64
5 pat5  paraplegic    cntrl     74   76
6 pat6  tetraplegic  treatm     90   86

> plot(x = d$age, y = d$score,
       main = "Score vs. Age",
       xlab = "Age of patients",
       ylab = "Achieved score",
       xlim = c(18, 100),
       ylim = c(0, 100),
       pch = 19,
       col = c("green", "blue", "red"),
       cex = c(3, 10))
```

-> How the plot function processes the data:

```
 x   y pch   col cex
29   3  19 green   3
42  32  19  blue  10
51  40  19   red   3
64  63  19 green  10
76  74  19  blue   3
86  90  19   red  10
```



The `plot` function

- `plot` can take many additional arguments:

```
> d
     id   SCIstatus    group  score   age
1 pat1  paraplegic    cntrl       3    29
2 pat2  tetraplegic  treatm      32    42
3 pat3  tetraplegic   cntrl      40    51
4 pat4  paraplegic   treatm      63    64
5 pat5  paraplegic    cntrl      74    76
6 pat6  tetraplegic  treatm      90    86
```

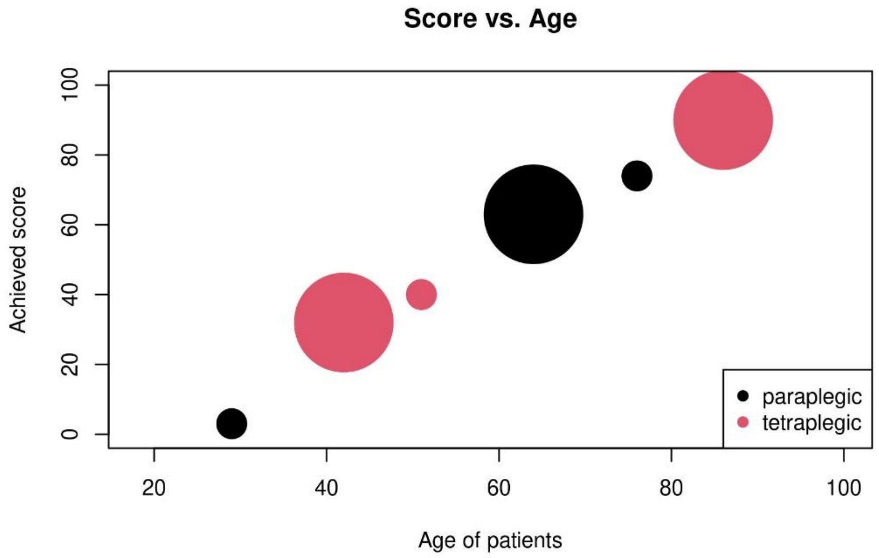
```
> plot(x = d$age, y = d$score,
       main = "Score vs. Age",
       xlab = "Age of patients",
       ylab = "Achieved score",
       xlim = c(18, 100),
       ylim = c(0, 100),
       pch = 19,
       col = d$SCIstatus,
       cex = c(3, 10))
```

Supplied as a factor

```
> legend("bottomright", legend=levels(d$SCIstatus),
         col=1:nlevels(d$SCIstatus), pch=19)
```

-> How the plot function processes the data:

```
 x   y pch col cex
29   3  19   1   3
42  32  19   2  10
51  40  19   2   3
64  63  19   1  10
76  74  19   1   3
86  90  19   2  10
```



The `plot` function

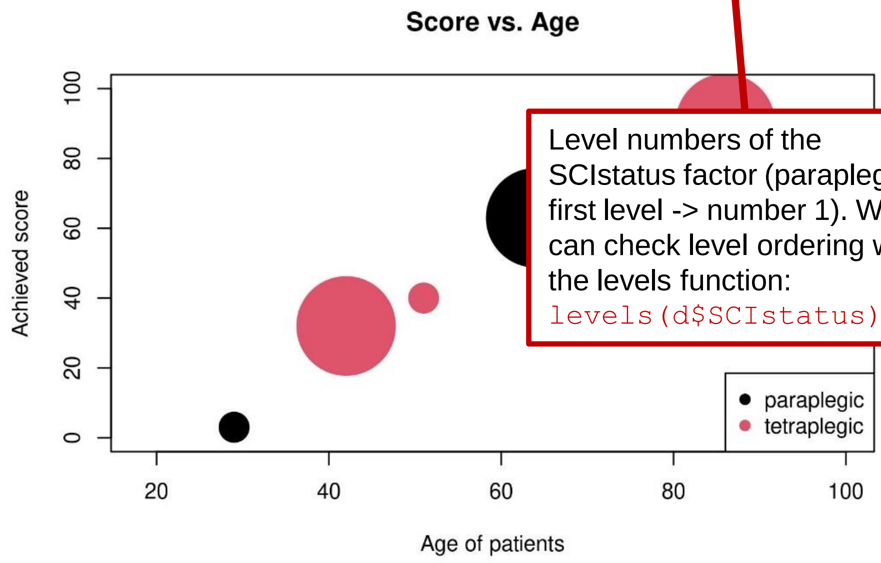
- `plot` can take many additional arguments:

```
> d
     id   SCIstatus    group  score   age
1 pat1  paraplegic     cntrl      3    29
2 pat2  tetraplegic  treatm      32    42
3 pat3  tetraplegic   cntrl      40    51
4 pat4  paraplegic   treatm      63    64
5 pat5  paraplegic    cntrl      74    76
6 pat6  tetraplegic  treatm      90    86

> plot(x = d$age, y = d$score,
       main = "Score vs. Age",
       xlab = "Age of patients",
       ylab = "Achieved score",
       xlim = c(18, 100),
       ylim = c(0, 100),
       pch = 19,
       col = d$SCIstatus,
       cex = c(3, 10))

> legend("bottomright", legend=levels(d$SCIstatus),
         col=1:nlevels(d$SCIstatus), pch=19)
```

Supplied as a factor

-> How the plot function processes the data:

```
 x   y pch col cex
29   3  19   1   3
42  32  19   2  10
51  40  19   2   3
64  63  19   1  10
76  74  19   1   3
86  90  19   2  10
```



Level numbers of the SCIstatus factor (paraplegic is first level -> number 1). We can check level ordering with the levels function: `levels(d$SCIstatus)`

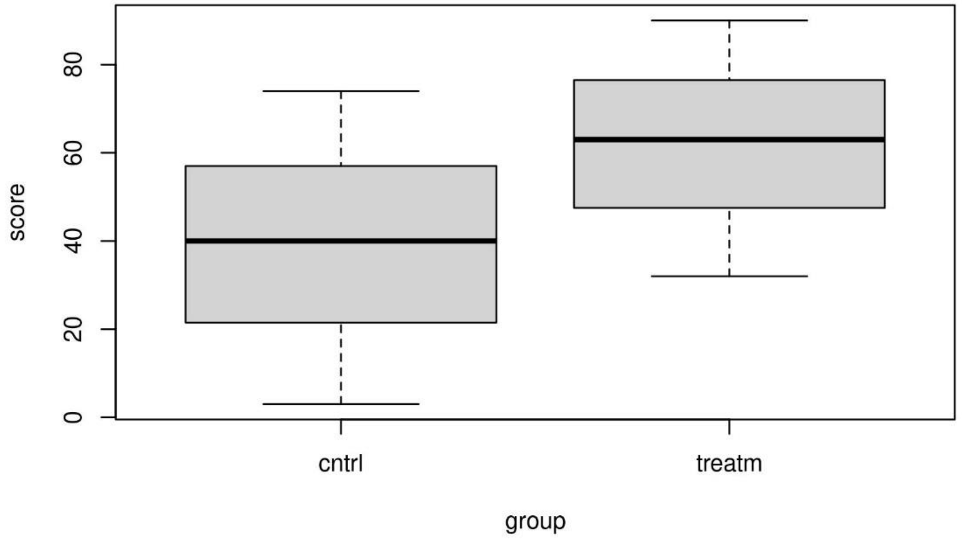
The `plot` function (`boxplot`)

- Plotting a metric variable vs a categorical variable (coded as a **factor**):

```
> d
     id   SCIstatus    group  score  age
1 pat1  paraplegic    cntrl      3   29
2 pat2  tetraplegic  treatm     32   42
3 pat3  tetraplegic   cntrl     40   51
4 pat4  paraplegic   treatm     63   64
5 pat5  paraplegic    cntrl     74   76
6 pat6  tetraplegic  treatm     90   86

> plot(score ~ group, data = d)
```

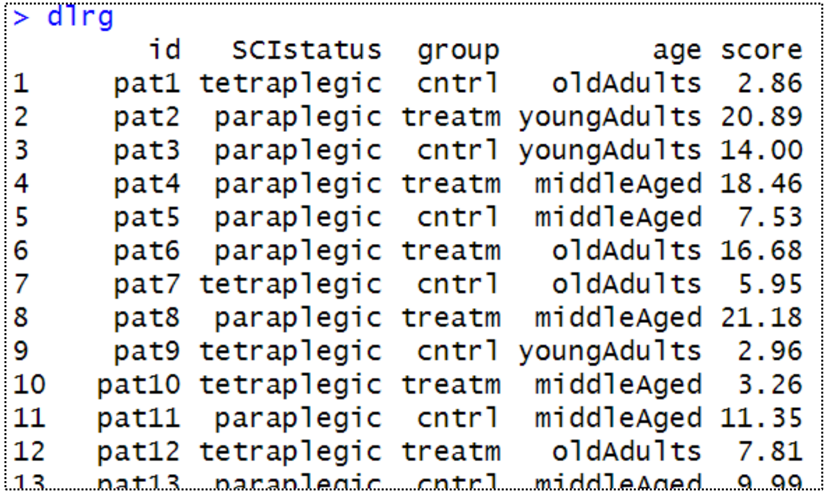
- The plot function internally calls the `boxplot` function, we can also create the same plot using `boxplot` directly:

```
> boxplot(score ~ group, data = d)
```



Creating figures with `ggplot`

- **`ggplot2`** is a very popular package for the creation of (complex) data visualizations
- We will look at its application using this exemplary data frame:



```
> dlrg
        id   SCIstatus  group         age score
1     pat1 tetraplegic  cntrl   oldAdults  2.86
2     pat2  paraplegic treatm youngAdults 20.89
3     pat3  paraplegic  cntrl youngAdults 14.00
4     pat4  paraplegic treatm  middleAged 18.46
5     pat5  paraplegic  cntrl  middleAged  7.53
6     pat6  paraplegic treatm   oldAdults 16.68
7     pat7 tetraplegic  cntrl   oldAdults  5.95
8     pat8  paraplegic treatm  middleAged 21.18
9     pat9 tetraplegic  cntrl youngAdults  2.96
10   pat10 tetraplegic treatm  middleAged  3.26
11   pat11  paraplegic  cntrl  middleAged 11.35
12   pat12 tetraplegic treatm   oldAdults  7.81
13   pat13  paraplegic  cntrl  middleAged  9.99
```

- Install and load ggplot2 package:

```
> install.packages("ggplot2")
> library(ggplot2)
```

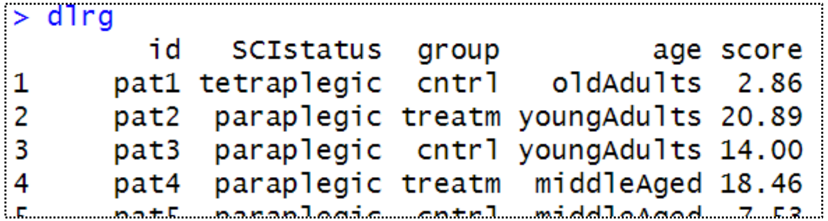
- The syntax of ggplot is different from base plotting functions
  - The commands follow a **layer-by-layer principle**

Creating figures with `ggplot`

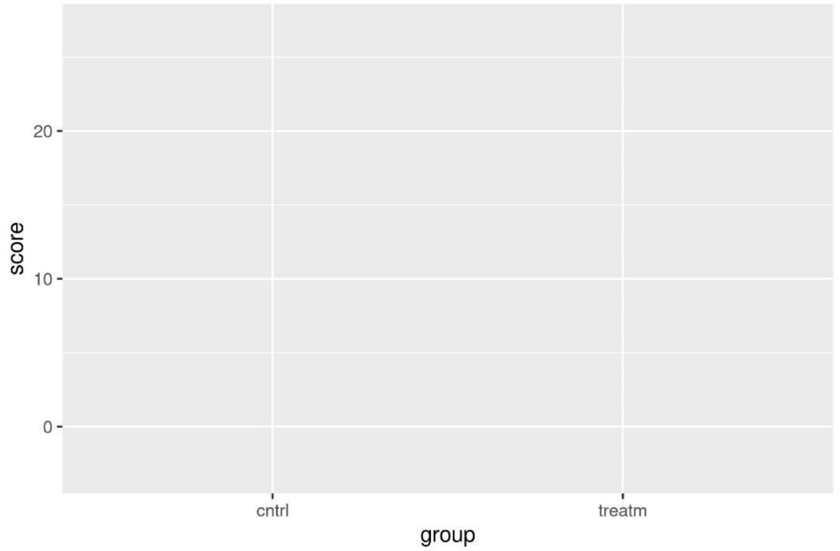
- The commands follow a **layer-by-layer principle**
- The **first layer** is created with the `ggplot` function, it only defines the basic structure of the figure (nothing is yet plotted):

```
> ggplot(data = dlrg,
         mapping = aes(x = group, y = score))
```

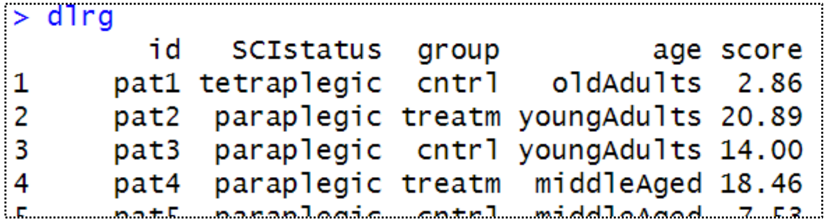
- Two arguments are essential for the `ggplot` function:
  - `data` to pass the data frame
  - `mapping` to define figure structure (using `aes` function)



```
> dlrg
          id    SCIstatus  group          age score
1       pat1 tetraplegic  cntrl    oldAdults  2.86
2       pat2   paraplegic treatm youngAdults 20.89
3       pat3   paraplegic  cntrl youngAdults 14.00
4       pat4   paraplegic treatm  middleAged 18.46
```



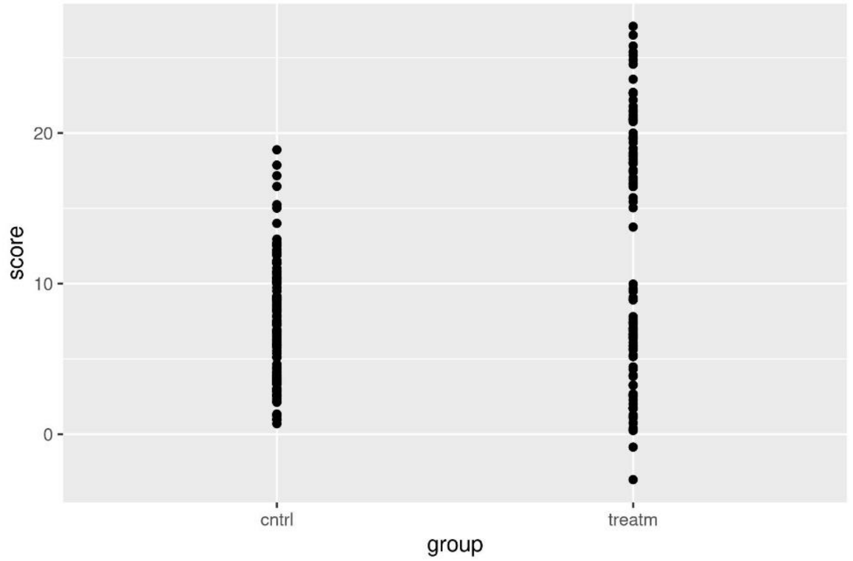
Creating figures with `ggplot`



```
> dlrg
        id   SCIstatus  group         age score
1     pat1 tetraplegic  cntrl   oldAdults  2.86
2     pat2  paraplegic treatm youngAdults 20.89
3     pat3  paraplegic  cntrl youngAdults 14.00
4     pat4  paraplegic treatm  middleAged 18.46
```

- We can add new layers with the + sign
- To add **points**, use the `geom_point` function:

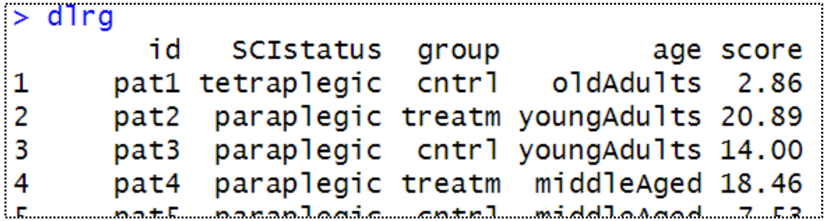
```
> ggplot(data = dlrg,
         mapping = aes(x = group, y = score)) +
         geom_point()
```



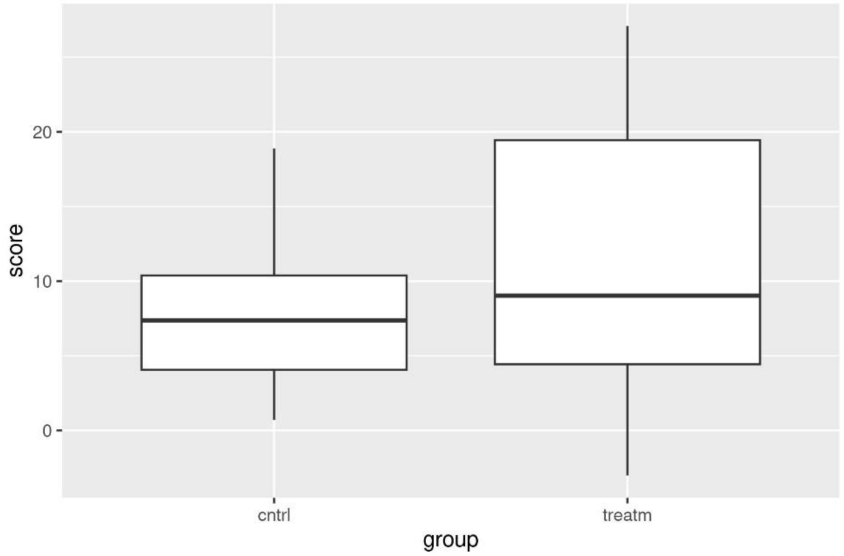
Creating figures with ggplot

- We can add new layers with the + sign
- To add **boxplots**, use the geom_boxplot function:

```
> ggplot(data = dlrg,
         mapping = aes(x = group, y = score)) +
         geom_boxplot()
```



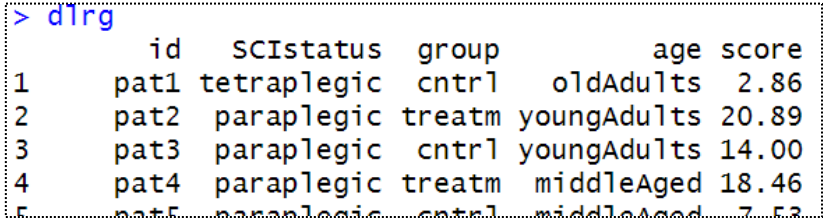
```
> dlrg
       id   SCIstatus  group         age score
1    pat1 tetraplegic  cntrl   oldAdults  2.86
2    pat2  paraplegic treatm youngAdults 20.89
3    pat3  paraplegic  cntrl youngAdults 14.00
4    pat4  paraplegic treatm  middleAged 18.46
```



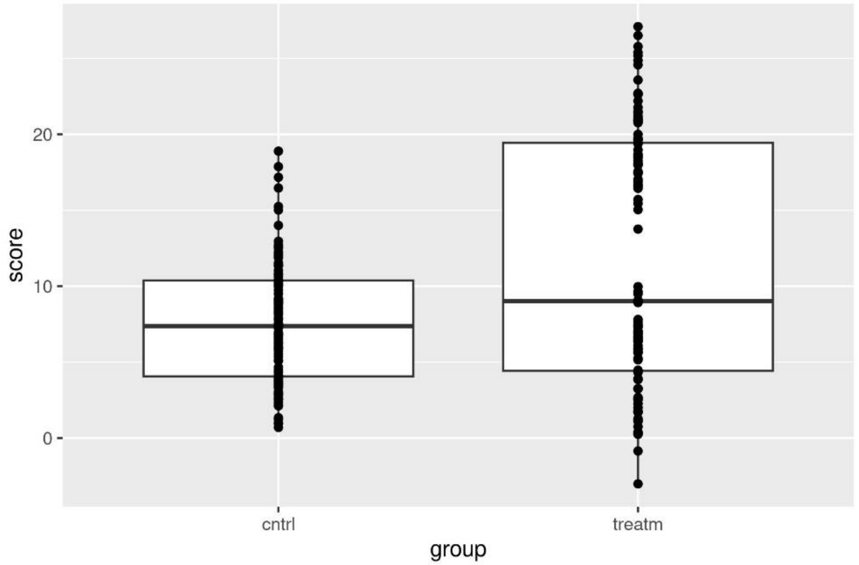
Creating figures with `ggplot`

- We can add new layers with the + sign
- We could also add both layers:

```
> ggplot(data = dlrg,
         mapping = aes(x = group, y = score)) +
         geom_boxplot()+
         geom_point()
```



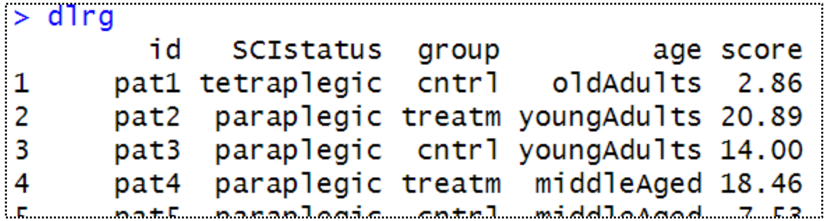
```
> dlrg
          id   SCIstatus  group          age score
1       pat1 tetraplegic  cntrl    oldAdults  2.86
2       pat2  paraplegic treatm youngAdults 20.89
3       pat3  paraplegic  cntrl youngAdults 14.00
4       pat4  paraplegic treatm  middleAged 18.46
```



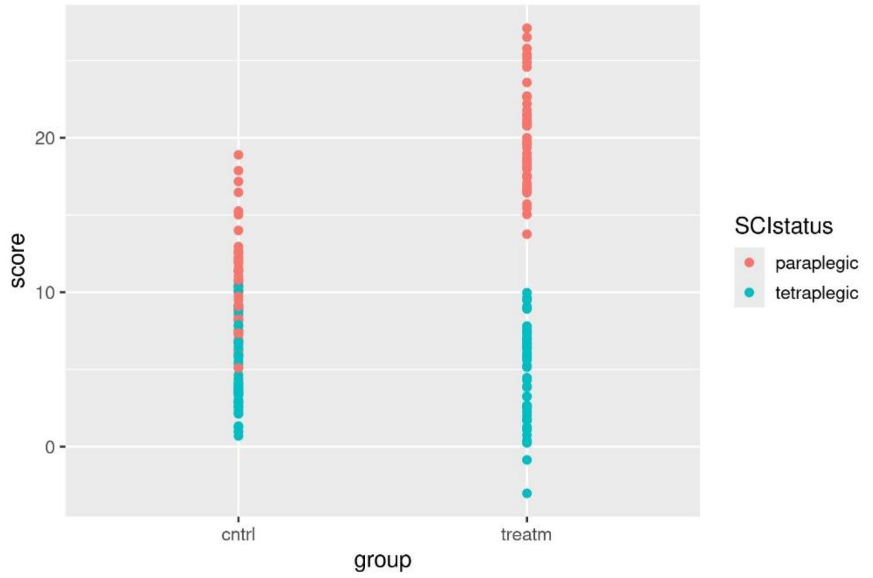
Creating figures with `ggplot`

- We can add new layers with the + sign
- We can change further aesthetics in the `aes` function, e.g. adding colour:

```
> ggplot(data = dlrg,
         mapping = aes(x = group, y = score,
                       colour = SCIstatus)) +
         geom_point()
```



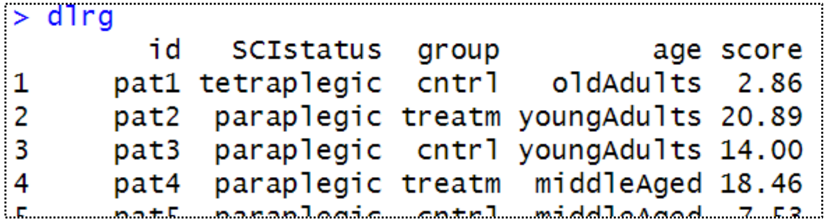
```
> dlrg
          id   SCIstatus  group          age score
1       pat1 tetraplegic  cntrl    oldAdults  2.86
2       pat2  paraplegic treatm youngAdults 20.89
3       pat3  paraplegic  cntrl youngAdults 14.00
4       pat4  paraplegic treatm  middleAged 18.46
5       pat5  paraplegic  cntrl  middleAged  7.53
```



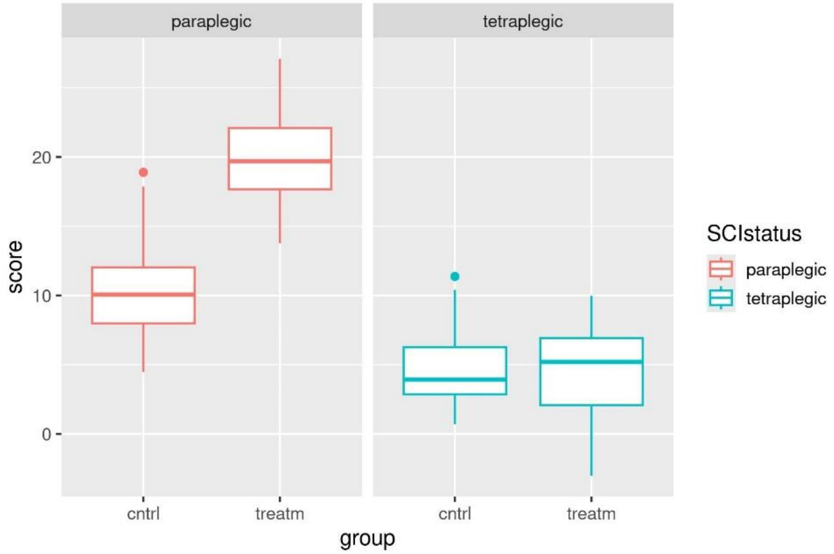
Creating figures with `ggplot`

- We can add new layers with the + sign
- The ggplot function is especially useful for quickly plotting **multiple settings**
  - We can use the `facet_wrap` function for that purpose
  - E.g. splitting the plot along the `group` variable:

```
> ggplot(data = dlrg,
        mapping = aes(x = group, y = score,
                      colour = SCIstatus)) +
        facet_wrap(~ SCIstatus) +
        geom_boxplot()
```



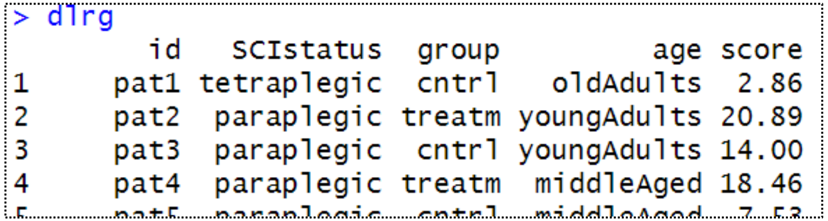
```
> dlrg
         id   SCIstatus  group          age score
1      pat1 tetraplegic  cntrl    oldAdults  2.86
2      pat2  paraplegic treatm youngAdults 20.89
3      pat3  paraplegic  cntrl youngAdults 14.00
4      pat4  paraplegic treatm  middleAged 18.46
```



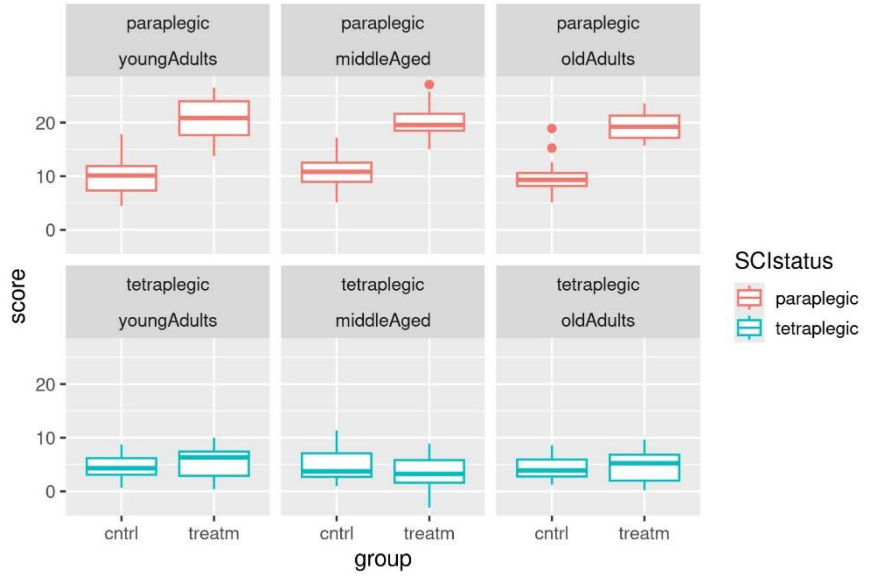
Creating figures with `ggplot`

- We can add new layers with the + sign
- ... adding the `age` variable:

```
> ggplot(data = dlrg,
         mapping = aes(x = group, y = score,
                       colour = SCIstatus)) +
         facet_wrap(~ SCIstatus + age) +
         geom_boxplot()
```



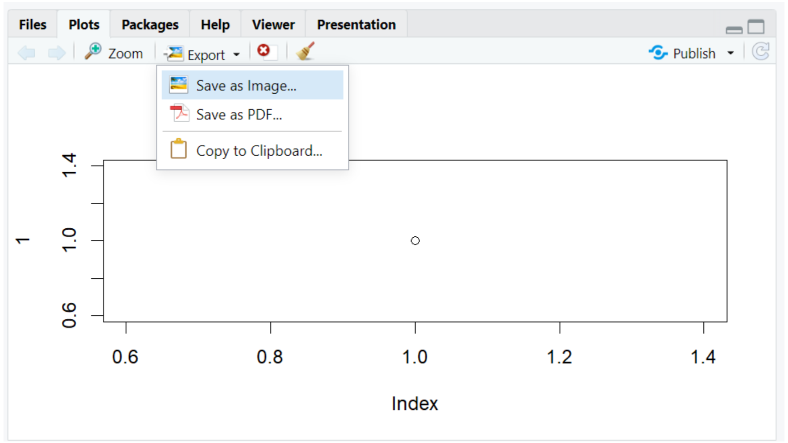
```
> dlrg
          id   SCIstatus  group         age score
1       pat1 tetraplegic  cntrl   oldAdults  2.86
2       pat2  paraplegic treatm youngAdults 20.89
3       pat3  paraplegic  cntrl youngAdults 14.00
4       pat4  paraplegic treatm  middleAged 18.46
```



Exporting a figure

- In Rstudio we can export figures using the graphical interface
  - In the **Plots** tab > **Export** > **Save as Image/PDF**

- Alternatively, we can export figures using specific R commands
  - E.g. the `jpeg` function can be used to export a figure to a jpeg file (similar functions exist for png, bmp, pdf, …):

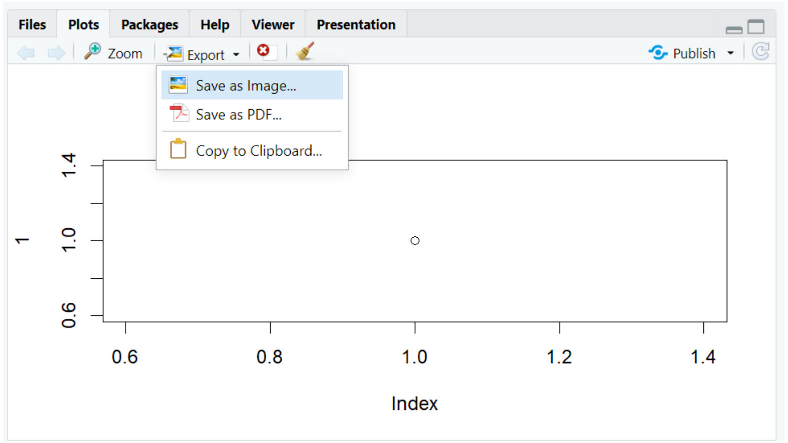


```
> jpeg(filename = "myplot.jpg",
           width = 10, height = 10,
           units = "cm",
           res = 300)
> plot(x = 1, y = 1)
> dev.off()
```

Exporting a figure

- In Rstudio we can export figures using the graphical interface
  - In the **Plots** tab > **Export** > **Save as Image/PDF**

- Alternatively, we can export figures using specific R commands
  - E.g. the `jpeg` function can be used to export a figure to a jpeg file (similar functions exist for png, bmp, pdf, …):



```
> jpeg(filename = "myplot.jpg",
           width = 10, height = 10,
           units = "cm",
           res = 300)
> plot(x = 1, y = 1)
> dev.off()
```

Start jpeg graphics device

R code to create plot

Stop graphics device

→ **Image file will be saved in working directory**

Exercise: Data visualization

- Find the exercise at: https://github.com/Swiss-Paraplegic-Research/Workshop/tree/main/**Part4_DataVisual/Exercise**