

dviz-practice Chapter 1: A quick introduction to base R graphics Assignment: Data Visualization in R

A quick introduction to base R graphics

1 README

- This assignment is based on Pearson's archived DataCamp course "Data Visualization in R" (2016).
- Each chapter comes with one videos, which you should watch before completing the exercises. There are also slides available ([GDrive](#)).
- For the exercises, create or complete R code blocks as needed.
- When you completed all exercises and watched all videos, update the #+AUTHOR: information with your name and (pledged), and submit here.

2 **TODO** A quick introduction to base R graphics

2.1 README

- This assignment is based on Pearson's (now archived) DataCamp course "Data Visualization in R" (2016)
- Make sure you add your name in the #+AUTHOR: meta headline for the honor pledge.
- When you've completed the file, submit it in Canvas
- You'll get solutions after the deadline has passed

2.2 The world of data visualization

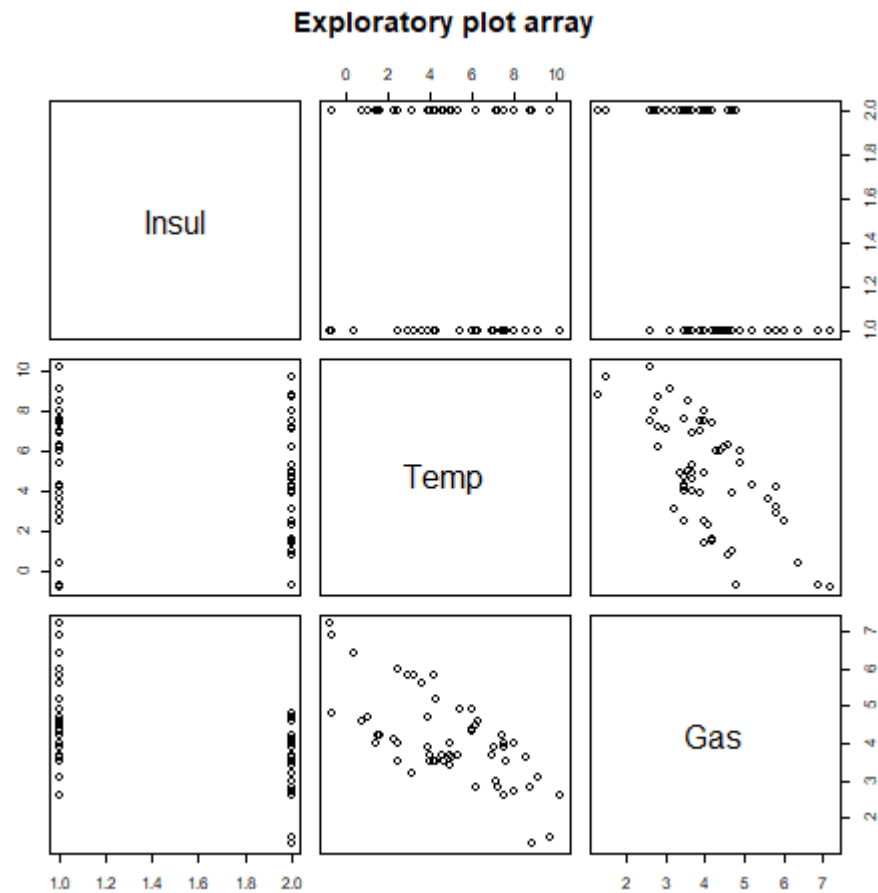
Watch the video and/or look at the slides ([GDrive](#)).

2.3 Creating an exploratory plot array

1. Create a code block that prints graphics output to plot1.png
2. Make the whiteside data set from the MASS package available in your R session
3. Display the structure of the whiteside data set
4. Apply the plot function to the whiteside data frame
5. Title the plot "Exploratory plot array"

— PUT YOUR CODE BELOW THIS LINE —

```
library(MASS)
str(whiteside)
plot(whiteside, main = "Exploratory plot array")
```

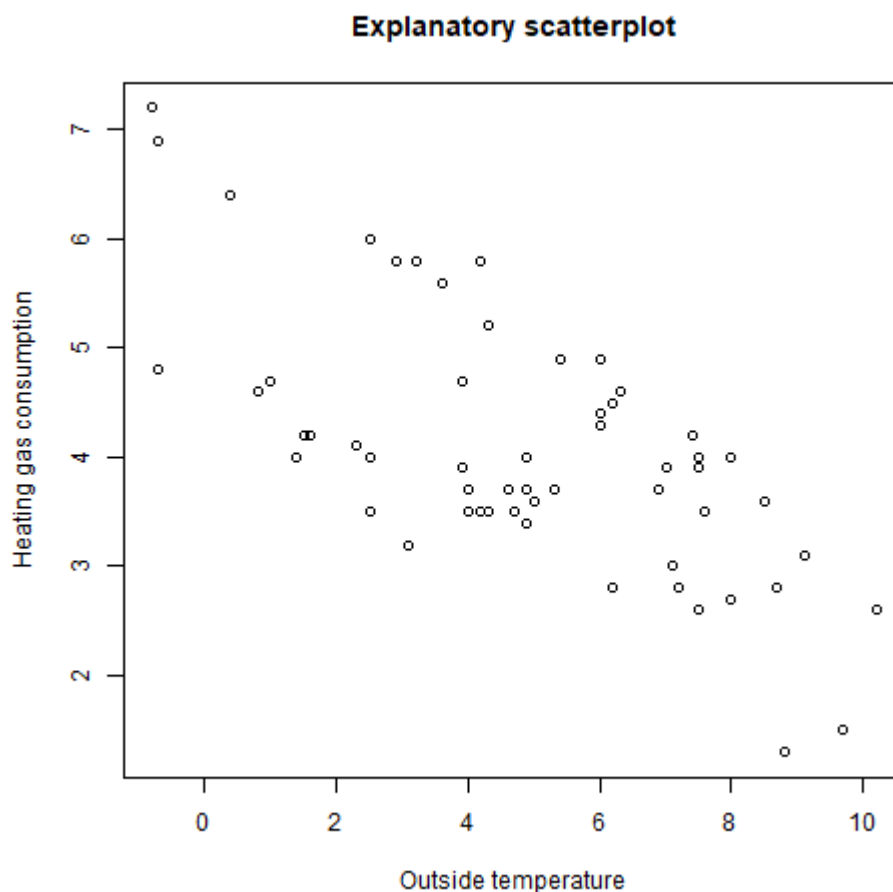


2.4 Creating an explanatory scatterplot

1. Use the `plot` function to construct a scatterplot of the heating gas consumption, `Gas`, versus the outside temperature, `Temp`, from the `whiteside` data frame in the `MASS` package.
2. Label the x- and y-axes to indicate the variables in the plot, e.g. `~"Outside temperature"`, and `"Heating gas consumption"`, resp.
3. Title the plot `"Explanatory scatterplot"`
4. Print the plot to a file `plot2.png`

— PUT YOUR CODE BELOW THIS LINE —

```
plot(
  x = whiteside$Temp,
  y = whiteside$Gas,
  xlab = "Outside temperature",
  ylab = "Heating gas consumption",
  main = "Explanatory scatterplot")
```

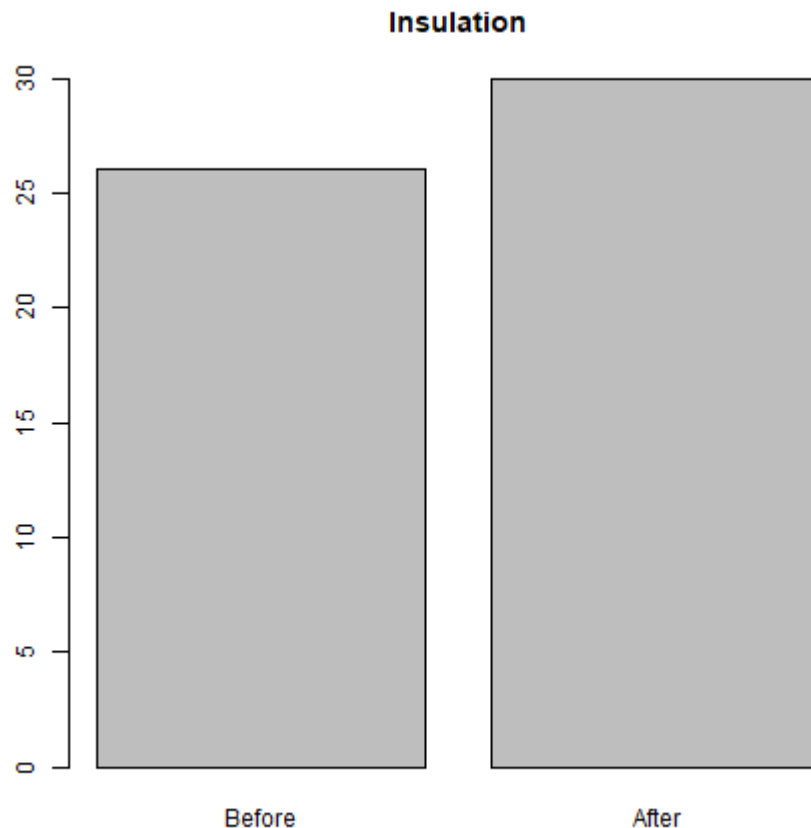


2.5 The `plot()` function is generic

1. Apply the `plot` function to the `Insul` variable from the `whiteside` data frame in the `MASS` package.
2. Title the plot "Insulation"
3. Name the resulting plot `plot3.png`

— PUT YOUR CODE BELOW THIS LINE —

```
plot(  
  x = whiteside$Insul,  
  main = "Insulation")
```



2.6 A preview of some more and less useful techniques

Watch the video and/or look at the slides ([GDrive](#)).

2.7 Adding details to a plot using point shapes, color, and reference lines

1. Create an R code block to check that the data set `Cars93` from the `MASS` package is in your work space, and to look at its structure.
2. Create another R code block that stores output in the file `plot4.png`
3. Use `plot` to create a scatterplot of the `Max.Price` variable vs. the `Price` variable.
4. Specify `pch` and `col` parameters so that the data points are represented as red solid triangles. The `pch` value for solid triangle symbols is 17.
5. Use the `points` function to add a second set of points to your scatterplot, representing `Min.Price` versus `Price`.
6. Specify the new data points as blue solid circles. The `pch` value for solid circles is 16.
7. Use the `abline` function to add a dashed equality reference line (i.e. a line with y-intercept 0 and slope 1). Check `help(abline)` to find out what its arguments refer to. The `lty` value for a dashed line is 2.
8. Give your plot a suitable title, and label the axis appropriately. You can either do this in the `plot` function as a `main` parameter, or use the `title` function.

— PUT YOUR CODE BELOW THIS LINE —

```
library(MASS)
str(Cars93)
```

```
'data.frame': 93 obs. of 27 variables:
 $ Manufacturer : Factor w/ 32 levels "Acura","Audi",...: 1 1 2 2 3 4 4 4 4 5 ...
 $ Model        : Factor w/ 93 levels "100","190E","240",...: 49 56 9 1 6 24 54 74 73 35 ..
 $ Type         : Factor w/ 6 levels "Compact","Large",...: 4 3 1 3 3 3 2 2 3 2 ...
 $ Min.Price    : num 12.9 29.2 25.9 30.8 23.7 14.2 19.9 22.6 26.3 33 ...
 $ Price        : num 15.9 33.9 29.1 37.7 30 15.7 20.8 23.7 26.3 34.7 ...
 $ Max.Price    : num 18.8 38.7 32.3 44.6 36.2 17.3 21.7 24.9 26.3 36.3 ...
 $ MPG.city     : int 25 18 20 19 22 22 19 16 19 16 ...
 $ MPG.highway  : int 31 25 26 26 30 31 28 25 27 25 ...
 $ AirBags      : Factor w/ 3 levels "Driver & Passenger",...: 3 1 2 1 2 2 2 2 2 2 ...
 $ DriveTrain   : Factor w/ 3 levels "4WD","Front",...: 2 2 2 2 3 2 2 3 2 2 ...
 $ Cylinders    : Factor w/ 6 levels "3","4","5","6",...: 2 4 4 4 2 2 4 4 4 5 ...
 $ EngineSize   : num 1.8 3.2 2.8 2.8 3.5 2.2 3.8 5.7 3.8 4.9 ...
 $ Horsepower   : int 140 200 172 172 208 110 170 180 170 200 ...
 $ RPM          : int 6300 5500 5500 5500 5700 5200 4800 4000 4800 4100 ...
 $ Rev.per.mile : int 2890 2335 2280 2535 2545 2565 1570 1320 1690 1510 ...
 $ Man.trans.avail : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 1 1 1 ...
 $ Fuel.tank.capacity: num 13.2 18 16.9 21.1 21.1 16.4 18 23 18.8 18 ...
 $ Passengers   : int 5 5 5 6 4 6 6 6 5 6 ...
 $ Length       : int 177 195 180 193 186 189 200 216 198 206 ...
 $ Wheelbase    : int 102 115 102 106 109 105 111 116 108 114 ...
 $ Width        : int 68 71 67 70 69 69 74 78 73 73 ...
 $ Turn.circle  : int 37 38 37 37 39 41 42 45 41 43 ...
 $ Rear.seat.room : num 26.5 30 28 31 27 28 30.5 30.5 26.5 35 ...
 $ Luggage.room : int 11 15 14 17 13 16 17 21 14 18 ...
 $ Weight       : int 2705 3560 3375 3405 3640 2880 3470 4105 3495 3620 ...
 $ Origin       : Factor w/ 2 levels "USA","non-USA": 2 2 2 2 2 1 1 1 1 1 ...
 $ Make        : Factor w/ 93 levels "Acura Integra",...: 1 2 4 3 5 6 7 9 8 10 ...
```

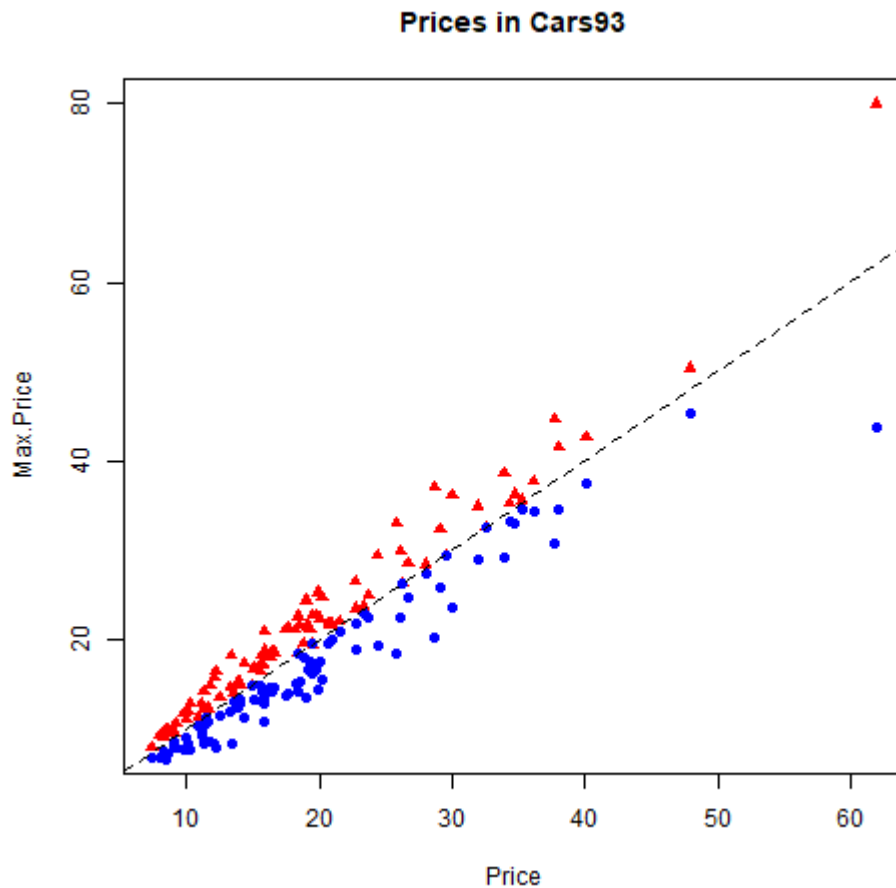
```
library(MASS)

plot(
  data=Cars93,          # generic plot
  Max.Price ~ Price,    # data frame
  pch=17,               # y = Max.Price, x = Price
  col="red")            # point symbol triangle
                        # point color red

points(
  data=Cars93,          # add some points to existing plot
  Min.Price ~ Price,
  pch=16,
  col="blue")

abline(
  a=0,                  # draw a straight line
  b=1,                  # intercept is 0
  lty=2)                # slope is 1
                        # dashed line type

title(main="Prices in Cars93")
```



2.8 Creating multiple plot arrays

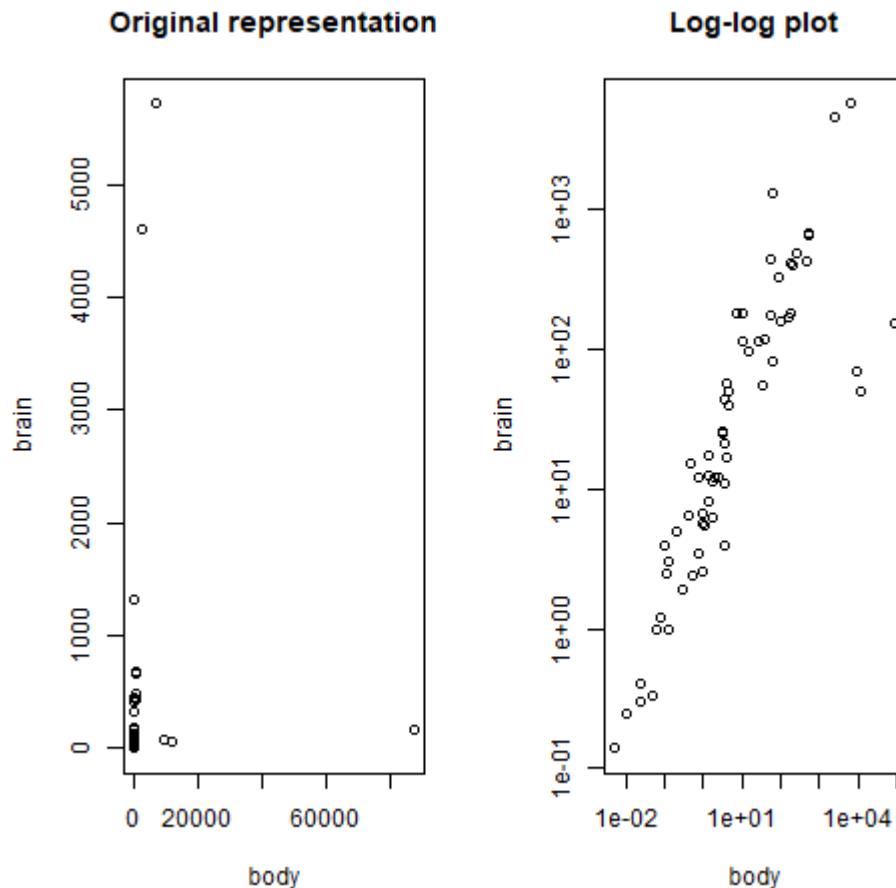
1. Create an R code block and look at the structure of the `Animals2` data set in the `robustbase` package.
2. Create another R code block that writes graphics output to a file `plot5.png`.
3. Use the `par` function and set the `mfrow` parameter to create a side-by-side plot array with 1 row and 2 columns.
4. Use the `plot` function to create a scatterplot of the variables `brain` vs. `body` from the `Animals2` data frame, without customization.
5. Add the title "Original representation" to the plot using the `title` function.
6. Add another `plot` command in the same code block, with the same variables, but add the parameter `log="xy"` to the arguments. This creates a plot of both variables in log scale.
7. Add the title "Log-log plot" to the plot using the `title` function.

— PUT YOUR CODE BELOW THIS LINE —

```
library(robustbase)
str(Animals2)
```

```
'data.frame': 65 obs. of 2 variables:
 $ body : num 1.35 465 36.33 27.66 1.04 ...
 $ brain: num 8.1 423 119.5 115 5.5 ...
```

```
par(mfrow=c(1,2))
plot(brain ~ body, data=Animals2)
title("Original representation")
plot(brain ~ body, data=Animals2, log="xy")
title("Log-log plot")
```



2.9 Avoid pie charts

1. Create an R code block and look at the structure of the `dataCar` data set in the `insuranceData` package. Remember that you may have to install packages (on the R console, not in the Org-mode file), and that you must load packages (with `library`) and sometimes load data sets, too (with `data`).
2. Create a new R code block that writes graphics output to the file `plot6.png`.
3. Set up a side-by-side plot array with 1 row and 2 columns.
4. Use `table` to create a table `tbl` of counts of the distinct levels of the `veh_body` variable in the `dataCar` data frame.
5. Use `sort` to sort the table `tbl`, and set the parameter `decreasing=TRUE` to sort in descending order. Store the sorted table in `tbl_sorted`.
6. Pass `tbl_sorted` as the argument to the plotting function `pie`. This will create a pie chart.
7. Use `title` to title this plot "Pie chart".
8. Use the plotting function `barplot` and the function `title` to create a barplot titled "Bar chart" from the data of `tbl_sorted`.

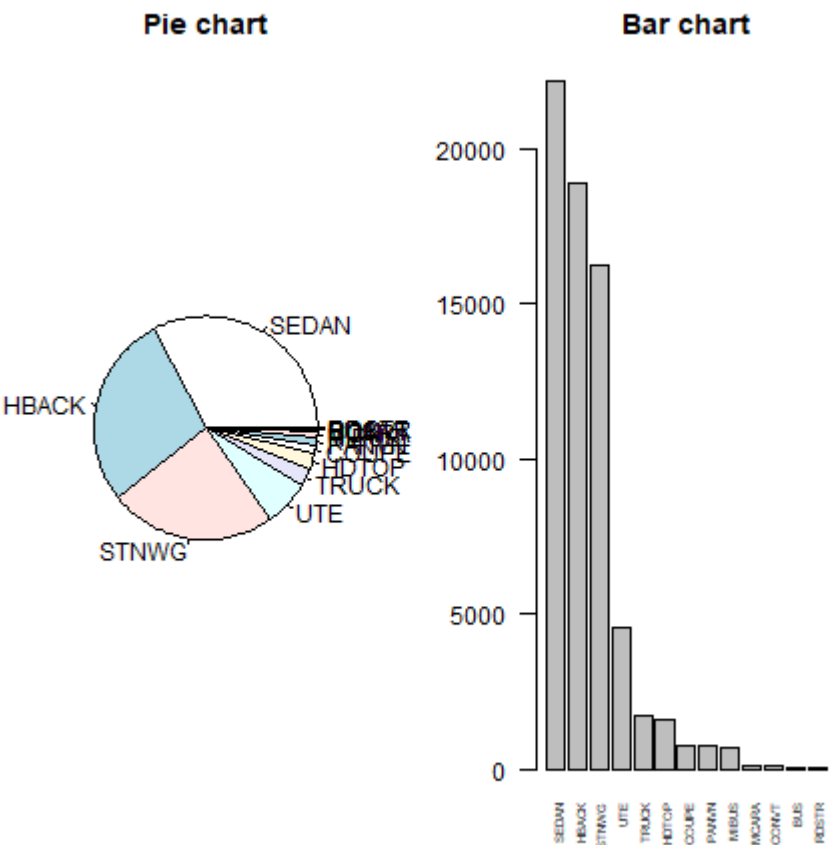
9. Inside `barplot`, set the parameters `las=2` to make the sets of x- and y-labels perpendicular to the axes, and `cex.names=0.5` to make the name labels half the default size.

— PUT YOUR CODE BELOW THIS LINE —

```
library(insuranceData)
data(dataCar)
str(dataCar)
```

```
'data.frame': 67856 obs. of 11 variables:
 $ veh_value: num 1.06 1.03 3.26 4.14 0.72 2.01 1.6 1.47 0.52 0.38 ...
 $ exposure : num 0.304 0.649 0.569 0.318 0.649 ...
 $ clm      : int 0 0 0 0 0 0 0 0 0 0 ...
 $ numclaims: int 0 0 0 0 0 0 0 0 0 0 ...
 $ claimcst0: num 0 0 0 0 0 0 0 0 0 0 ...
 $ veh_body : Factor w/ 13 levels "BUS","CONVT",...: 4 4 13 11 4 5 8 4 4 4 ...
 $ veh_age  : int 3 2 2 2 4 3 3 2 4 4 ...
 $ gender   : Factor w/ 2 levels "F","M": 1 1 1 1 1 2 2 2 1 1 ...
 $ area     : Factor w/ 6 levels "A","B","C","D",...: 3 1 5 4 3 3 1 2 1 2 ...
 $ agecat   : int 2 4 2 2 2 4 4 6 3 4 ...
 $ X_OBSTAT_: Factor w/ 1 level "01101 0 0 0": 1 1 1 1 1 1 1 1 1 1 ...
```

```
par(mfrow=c(1,2))
tbl <- table(dataCar$veh_body)
tbl_sorted <- sort(tbl, decreasing=TRUE)
pie(tbl_sorted)
title("Pie chart")
barplot(tbl_sorted, las=2, cex.names=0.5)
title("Bar chart")
```

Author: [yourname] (pledged)
Created: 2022-09-22 Thu 18:11