Въведение в Статистиката с R – IV – Систематизиране на многомерни данни

В тази част ще опишем как може да бъде систематизирана информацията при работа с многомерни данни и как после това систематизиране може да ни бъде полезно.

1. Съхраняване и достигане на многомерни данни в data frames

Вече знаем, че след събирането на данните често е удачно те да бъдат съхранени в таблица, на всеки ред, на която съответства една статистическа единица, а на всяка колона един статистически признак(променлива).
R използва data frames за да съхрани тези променливи на едно място. Той има много функции за пряк достъп до данните, съхранени по този начин.
Можем да използваме таблица, направена например в Ексел, или да създадете направо data frame в R. Ако данните ни са съхранени в правоъгълен масив – тогава сме готови. Просто използваме функцията

data.frame

По-често обаче това не е вярно и тогава трябва да си направим този масив.
Да предположим, че имаме ръст, височина и пол на 6 човека и данните за тях са подадени в 3 различни променливи както по-долу.

```
> weight = c(150, 135, 210, 140)
> height = c(65, 61, 70, 65)
> gender = c("Fe","Fe","M","Fe")
> study = data.frame(weight,height,gender) # make the data frame
> study
  weight height gender
1    150     65     Fe
2    135     61     Fe
3    210     70      M
4    140     65     Fe
```

Тук колоните наследяват имената на променливите. Тези имена могат да бъдат сменени

```
> study = data.frame(w = weight, h = height, g = gender)
    w   h   g
1 150  65  Fe
2 135  61  Fe
3 210  70   M
4 140  65  Fe
```

Редовете също могат да бъдат именувани. Например за по-голяма яснота, за коя статистическа единица се отнасят. Например в случая

```
> row.names(study) = c("Mary","Alice","Bob","Judy")
> study
        w   h   g
Mary  150  65  Fe
Alice 135  61  Fe
Bob   210  70   M
Judy  140  65  Fe
```

Със същата команда имената на редовете могат да бъдат сменяни.

```
> row.names(study) = c("M","A","B","J")
> study
    w   h   g
M 150  65  Fe
A 135  61  Fe
```

B 210 70 M
J 140 65 Fe

Достъп до данни от data frame – Вече знаем, че за да достигнем до дадена променлива от таблица можем да използваме функцията attach и после променливите са достъпни със своите имена или като използваме конструкцията *име на таблица$име на променлива*

```
> study$w
[1] 150 135 210 140
> study$h
[1]  65 61 70 65
> study$g
[1]  Fe Fe M  Fe
Levels: Fe M
```

или

```
> attach(study)
> w
[1] 150 135 210 140
> h
[1] 65 61 70 65
> g
[1] Fe Fe M  Fe
Levels: Fe M
> detach(study)
```

Действието на командата attach се прекратява с командата detach.
При използването на командата attach трябва да се внимава защото ако след нея променяме някои от стойностите на променливите, те не се променят в изходната таблица. Например:

```
> attach(study)
> w[2]=100
> study
     w    h  g
M 150 65 Fe
A  135 61 Fe
B  210 70  M
J   140 65 Fe
> w
[1] 150 100 210 140
> detach(study)
```

Достъпът до данни направо от data frame става по аналогичен начин на достъп до данни от матрица. Чрез написване в квадратни скоби на номера на реда и номера на колоната на съответния елемент.

```
> study[1,2]
[1] 65
```

Същия резултат можем да получим ако напишем името на реда и името на колоната на съответния елемент.

```
> study['M','w']
[1] 150
```

Ако оставим номера на реда (или номера на колоната) празни ние достигаме до съответната колона(ред), чиито номер е попълнен.

```
> study[,1]  # достигаме до всички редове от първата колона
```

По аналогичен начин на номерата могат да бъдат използвани имената на редовете и колоните.

```
> study[,'w'] # достигаме до всички редове от колона с име weight
[1] 150 135 210 140
```
Можем да достигнем едновременно до повече от една колона
```
> study[,1:2]
   w  h
M 150 65
A  135 61
B  210 70
J   140 65
```
Да обърнем внимание, че функцията ***rm*** служи за премахване на променливи само от временната памет, но те остават в таблицата с данни.
```
> rm(w)
> w
Error: object 'w' not found
> study
    w    h   g
M  150   65  Fe
A   135  61  Fe
B   210  70   M
J   140   65  Fe
```
Можем да достигнем до цял ред, написвайки неговото име или номер. Например
```
>  study['M',]
    w   h  g
M  150 65  Fe
>  study[1,]
    w   h   g
M  150  65  Fe
```

Достъп до данни от list – Листът е по-обща концепция от data frame, защото тай е множество от обекти, всеки, от които може да съдържа други обекти. Data frame е лист, който може да съдържа само вектори стълбове.
За да се достигне до елементите на list може да се използва $ или двойни квадратни скоби [[ ]]. Например в нашата таблица study можем да достигнем до променливата w (първата колона) по всеки един от следните начини
```
> study$w
[1] 150 135 210 140
> study[['w']] # използвайки името
> study[[1]]   # използвайки номера на колоната.
[1] 150 135 210 140
```
Т.е. ако разглеждаме study като data frame ще постигнем същия резултат чрез командата
```
> study[,1]
[1] 150 135 210 140
```
Да обърнем внимание, че ако напишем само единични скоби, т.к. в смисъла на извеждане на елемент на data frame това е неразбираемо, по подразбиране R извежда цялата колона, но по по-различен начин, като стълб.
```
> study[1]
    w
M  150
A   135
B   210
J   140
```

За да се достигне само до данни за жените можем да постъпим по следния начин

```
> study[study$g == 'Fe', ]
   w   h  g
M 150  65 Fe
A 135  61 Fe
J 140  65 Fe
```

Оформяне на признаци по подгрупи в data frames чрез stack и unstack

Нека разгледаме множеството от данни PlantGrowth, което съдържа 30 наблюдения за теглата на растения, разделени в три групи, една контролна и две "обработвани с…"

```
> data(PlantGrowth)
> head(PlantGrowth)
  weight  group
1   4.17   ctrl
2   5.58   ctrl
3   5.18   ctrl
4   6.11   ctrl
5   4.50   ctrl
6   4.61   ctrl
...
```

Ако искаме да сравним например графиките с мустачки на разпределението на растенията според теглото им поотделно в трите групи можем да подходим по два начина.

Единият е поотделно за всяка група да достигнем до данните от тази група

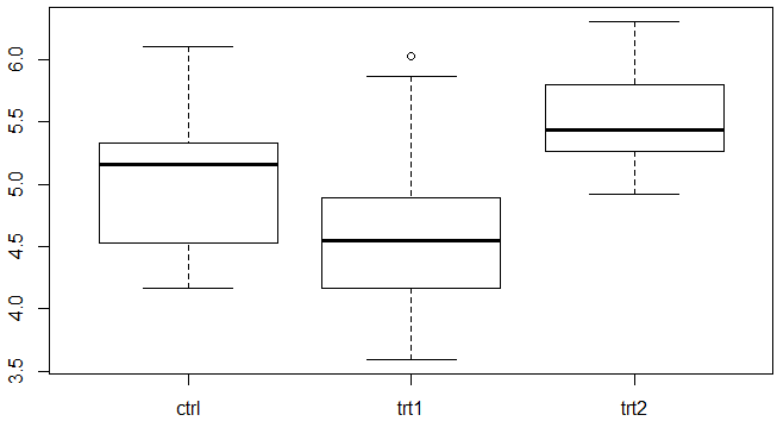
```
> attach(PlantGrowth)
> weight.ctrl = weight[group == "ctrl"]
```

и за тях да изчертаем графики с мустачки. Това би отнело известно време и поради повтарянето на едни и същи действия би било уморително. По-кратко е да използваме съвместно функциите unstuck и boxplot. Първата от тях, при правилно подадени данни (един количествен и един качествен признак), прави нови променливи за метрирания признак, поотделно в отделните подгрупи, обособени според значенията(levels - нивата) на качествения признак.

```
>    unstack(PlantGrowth)
>    ctrl  trt1  trt2
1    4.17 4.81 6.31
2    5.58 4.17 5.12
3    5.18 4.41 5.54
4    6.11 3.59 5.50
5    4.50 5.87 5.37
6    4.61 3.83 5.29
7    5.17 6.03 4.92
8    4.53 4.89 6.15
9    5.33 4.32 5.80
10   5.14 4.69 5.26
```

Тогава за да се направят boxplot поотделно в трите групи може да се използва командата

```
> boxplot(unstack(PlantGrowth))
```



2. Използване на символа “~” за задаване на модели в R

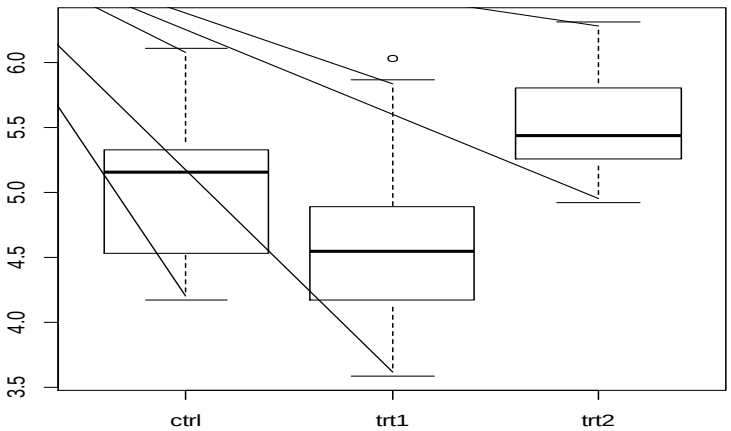
Символът “~” може да бъде използван в R при анализиране на зависимости, за отделяне на зависимата променлива от независимата променлива. В случая, когато и двата признака са метрирани може да бъде зададена и форма на зависимост между тях. Изглежда малко объркващо, но за напредналите е удобно и се използва много често при анализите в R. Вече използвахме този символ при анализиране на зависимости на един количествен (метриран) признак от един качествен (неметриран). В този случай синтаксисът на формулата е

количествен (зависима пром., response) ~ качествен (незав. пром., predictor).

Какво прави този символ? Той разделя количествената променлива в подгрупи, според значенията на качествената променлива. Може да се чете като „Моделирай количествената променлива по отделно по значенията на качествената променлива“ или „Раздели количествената променлива в подгрупи, според различните значения на качествената променлива“. При две променливи е сравнително лесно.

Например за да илюстрираме това да използваме таблицата PlantGrowth.

```
> attach(PlantGrowth)
> boxplot(weight ~ group)
```



При повече от две променливи този символ може да бъде използван с много формули. При това, при описването им, обичайните символи за аритметични операции не правят това, за което сме свикнали да ги използваме. Ето няколко различни приложения.

Да предположим, че работим с променливите Y, X, X1 и X2. Смисълът на формулата

- **Y ~ X** е Y моделирано по значенията на X, т.е. ако и двете променливи са количествени, това е все едно оценяваме коефициентите и параметрите на модела
$$Y = a + bX + \varepsilon.$$
- **Y ~ -1 + X1** е Y моделирано по значенията на X1 без сечение (-1 означава без сечение), т.е. линията на регресия минава през координатното начало. Т.е. ако и двете променливи са количествени, това е все едно оценяваме коефициентите и параметрите на модела
$$Y = bX + \varepsilon.$$
- **Y ~ X1 + X2** е Y моделирано по значенията на X1 и X2 както при многомерните регресионни модели. По принцип ако във формула сложим + и нова величина, това може да се чете като "включи тази величина в анализа".
- **Y ~ X1: X2** е Y моделирано по значенията само на взаимодействията между X1 и X2.
- **Y ~ X1 * X2** е Y моделирано по значенията на X1, X2 и X1*X2. По принцип ако във формула сложим * и нова величина, това може да се чете като "включи тази величина и взаимодействията с нея в анализа". Друг запис на същото е (**Y ~ (X1 + X2)^2**) Two-way interactions. Обърнете внимание, че знакът за степен тук не означава степен. По аналогичен начин ^3 без I отпред означава: "в модела да се включат всички взаимодействия до трети ред".

Например следващите три израза имат един и същ смисъл
y ~ u + v + w + u:v + u:w + v:w + u:v:w
y ~ u * v * w
y ~ (u + v + w)^3
Всички те описват, че в модела са включени независимите променливи u, v и w и всички взаимодействия между тях.

- **Y ~ X1+ I(X2^2)** е Y моделирано по значенията на X1 и $X2^2$. Т.е. ако използваме степени или други алгебрични изрази трябва да сложим израза в скоби и пред него да напишем I. Това означава, че вместо променливата ще използваме нейната втора степен или съответната аритметична функция.
- **Y ~ X1 | X2** е Y моделирано по значенията на X1 при условие X2.
- **Минусът** пред независим признак означава, че тази променлива или взаимодействие се изключват от модела.

Например следващите три израза имат един и същ смисъл
y ~ u + v + w + u:v + u:w + v:w
y ~ u * v * w - u:v:w
y ~ (u + v + w)^2
и той е изтрий взаимодействията от трети ред, но включи взаимодействията от втори ред.
Навсякъде типа на променливите определя формата на анализ.
Ако всички признаци са количествени имаме многомерен регресионен анализ.
Ако независимите променливи са качествени имаме дисперсионен анализ (ANOVA).
Трябва да отбележим, че смисълът на „моделирано по значенията на" може да бъде различен в зависимост от употребата на този символ с конкретна функция в R. Например, при употреба на символа "~" в комбинация с функцията ***boxplot*** неговият смисъл е по-различен отколкото при употреба на този символ с командата ***lm*** за построяване на линейни регресионни модели. Освен това, когато се използват математическите символи, в смисъла на алгебрични операции, те трябва да бъдат сложени в скоби и пред скобите да има символа *I*.

3. Начини за онагледяване на многомерни данни

Групиране на данни в таблици

R разполага с много възможности за онагледяване на многомерни данни. Например n-мерните кръстосани таблици са аналог на двумерните кръстосани таблици. Те се построяват с помощта на функцията ***table***.

Ако w,x,y,z са 4 променливи, тогава командата ***table(x,y)*** създава кръстосана таблица, ***table(x,y,z)*** създава кръстосани таблици на x и y по отделно за всяка различна фиксирана стойност на z. Накрая командата ***table(x,y,z,w)*** ще направи кръстосани таблици на x и y по отделно за всяка различна фиксирана комбинация от стойности на z и w.

Ако променливите са съхранени в data frame, да кажем нека той да се казва df, тогава командата table(df) ще се държи по описания по-горе начин, все едно всяка променлива е отделен вектор и колоните са подредени в съответния ред.

За да илюстрираме това нека разгледаме зависимостите между признаците, описани в множеството от данни Cars93, което се намира в библиотеката MASS.

```
> library(MASS); data(Cars93); attach(Cars93)
> ls(Cars93)
 [1] "AirBags"          "Cylinders"        "DriveTrain"       "EngineSize"
 [5] "Fuel.tank.capacity" "Horsepower"     "Length"           "Luggage.room"
 [9] "Make"             "Man.trans.avail"  "Manufacturer"     "Max.Price"
[13] "Min.Price"        "Model"            "MPG.city"         "MPG.highway"
[17] "Origin"           "Passengers"       "Price"            "Rear.seat.room"
[21] "Rev.per.mile"     "RPM"              "Turn.circle"      "Type"
[25] "Weight"           "Wheelbase"        "Width"
```

	Manufacturer	Model	Type	Min.Price	Price	Max.Price	MPG.city	MPG.highway	AirBags
1	Acura	Integra	Small	12.9	15.9	18.8	25	31	None
2	Acura	Legend	Midsize	29.2	33.9	38.7	18	25	Driver & Passenger
3	Audi	90	Compact	25.9	29.1	32.3	20	26	Driver only
4	Audi	100	Midsize	30.8	37.7	44.6	19	26	Driver & Passenger
5	BMW	535i	Midsize	23.7	30.0	36.2	22	30	Driver only
6	Buick	Century	Midsize	14.2	15.7	17.3	22	31	Driver only

	DriveTrain	Cylinders	Engine Size	Horsepower	RPM	Rev.per.mile	Man.trans.avail	Fuel.tank.capacity
1	Front	4	1.8	140	6300	2890	Yes	13.2
2	Front	6	3.2	200	5500	2335	Yes	18.0
3	Front	6	2.8	172	5500	2280	Yes	16.9
4	Front	6	2.8	172	5500	2535	Yes	21.1
5	Rear	4	3.5	208	5700	2545	Yes	21.1
6	Front	4	2.2	110	5200	2565	No	16.4

	Passengers	Length	Wheelbase	Width	Turn.circle	Rear.seat.room	Luggage.room	Weight	Origin	Make
1	5	177	102	68	37	26.5	11	2705	non-USA	Acura Integra
2	5	195	115	71	38	30.0	15	3560	non-USA	Acura Legend
3	5	180	102	67	37	28.0	14	3375	non-USA	Audi 90
4	6	193	106	70	37	31.0	17	3405	non-USA	Audi 100
5	4	186	109	69	39	27.0	13	3640	non-USA	BMW 535i
6	6	189	105	69	41	28.0	16	2880	USA	Buick Century

Да направим групировка според типа на колите

```
> table(Type);
Compact Large Midsize Small Sporty Van
     16     11      22     21     14     9
```

Ще превърнем цената в категорийна променлива, използвайки функцията cut.

```
>  price = cut(Price, c(0, 12, 20, max(Price)))
```

Ще преименуваме нивата

```
>  levels(price)=c("cheap", "okay", "expensive")
```

Сега да направим кръстосана таблица по цената, разгледана като категорийна променлива и тип

```
> table(price, Type)
```

	Type					
price	Compact	Large	Midsize	Small	Sporty	Van
cheap	3	0	0	18	1	0
okay	9	3	8	3	9	8
expensive	4	8	14	0	4	1

По аналогичен начин ще постъпим с изразходваното количество гориво при извън градско каране.

```
>  mpg = cut(MPG.highway,c(0,20,30,max(MPG.highway)))
>  levels(mpg) = c("gas guzzler","okay","miser")
# now look at the relationships
> table(price, Type, mpg)
, , mpg = gas guzzler
```

	Type					
price	Compact	Large	Midsize	Small	Sporty	Van
cheap	0	0	0	0	0	0
okay	0	0	0	0	0	2
expensive	0	0	0	0	0	0

, , mpg = okay

	Type					
price	Compact	Large	Midsize	Small	Sporty	Van
cheap	1	0	0	4	0	0
okay	5	3	6	0	6	6
expensive	4	8	14	0	4	1

, , mpg = miser

	Type					
price	Compact	Large	Midsize	Small	Sporty	Van
cheap	2	0	0	14	1	0
okay	4	0	2	3	3	0
expensive	0	0	0	0	0	0

По подобен начин може да се използва и функцията xtabs. При нея, обаче резултативната величина може да е само количествена. Да обърнем внимание, че на следващия ред цената

е с главна буква, т.е. та съдържа значенията на признака преди да ги превърнем в категорийни. Например

```
> xtabs(Price ~ Type)
Type
Compact   Large Midsize   Small  Sporty     Van
  291.4     267.3   598.8   213.5   271.5   171.9
```

При групировка при повече от два признака е много удобна функцията ***ftable*** от библиотеката ***stats***. По подразбиране първите вектори определят редовете, а само последния вектор определя колоните. Не е удачно да се използва с признаци с много възможни значения, като например непрекъснати количествени признаци, защото става прекалено голяма по обем. Да разгледаме данните от таблицата tires и да приложим тази функция към векторите, включени в нея.

```
> tires = read.csv(file = "C:\\Users\\User\\Desktop\\MoniStat\\tires.csv", header = TRUE,
+ sep = ";", dec = ",")
> ls(tires)
 [1] "N"    "X1"   "X10"  "X11_1" "X11_2" "X11_3" "X12_1" "X12_2" "X12_3" "X2"   "X3"
[12] "X4"   "X5"   "X6"   "X7"   "X8"   "X9"
> attach(tires)
> head(tires)
  N X1 X2 X3    X4       X5       X6     X7    X8 X9 X10 X11_1 X11_2 X11_3 X12_1 X12_2
X12_3
1 1 T1  P1  S1  93.98  51656.45 1196  Зимни 1л 14   1    T1     T2     T3     T4     T5     T6
2 2 T2  P2  S2  94.76  40885.81  760  Зимни 1л 15   2    T2     T4     T3     T5     T7     T6
3 3 T3  P3  S3 104.28  25534.63  732  Зимни 1л 16   3    T1     T5     T2     T4     T7     T6
4 4 T4  P4  S4 104.18  33402.56 1163  Зимни 1л 16   4    T1     T2     T3     T5     T7     T6
5 5 T5  P5  S5  98.50  28624.92  865  Зимни 1д 16   1    T2     T4     T3     T5     T7     T6
6 6 T1  P6  S1 111.22  44554.31  727  Зимни 1д 17   1    T1     T5     T2     T4     T7     T6

> ftable(X7, X1)
        X1    T1  T2  T3  T4  T5  T6  T7
X7
Зимни         12  12  12  14  15  29  28
Летни         10 12  11  12  13  34  36

> ftable(X7, X8, X1)
             X1    T1 T2 T3 T4 T5 T6 T7
X7      X8
Зимни 1д          2  4  4  3  4 10  8
        1л        2  2  1  2  0  3  2
        2д        4  3  3  3  4  7  8
        2л        4  3  4  6  7  9 10
Летни 1д          3  2  4  2  4 10 10
        1л        2  2  2  2  1  3  3
        2д        2  5  2  4  4 11 11
        2л        3  3  3  4  4 10 12

> ftable(X7, X2, X1)
                X1    T1 T2  T3 T4 T5 T6 T7
X7      X2
Зимни   P1          2  2  2  4  3  1  4
```

	P2	0	3	3	3	0	5	4
	P3	0	3	3	0	1	6	5
	P4	3	2	0	2	2	8	4
	P5	3	0	2	2	4	6	6
	P6	4	2	2	3	5	3	5
Летни	P1	1	1	2	1	3	5	7
	P2	3	3	1	2	3	3	6
	P3	1	2	2	3	1	7	5
	P4	1	2	3	1	2	5	7
	P5	2	2	1	2	3	5	5
	P6	2	2	2	3	1	9	6

> ftable(X7, X2, X8, X1)

			X1	T1	T2	T3	T4	T5	T6	T7
X7	X2	X8								
Зимни	P1	1д		0	1	1	0	0	0	2
		1л		2	0	0	1	0	1	0
		2д		0	0	0	2	0	0	1
		2л		0	1	1	1	3	0	1
	P2	1д		0	1	0	1	0	3	1
		1л		0	1	0	0	0	0	1
		2д		0	0	2	0	0	1	2
		2л		0	1	1	2	0	1	0
	P3	1д		0	1	1	0	0	3	0
		1л		0	0	1	0	0	1	0
		2д		0	2	0	0	1	0	1
		2л		0	0	1	0	0	2	4
	P4	1д		0	0	0	0	2	2	0
		1л		0	1	0	1	0	0	1
		2д		1	0	0	1	0	3	1
		2л		2	1	0	0	0	3	2
	P5	1д		0	0	0	2	1	2	2
		1л		0	0	0	0	0	1	0
		2д		1	0	1	0	0	2	3
		2л		2	0	1	0	3	1	1
	P6	1д		2	1	2	0	1	0	3
		1л		0	0	0	0	0	0	0
		2д		2	1	0	0	3	1	0
		2л		0	0	0	3	1	2	2
Летни	P1	1д		0	1	0	0	1	1	2
		1л		1	0	1	0	1	0	1
		2д		0	0	0	0	1	1	3
		2л		0	0	1	1	0	3	1
	P2	1д		1	0	0	0	0	2	2
		1л		1	1	0	1	0	0	0
		2д		0	1	0	1	1	1	1
		2л		1	1	1	0	2	0	3
	P3	1д		0	0	0	0	1	1	2
		1л		0	0	1	0	0	1	1
		2д		0	1	0	1	0	2	1
		2л		1	1	1	2	0	3	1

```
P4   1д    1 0 2 1 0 3 1
     1л    0 0 0 0 0 0 0
     2д    0 1 1 0 1 2 3
     2л    0 1 0 0 1 0 3
P5   1д    0 1 1 0 2 2 2
     1л    0 0 0 0 0 0 0
     2д    1 1 0 1 1 2 1
     2л    1 0 0 1 0 1 2
P6   1д    1 0 1 1 0 1 1
     1л    0 1 0 1 0 2 1
     2д    1 1 1 1 0 3 2
     2л    0 0 0 0 1 3 2
```

Може ние да изберем кои променливи да са в редовете и кои в колоните.

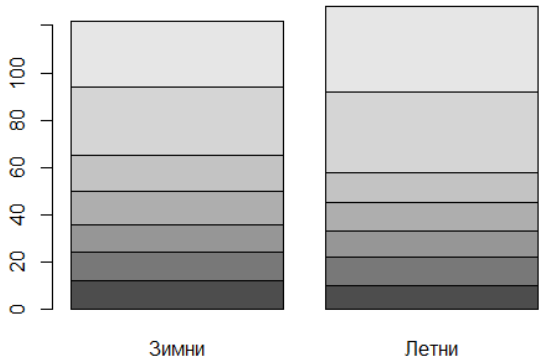
```
> ftable(X7, X2, X8, X1, row.vars = c("X7", "X2"), col.vars = c("X8", "X1"))
      X8          1д                    1л                    2д                    2л
      X1 T1 T2 T3 T4 T5 T6 T7  T1 T2 T3 T4 T5 T6 T7  T1 T2 T3 T4 T5 T6 T7  T1 T2 T3 T4 T5
T6 T7
X7    X2
Зимни P1   0  1  1  0  0  0  2   2  0  0  1  0  1  0     0  0  0  2  0  0  1    0  1  1  1  3  0  1
      P2   0  1  0  1  0  3  1   0  1  0  0  0  0  1     0  0  2  0  0  1  2    0  1  1  2  0  1  0
      P3   0  1  1  0  0  3  0   0  0  1  0  0  1  0     0  2  0  0  1  0  1    0  0  1  0  0  2  4
      P4   0  0  0  0  2  2  0   0  1  0  1  0  0  1     1  0  0  1  0  3  1    2  1  0  0  0  3  2
      P5   0  0  0  2  1  2  2   0  0  0  0  0  1  0     1  0  1  0  0  2  3    2  0  1  0  3  1  1
      P6   2  1  2  0  1  0  3   0  0  0  0  0  0  0     2  1  0  0  3  1  0    0  0  0  3  1  2  2
Летни P1   0  1  0  0  1  1  2   1  0  1  0  1  0  1     0  0  0  0  1  1  3    0  0  1  1  0  3  1
      P2   1  0  0  0  0  2  2   1  1  0  1  0  0  0     0  1  0  1  1  1  1    1  1  1  0  2  0  3
      P3   0  0  0  0  1  1  2   0  0  1  0  0  1  1     0  1  0  1  0  2  1    1  1  1  2  0  3  1
      P4   1  0  2  1  0  3  1   0  0  0  0  0  0  0     0  1  1  0  1  2  3    0  1  0  0  1  0  3
      P5   0  1  1  0  2  2  2   0  0  0  0  0  0  0     1  1  0  1  1  2  1    1  0  0  1  0  1  2
      P6   1  0  1  1  0  1  1   0  1  0  1  0  2  1     1  1  1  1  0  3  2    0  0  0  0  1  3  2
```

Изчертаване на barplots за многомерни данни

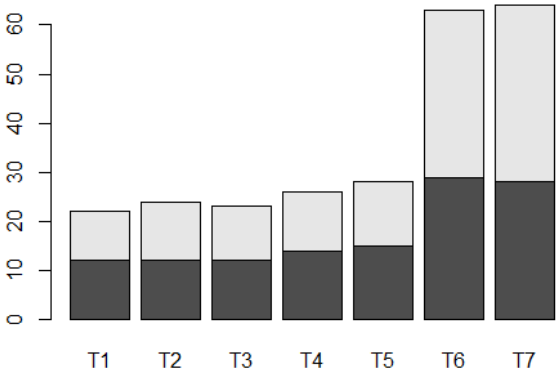
Да припомним, че за онагледяване на разпределението по категориен признак, една от най-често използваните графики е barplot. За да я използваме трябва първо да направим групировка с функцията table или с някоя подобна. barplot изчертава структурата на всяка колона като от таблицата като спазва съотношението, че отношението на лицето на частта, която представя подсъвкупността към лицето на целия правоъгълник трябва да е равно на пропорцията на единиците от подсъвкупността към броя на единиците в цялата съвкупност.

```
> table(X7, X1) # Според сезона X7 разделени, по видове гуми X1
                X1
  X7        T1 T2 T3 T4 T5 T6 T7
  Зимни     12 12 12 14 15 29 28
  Летни     10 12 11 12 13 34 36
```

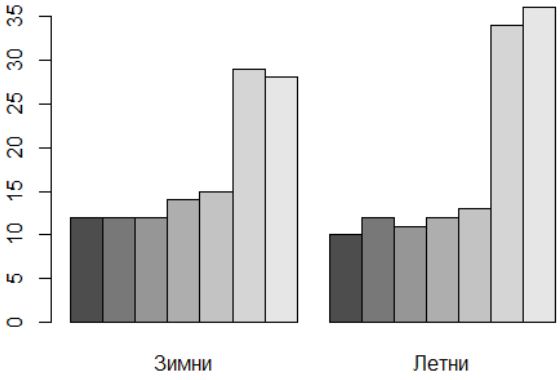
```
>  barplot(table(X1, X7))
```



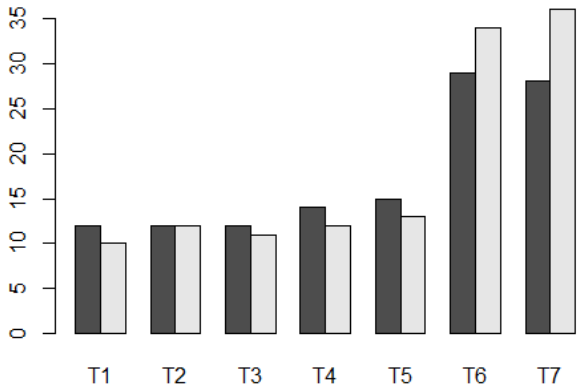
> barplot(table(X7, X1)) # Според видовете гуми X1 разделени, по сезони X7 (зимни, летни).



Ако искаме правоъгълниците да са един до друг използваме аргумента beside = TRUE.
> barplot(table(X1, X7), beside = TRUE)



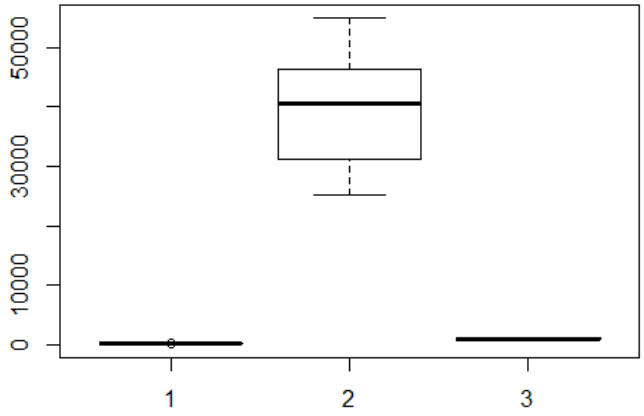
> barplot(table(X7, X1), beside = TRUE)



Изчертаване на boxplots за многомерни данни

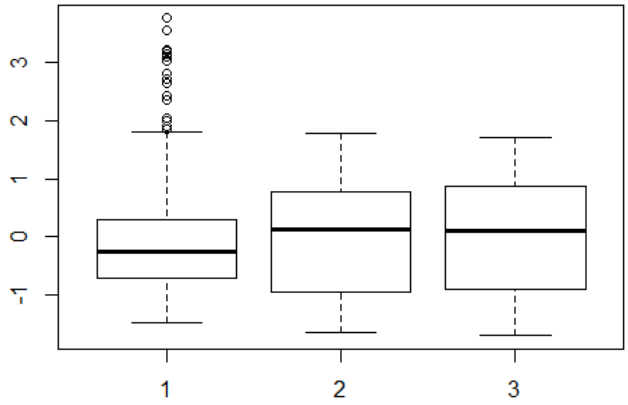
Командата ***boxplot(x,y,z)*** е друг удобен начин за онагледяване на многомерни данни. Тя се използва при метрирани признаци.

```
> boxplot(X4, X5, X6)
```



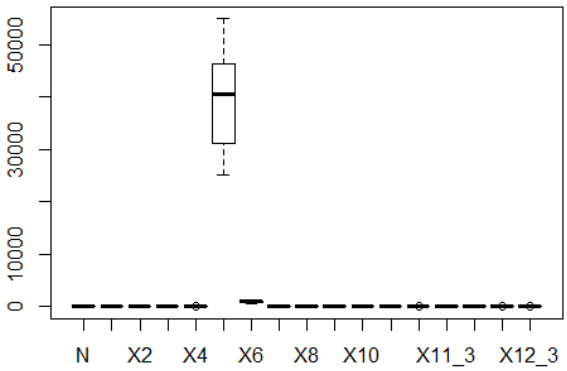
Изчертава всички графики с мустачки на една и съща скала. Ако данните не са предварително центрирани и нормирани картината не е достатъчно отчетлива. По тази причина, преди да бъде използвана е добре данните да се центрират с тяхната средна и да се нормират със средно-квадратичното си отклонение. Т.е. добре е преди да изчертаем много графики с мустачки на една и съща скала да трансформираме данните и за преминем към работа със z score.

```
> z4 = (X4-mean(X4))/sd(X4)
> z5 = (X5-mean(X5))/sd(X5)
> z6 = (X6-mean(X6))/sd(X6)
> boxplot(z4, z5, z6)
```



По аналогичен начин може да се изчертаят boxplot за всички количествени признаци от цял data frame. Например

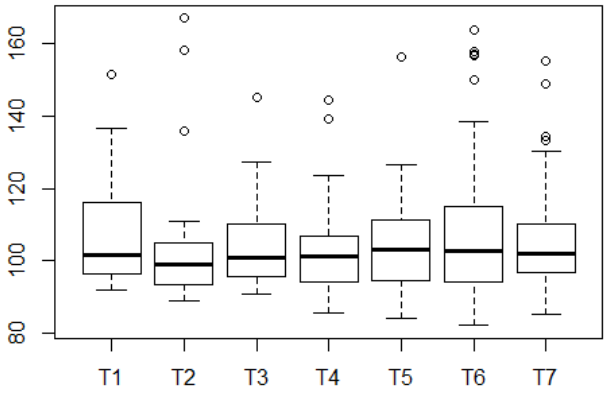
```
> boxplot(tires)
```



Тук, обаче отново важи правилото, че за да бъде картинката добра, те трябва да са предварително центрирани и нормирани.

Най-честата употреба на boxplot е в съчетание с израз за формула, т.е. изчертаване на boxplot на даден метриран признак (по-долу това е X4) по подгрупи по някой неметриран (по-долу това е X1). Различните групи(нива) на неметрирания признак оформят подгрупите, а графиките с мустачки се изчертават за метрираната величина, според измерените значения на този признак, при единиците, които са попаднали в подгрупите, определени по неметрирания. Построяването на тези графики една до друга е полезно най-вече при сравняване на подобни разпределения особено, когато данните във всяка група са много.

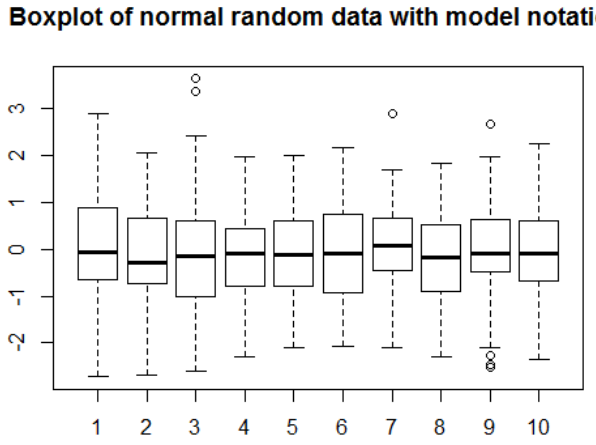
```
> boxplot(X4 ~ X1)
```



Пример: Симулирайте данни от 1000 наблюдения върху стандартно нормално разпределена случайна величина (да я означим с X) и постройте друга фактор променлива, с 10 възможни значения, всяко от които се повтаря по 100 пъти (да я означим с Y). Т.е. X е метриран признак, а Y е факторна променлива със същата дължина както и X. След това, като използвате записа за формула и функцията boxplot изчертайте в един графичен прозорец, поотделно графики с мустачки на X поотделно в групите, обособени, при групиране по Y.

Решение:

```
> y = rnorm(1000)            # симулира 1000 независими реализации на стандартно нормално
                               # разпределена случайна величина
> f = factor(rep(1:10,100))  # числата 1, 2, ..., 10, повторени по 100 пъти
> boxplot(y ~ f , main = "Boxplot of normal random data with model notation")
```

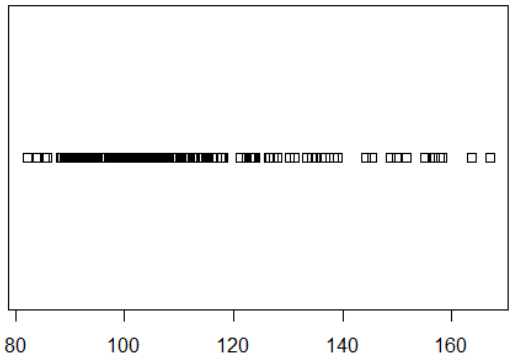


Изчертаване на stripcharts за многомерни данни

Съчетанието между онагледяването на самите данни и простотата на графиката, като че ли са постигнати най-вече при така наречените stripcharts. Те са подходящи особено, когато данните не са прекалено много на брой. Те изчертават подредените действителни данни по начин, подобен на ***rug***, който се използва при хистограмите. Те, подобно на ***boxplot***, могат да бъдат използвани с:

- непосредствено посочване на метрираните признаци, за които ще се построяват, но в този случай те се обединяват и за целта трябва да бъде зададен ***method = "stack"***. Например в таблицата tires

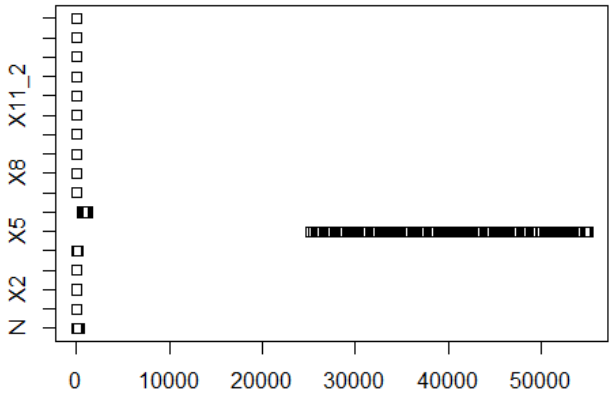
```
> stripchart(X4, X5, X6, method = "stack")
```



Това не е най-добрият начин, т.к. обединява данните. Поради това почти не се използва.

- Чрез посочване на data frame и построяването им за всички признаци от тази графика. Използва се само, когато целия data frame е от метрирани признаци, но дори в този случай данните трябва да се центрират и нормират за да се получи по-хубава графика. Без центиране и нормиране, и при наличие на качествени признаци, графиката изглежда така.

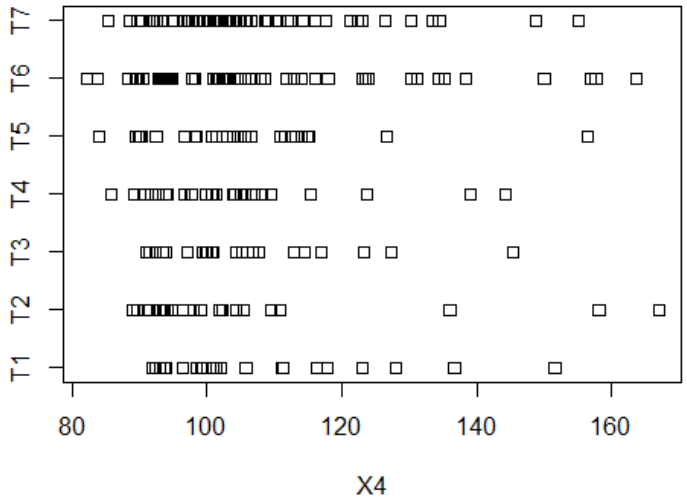
> stripchart(tires)



Т.е. в нашия случай тя не е удобна за работа.

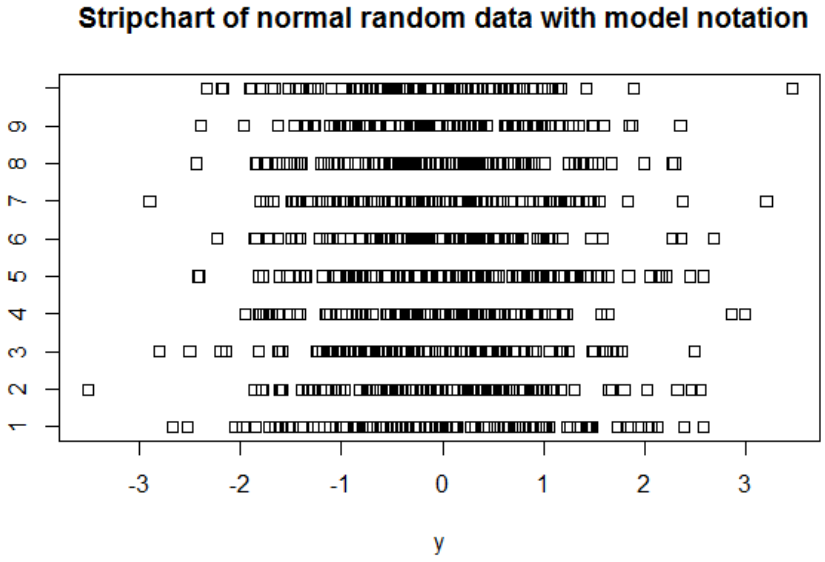
- Най-добрият начин за използване на тези графики е чрез задаване на формула, която да посочи кой метриран признак по кой неметриран да го разделим в подгрупи, за които да изчертаем stripcharts. Например в таблицата tires

> stripchart (X4 ~ X1)



Пример: Симулирайте данни от 1000 наблюдения върху стандартно нормално разпределена случайна величина (да я означим с X) и постройте друга фактор променлива, с 10 възможни значения, всяко от които се повтаря по 100 пъти (да я означим с Y). Т.е. X е метриран признак, а Y е факторна(качествена) променлива със същата дължина както и X. След това, като използвате записа за формула и функцията boxplot изчертайте в един графичен прозорец, поотделно графики с мустачки на X поотделно в групите, обособени, при групиране по Y.
Решение:

```
> y = rnorm(1000)
> f = factor(rep(1:10,100))
> stripchart (y ~ f , main = " Stripchart of normal random data with model notation")
```



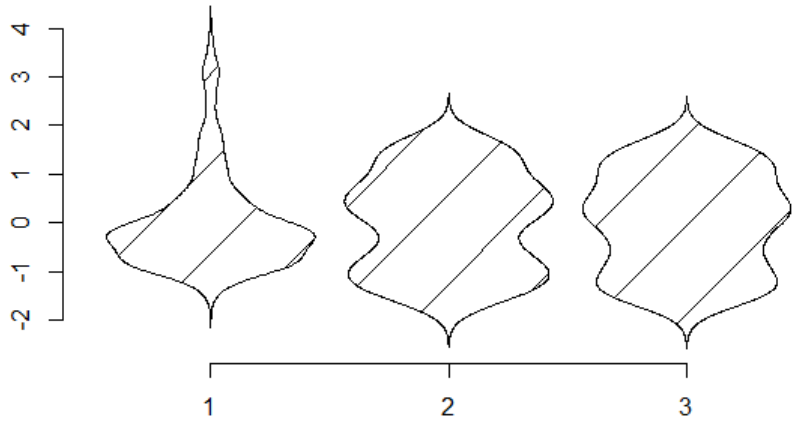
Изчертаване на violinplots за многомерни данни

Функцията ***simple.violinplot*** от библиотеката ***UsingR*** може да бъде използвана на мястото на ***boxplot*** за сраняване на различни разпределения. Тя изчертава емпиричната плътност заедно с неин огледален образ, за подсилване на визуалния ефект.

- Може да се използва с непосредствено задаване на вектори и изчертава поотделно техните ***simple.violinplot*** на една скала. За да се получи хубава графика е необходимо данните първо да бъдат центрирани и нормирани или поне да са с близки значения.

Например в таблицата tires разпределенията на цените - X4, пробега - X5 и продължителността на живот - X6, центрирани и нормирани могат да бъдат сравнени с

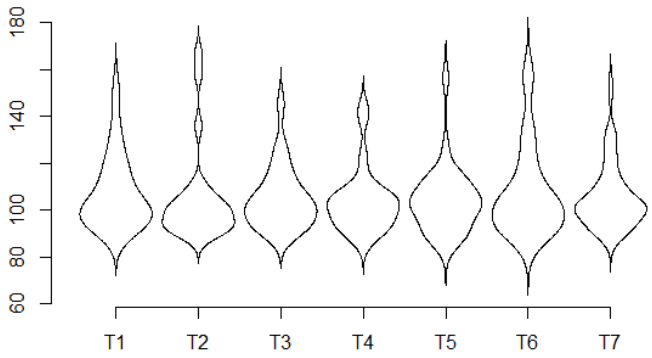
```
> library(UsingR)
> z4 = (X4-mean(X4))/sd(X4)
> z5 = (X5-mean(X5))/sd(X5)
> z6 = (X6-mean(X6))/sd(X6)
> simple.violinplot (z4, z5, z6)
```



Наблюдаваме, че разпределението на цените е най-асиметрично и концентрирано около малките стойности. Разпределението на пробега е бимодално, т.е. в извадката ни са попаднали гуми със сравнително малък и сравнително голям пробег и нямаме концентриране около средното. Разпределението на гумите според продължителността на живот е сравнително равномерно в наблюдаваните подинтервали.

- Най-добрият начин за използване на тези графики е чрез задаване на формула, която да посочи кой метриран признак по кой неметриран да го разделим в подгрупи, за които да изчертаем ***simple.violinplot***. Например в таблицата tires разпределенията на цените според видовете на гумите може да бъде сравнено с

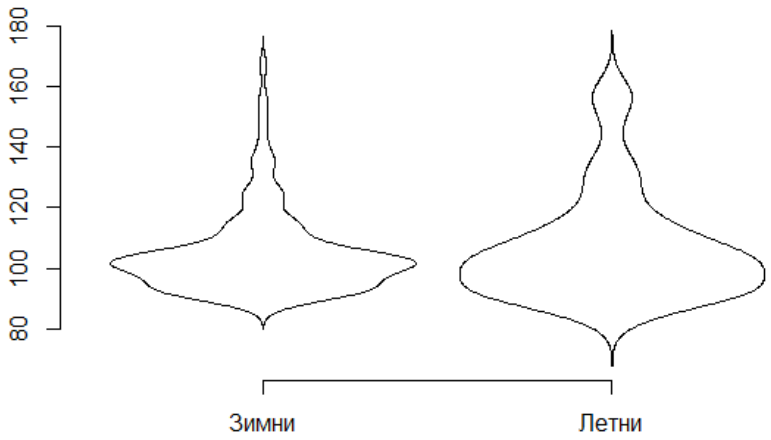
```
> simple.violinplot(X4 ~ X1)
```



Виждаме, че във всички подгрупи разпределението е почти едно и също. Това би трябвало да ни наведе на мисълта, че според данните от извадката, видовете(марката) на гумите не оказва влияние на тяхната цена.

Сега да видим дали сезонът оказва влияние на тяхната цена.

```
> simple.violinplot(X4 ~ X7)
```



Цените на зимните гуми са по-концентрирани около тяхната средна, докато при летните гуми се наблюдава по-голямо разнообразие.

Изчертаване на simple.densityplot за многомерни данни

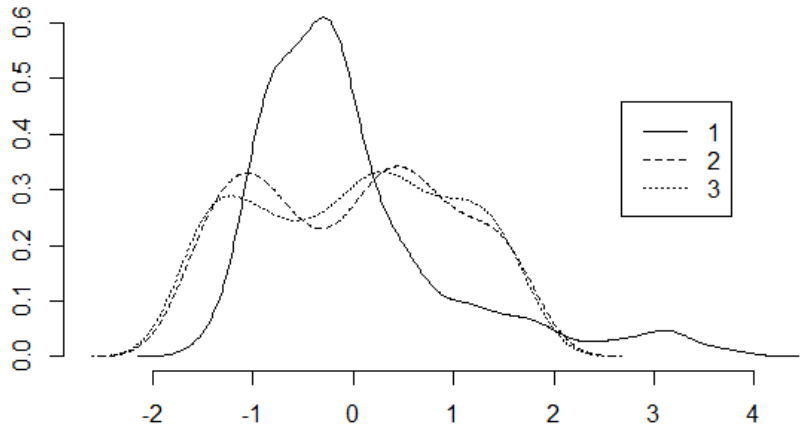
Функцията ***simple.densityplot*** от библиотеката UsingR може да бъде използвана на мястото на ***boxplot*** за сравняване на различни разпределения. Тя изчертава емпиричната плътност на раз пределение. Формата на резултата зависи съществено от ширината на избраните подинтервал и. Тя се задава в параметъра ***bw***, чието съкращение идва от bandwidth(честотна лента). Колкот о ширината на подинтервалите е по-голяма, толкова приближаващата плътността крива е по-гладка. Ако искаме да увеличим назъбеността ѝ, намаляваме параметъра bw.

Самият алгоритъм за построяването на гладката крива, понякога е доста сложен. Плътността може да бъде разглеждана като гладък и непрекъснат аналог на хистограмата. Т.к. многомерната хистограма би изглеждала претрупана и поради тази причина ужасна.

Същата гладка крива може да бъде получена с помощта на функциите ***lines*** и ***density***, като преди това трябва да сме използвали функцията ***hist***.

- Тази функция изчертава на една скала графиките на плътността на векторите, зададени като параметри. По тази причина е удачно първо да центрираме и нормираме величините си. Например в таблицата tires поотделно емпиричните плътности на разпределение на цените - X4, пробега - X5 и продължителността на живот - X6, центрирани и нормирани могат да бъдат видени с

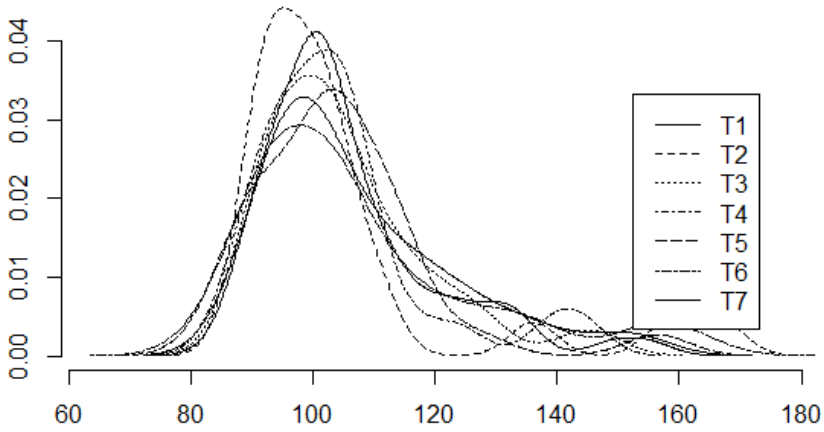
```
> z4 = (X4-mean(X4))/sd(X4)
> z5 = (X5-mean(X5))/sd(X5)
> z6 = (X6-mean(X6))/sd(X6)
> simple.densityplot(z4, z5, z6)
```



Забелязва се, че центрирани и нормирани пробегът и продължителността на живот имат почти едно и също разпределение.

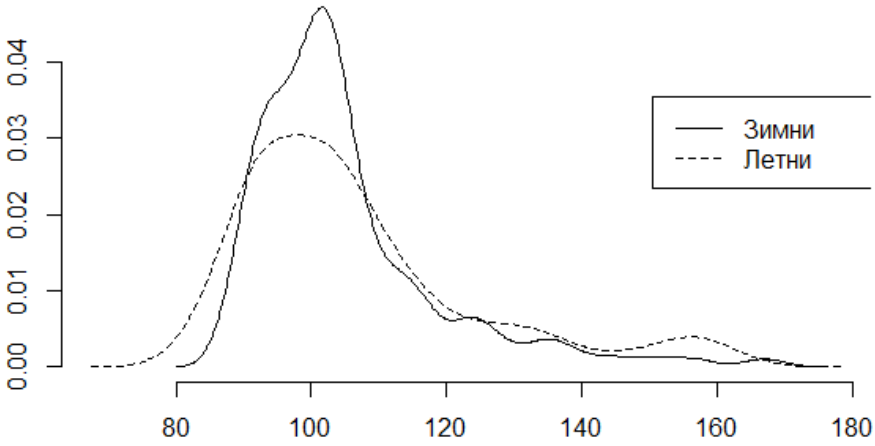
- Друг добър начин за използване на тези графики е чрез задаване на формула, която да посочи кой метриран признак по кой неметриран да го разделим в подгрупи, за които да изчертаем simple.densityplot. Например в таблицата tires емпиричните плътности на разпределенията на цените според видовете на гумите могат да бъдат сравнени с

> simple.densityplot (X4 ~ X1)



Забелязва се, че емпиричните плътности си приличат. Т.е. това може да е сигнал, че вида на гумите не влияе на цената им.
За да видим дали сезонът влияе на цената можем да използваме

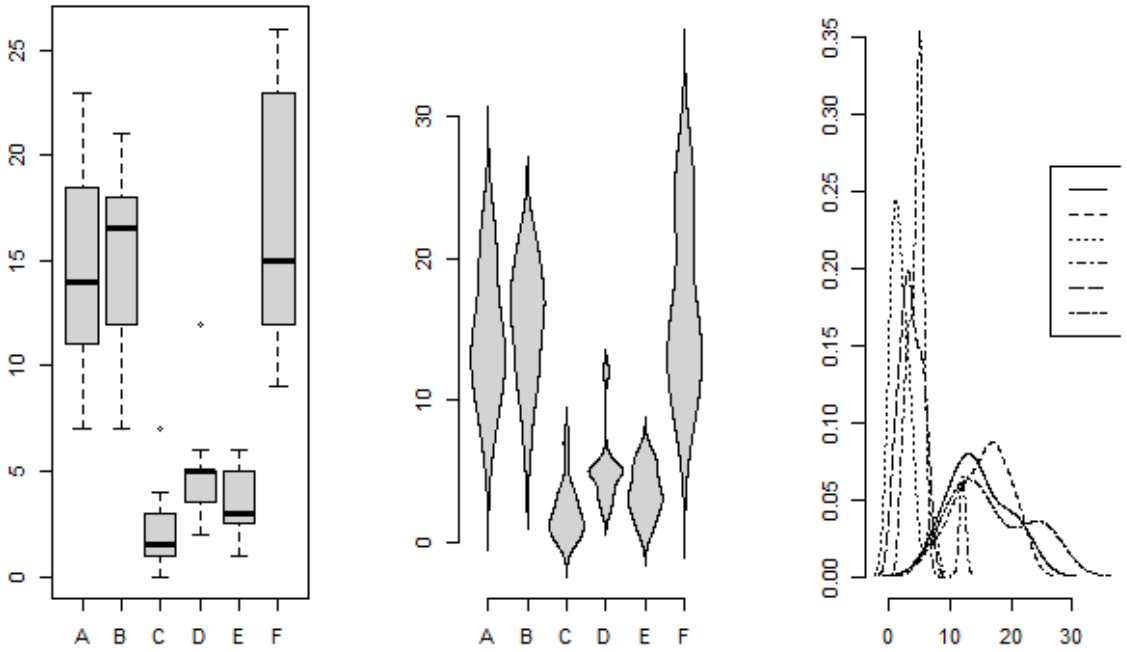
> simple.densityplot (X4 ~ X7)



Тези разпределения също си приличат, само цените на зимните гуми са по-концентрирани около тяхната средна, за това при това разпределение се наблюдава по-голям ексцес.

Пример: В данните InsectSprays сравнете броят на унищожените вредители при използването на 6-те различни спрея за едно и също време. Използвайте ***boxplot***, ***simple.violinplot*** и ***simple.densityplot***.

```
> par(mfrow = c(1,3))  # 3 graphs per page
> data(InsectSprays)  # load in the data
> boxplot(count ~ spray, data = InsectSprays, col = "lightgray")
> simple.violinplot(count ~ spray, data = InsectSprays, col = "lightgray")
> simple.densityplot(count ~ spray, data = InsectSprays)
```



Наблюдава се, че според броя на унищожените от тях насекоми, спрейовете могат да бъдат разделени в две групи. В едната група са ефективните спрейовете A, B и F в другата група са останалите.

Изчертаване на scatterplots за многомерни данни

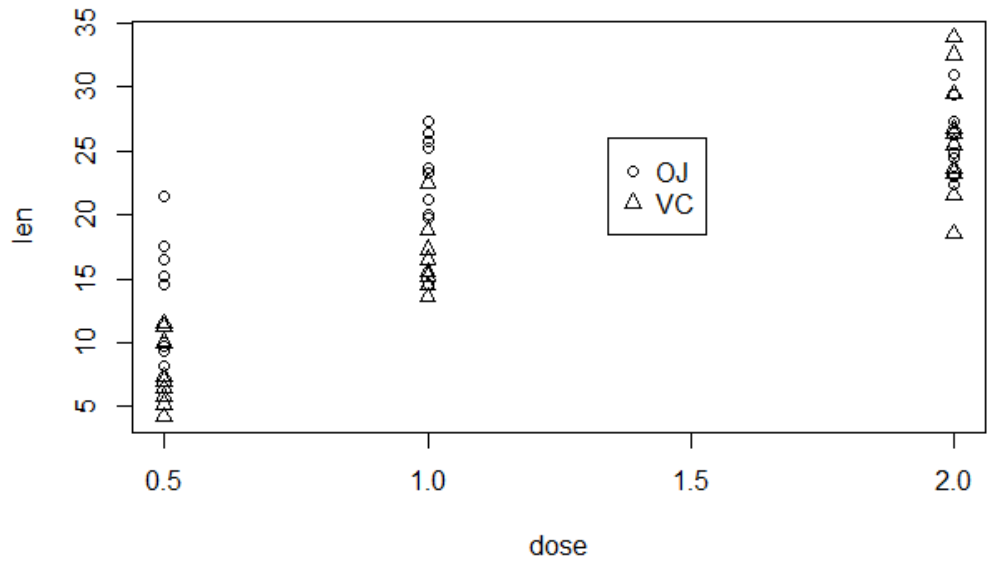
Да предположим, че имаме една независима променлива X и две зависими променливи Y и Z. Ако искате да изчертаете двете корелационни полета на една и съща координатна система можете да го направите с настройване на символа, с който да се изобразят точките от съответното корелационно поле. Това става с помощта на функциите ***plot*** и ***points***, като във втората използвате параметъра ***pch*** (plot character).

```
> plot(X, Y) # изчертава корелационното поле на X и Y с точки (scatterplot)
> points(X, Z, pch="2") # изчертава точките от корелационното поле на X и Z като триъгълници.
```

Да отбележим, че винаги първата функция е ***plot***, а втората ***points*** за да използваме същия графичен прозорец.

Ако искаме да изчертаем две корелационни полета на една и съща графика на два количествени признака, (независимата променлива да има малко значения) но с различни символи по подгрупи, можем да използваме функцията ***plot*** символа за формула ~ между метрираните признаци и параметъра ***pch = as.numeric***(признакът, който оформя подгрупите). Да обърнем внимание, че ако независимата променлива е категорийна няма да получим корелационни полета, а ще получим графики с мустачки, макар и пак да употребим функцията ***plot***. Например по данните от ToothGrowth, ако искаме да разгледаме зависимостта на дължината на зъбите (len) при прасета от дозата витамин C (dose), която те приемат и разделени в две групи (supp) според вида на витамина, можем да използваме

```
> data("ToothGrowth")
> attach(ToothGrowth)
> plot(len ~ dose, pch = as.numeric(supp))
# click mouse to add legend.
> tmp = levels(supp) # store for a second
> legend(locator(1), legend = tmp, pch = 1:length(tmp))
> detach(ToothGrowth)
```



От графиката се вижда, че за всички дози витамин С от вида (VC) е по-малко ефективен.

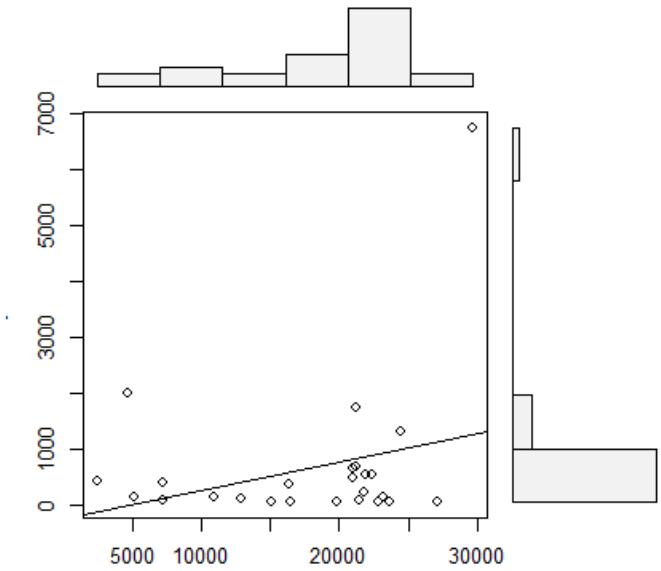
Ако имаме само два количествени признака и ако независимата променлива е случайна и има много възможни значения, е по-добре да изчертаем до корелационното поле, хистограмите на наблюдаваните признаци. Това може да стане с помощта на функцията

simple.scatterplot.

Да отбележим, че в предния пример това не е удачно защото независимата променлива не е случайна.

Например: Въпросът за отделянето на въглероден двуокис CO2 в наши дни е гореща тема във връзка с влиянието му върху парниковия ефект. Данните emissions съдържат Брутният вътрешен продукт за цялата страна (Gross Domestic Product GDP), Брутния вътрешен продукт на глава от населението (perCapita) и количеството на въглероден двуокис (CO2) в няколко европейски страни и САЩ през 1999 г. Изчертайте корелационно поле на данните, като около осите сложите хистограмите на разпределение на двата признака.

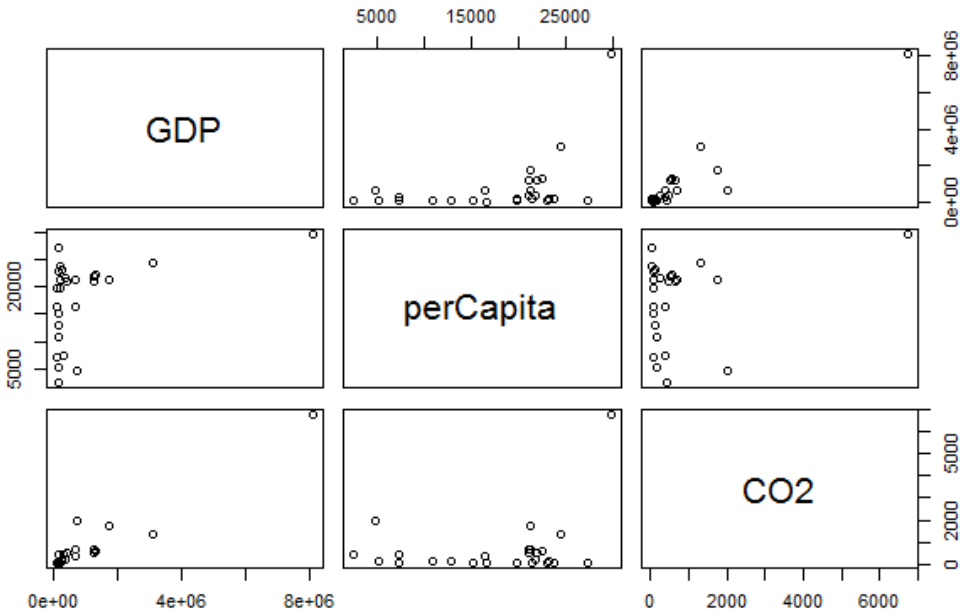
```
> library(UsingR)
> data(emissions) # or read in from dataset
> attach(emissions)
> ls(emissions)
[1] "CO2"       "GDP"       "perCapita"
> head(emissions)
                    GDP    perCapita   CO2
UnitedStates       8083000    29647    6750
Japan               3080000     24409     1320
Germany             1740000     21197     1740
France              1320000     22381      550
UnitedKingdom    1242000     21010      675
Italy               1240000     21856      540
> simple.scatterplot(perCapita,CO2)
> title("GDP/capita vs. CO2 emissions 1999")
> detach(emissions)
```



От тази графика, освен зависимостта от корелационното поле се забелязва, че разпределението на страните според Брутния им вътрешен продукт на глава от населението е много по-равномерно отколкото разпределнието на страните според въглеродните емисии CO2, при които от своя страна се наблюдава и силно отличаващо се наблюдение (outlier).

Ако искаме да изчертаем матрица от корелационни полета на няколко метрирани признака, това става с функцията ***pairs***. По диагонала се появяват имената на променливите, които съответстват на осите на съседните корелационни полета. Например

```
> pairs(emissions)
```



Най-лесно се построява такава графика, когато параметърът с променливите е подаден като data frame както е по-горе. Можем да проверим, че данните са от data frame с

```
> class(emissions)
[1] "data.frame"
```

Ако данните не са в data frame можем да построим data frame с помощта на функцията ***cbind***. *Например*:

pairs(cbind(x, y, z))

Функцията pairs има много опции, които да разкрасят графиката. Пакетът Ggobi (http://www.ggobi.org) позволява да се подчертават данни на някоя графика или да се изтриват.

Задача за самостоятелна работа. В таблицата emissions намерете силно отличаващото се наблюдение outlier за отделените въглеводородни емисии CO2. Използвайте функцията ***identify*** за да определите реда, който съответства на аутлайъра. Махнете този аутлайър и повторете горните графики без него.

Упътване:

```
> library(UsingR)
> data(emissions)
> attach(emissions)
```

По настоящем R има няколко начина за взаимодействия с графиката от корелационното поле. Някои от тях ни позволяват да локализираме точно коя точка, за кое наблюдение се отнася. Функцията ***identify*** намира индексите (номера на реда в таблицата) на най-близката точка до координатите на точката, в която сме кликнали с мишката. Ако искаме да направим това повече от веднъж задаваме на параметъра n, броят на кликанията ни.

```
> Myresult = simple.lm(CO2, GDP)
> identify(CO2, GDP, n = 1)
[1] 1
> new = emissions[-1, ] # изтриваме реда. Да обърнем внимание, че тук запетайката е важна,
                        # иначе ще изтрием първата колона.
```

Пакетът ***lattice***

В пакетът ***lattice*** са внедрени много графични концепции на Bill Cleveland. Част от него е пакетът ***grid***, който позволява по различен начин да бъдат онагледени многомерни данни. Тези пакети не са част от базовата версия на R, но от версия 1.5.0 те се препоръчват. Те имат много възможности, но ние ще илюстрираме само някои от тях. Нека разгледаме данните ***Cars93*** от пакета MASS, които съдържат 93 реда и 27 колони.

```
> ls(Cars93)
 [1] "AirBags"           "Cylinders"          "DriveTrain"
 [4] "EngineSize"        "Fuel.tank.capacity" "Horsepower"
 [7] "Length"            "Luggage.room"       "Make"
[10] "Man.trans.avail"   "Manufacturer"       "Max.Price"
[13] "Min.Price"         "Model"              "MPG.city"
[16] "MPG.highway"       "Origin"             "Passengers"
[19] "Price"             "Rear.seat.room"     "Rev.per.mile"
[22] "RPM"               "Turn.circle"        "Type"
[25] "Weight"            "Wheelbase"          "Width"
```

Приемаме, че те са заредени, но не е използвана функцията ***attach*** за да можем да илюстрираме параметъра ***data*** по-долу.

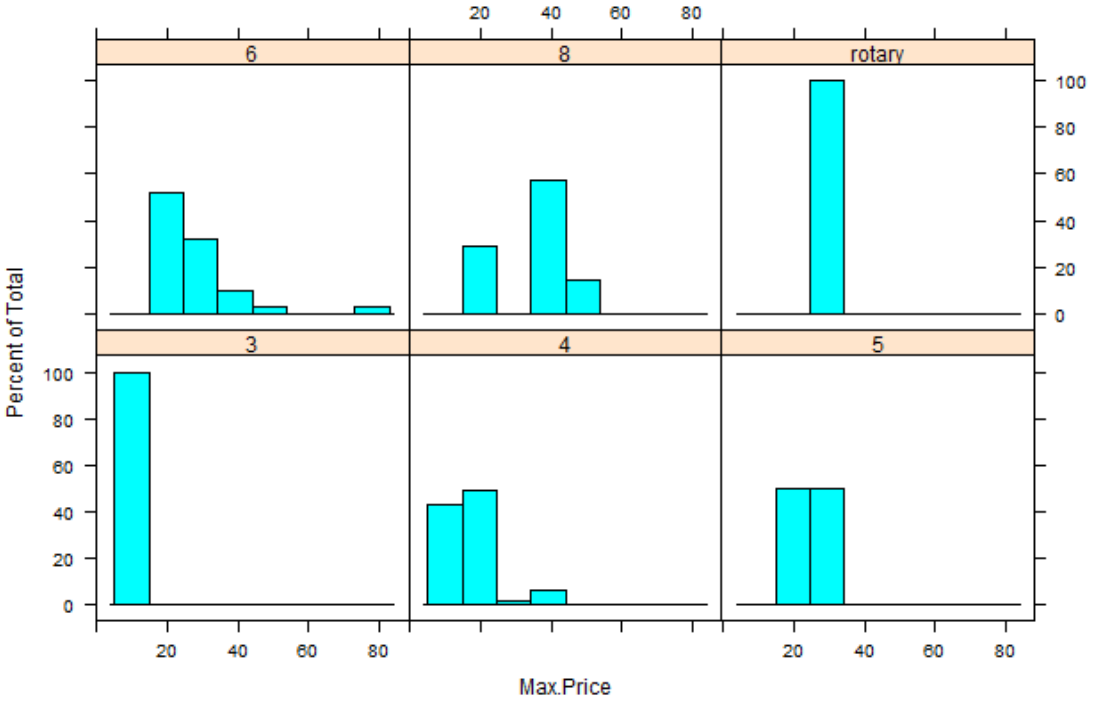
Ще използваме конструкцията на формула

Празно или зависима променлива, която обикновено е метриран признак ~ метриран признак | качествен признак или признак с малко възможни значения

Основната идея е, според различните значения на признака в условието, в отделни правоъгълници, да бъдат построени еднотипни графики за метрирания признак. За едномерни графики лявата страна в горната формула е празна. Имената на функциите са естествени, но често са различни от досега срещаните.

Например, в следващия ред, функцията ***hist*** е заменена с ***histogram*** от пакета (lattice).

```
> library(lattice)
> histogram( ~ Max.Price | Cylinders , data = Cars93)
```

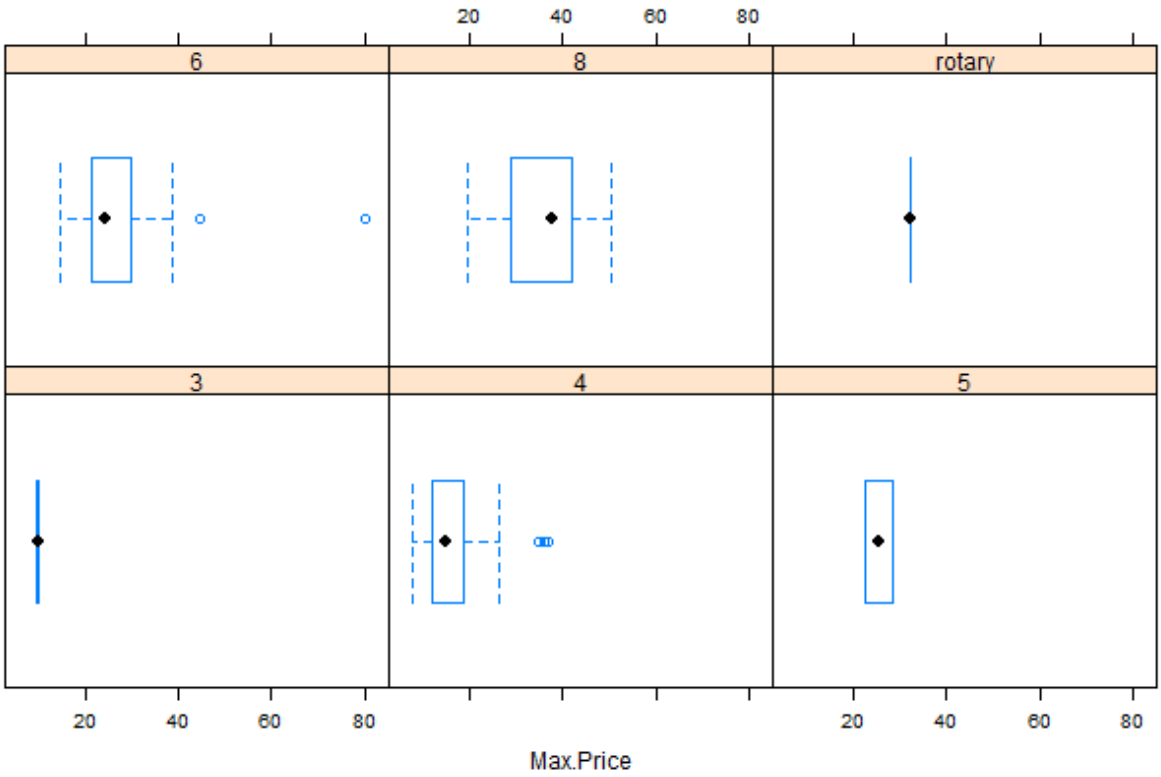


По-горе максималните цени са групирани според броевете на цилиндрите и според тази групировка са построени хистограми на разпределението на максималните цени в подгрупите. Да отбележим, че мястото на зависимата променлива е оставено празно.

Графиките с мустачки също са едномерни. За да се изчертаят много графики на веднъж с пакета ***lattice*** вместо командата ***boxplots***, се използва командата ***bwplot***.

Например. Нека отново отделим в отделни групи максималните цени, според броевете на цилиндрите. Да построим графиките с мустачки на разпределенията на максималните цени в подгрупите. Ще оставим отново мястото на зависимата променлива празно.

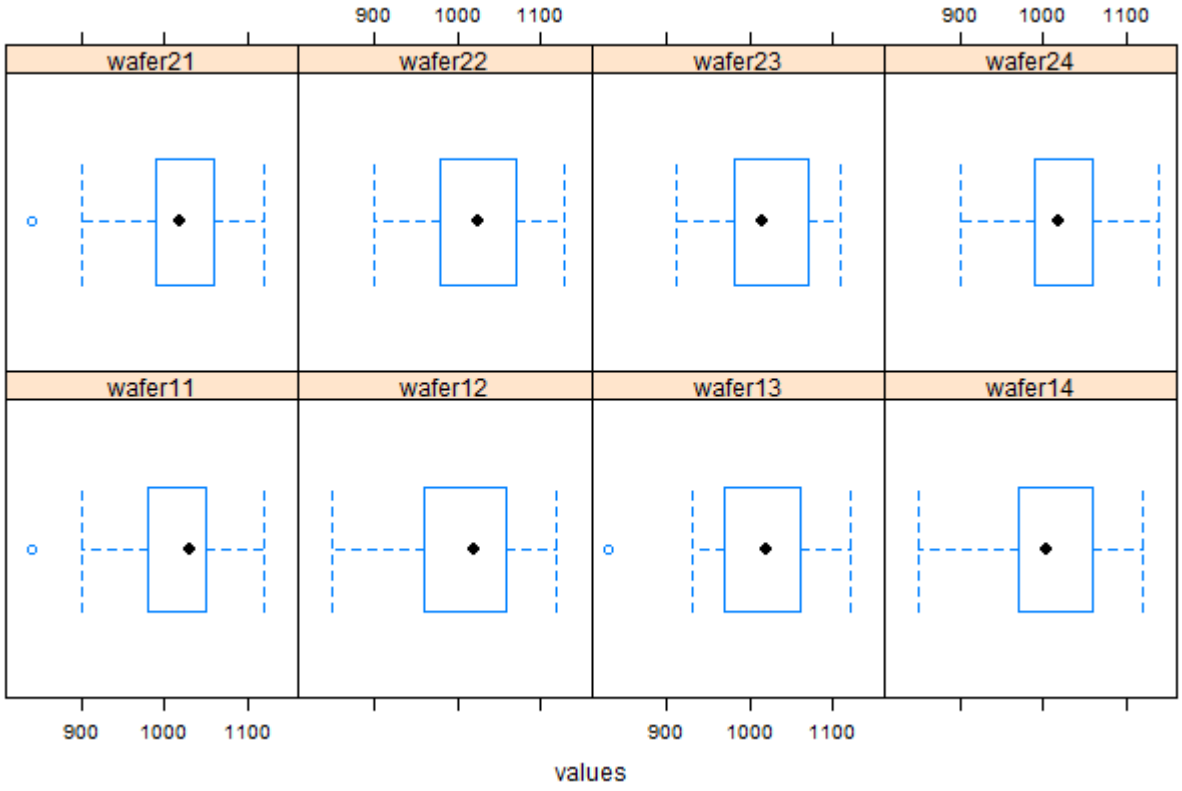
```
> bwplot( ~ Max.Price | Cylinders , data = Cars93)
```



Задача: В таблицата chips се съдържат данни за дебелината на интегрирани чипове. Разгледани са два вида чипове и всеки вид е мерен на 4 места. Направете панел, с 8 графики на кутии с мустачки за дебелината на чипа на всяко място на измерване и от всеки вид.

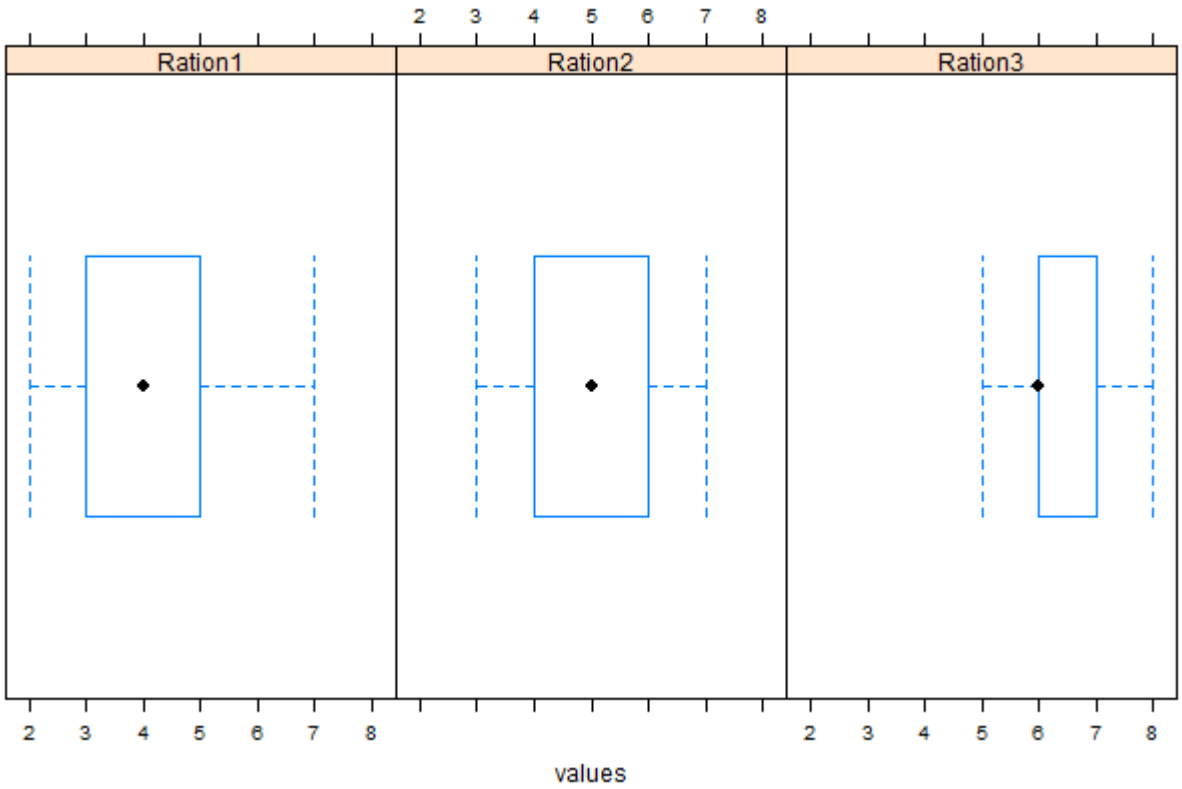
```
> data(chips)
> ls(chips)
[1] "wafer11" "wafer12" "wafer13" "wafer14" "wafer21" "wafer22" "wafer23" "wafer24"
> head(chips)
  wafer11 wafer12 wafer13 wafer14 wafer21 wafer22 wafer23 wafer24
1     950     930     950     930    1010     980     970     980
2    1050    1050    1030    1040    1050    1050    1050    1060
……….
> t1=stack(chips)  # обединяваме метрираните и правим неметриран признак, по значенията на
                   # който ще оформим подгрупите
> t1
   values     ind
1     950 wafer11
2    1050 wafer11
3     940 wafer11
```

……
> bwplot(~ values | ind , data = t1)



Задача: В таблицата chicken се съдържат данни за теглото на пилетата при три различни начина на хранене. Направете панел с 3 boxplot за всяка от трите диети. Има ли разлика в средните тегла?

```
> data(chicken)
> ls(chicken)
[1] "Ration1" "Ration2" "Ration3"
> head(chicken)
  Ration1 Ration2 Ration3
1       4       3       6
2       4       4       7
> t1 = stack(chicken)  # обединяваме метрираните и правим неметриран признак, по значенията
                       # на който ще оформим подгрупите.
> t1
   values    ind
1       4 Ration1
2       4 Ration1
3       7 Ration1
4       3 Ration1
……
> bwplot( ~ values | ind , data = t1)
```

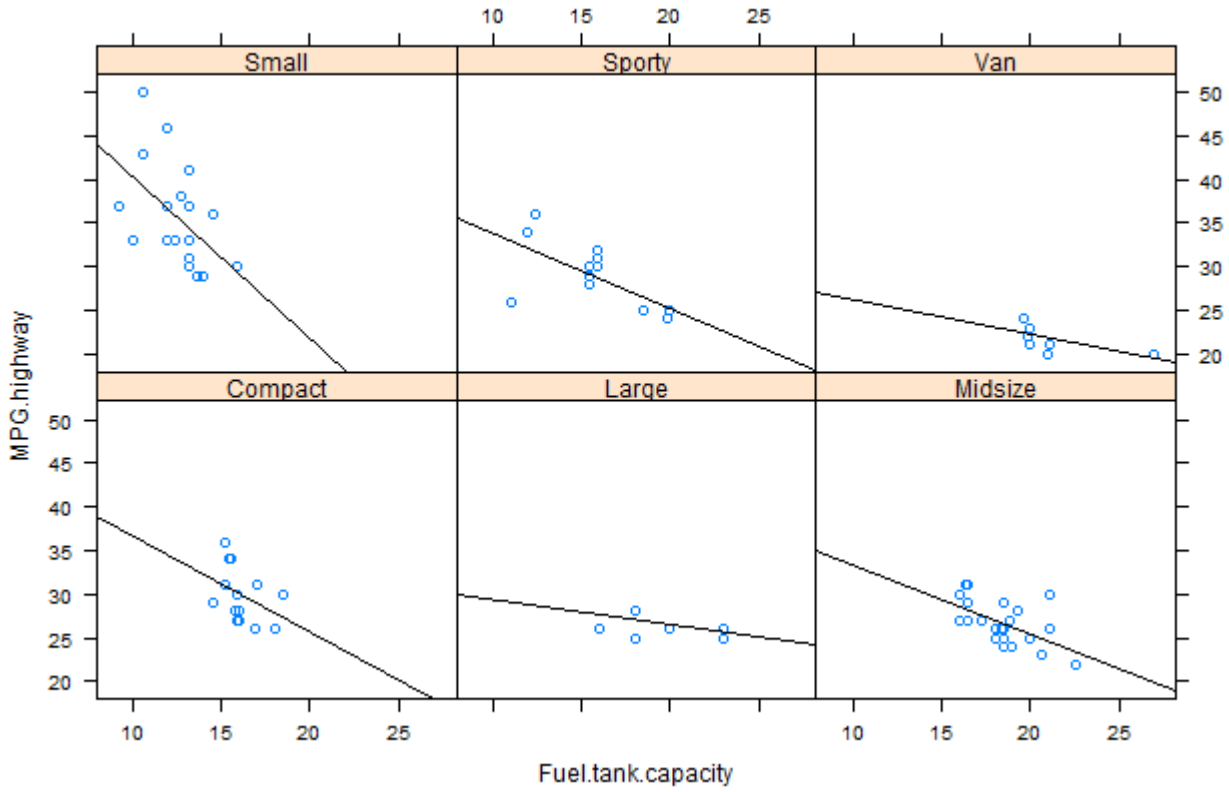


Последният начин на хранене, очевидно дава най-добър резултат.

Сравняването на корелационните полета и линиите на регресия на зависимостта между един зависим и един независим признак в подгрупите, образувани по еднаквите значения на величината от условието, могат да бъдат много информативни относно влиянието на величината в условието върху зависимостта между независимата и зависимата променливи. За да се изчертаят много корелационни полета на веднъж и поотделно за подгрупите, образувани след групировката по величината в условието с пакета ***lattice*** вместо командата ***plot***, се използва командата ***xyplot***.

Например. Нека отново отделим в отделни групи колите според техния вид: малка, средна, компактна и т.н. Във всяка от получените групи да моделираме зависимостта на консумацията на гориво на магистрала, от капацитета на резервоара. Да построим корелационните полета в подгрупите и правите на регресия. За целта ще напишем функция plot.regression, която да добавя линиите на регресия върху корелационните полета. Да обърнем внимание, че и чертането на корелационното поле и при добавянето на линията на регресия се използва все функцията ***xyplot***, но във втория случай имаме допълнителен параметър ***panel =*** след който се пише името на функцията.

```
> attach(Cars93) # don't need data = Cars93 now
> xyplot(MPG.highway ~ Fuel.tank.capacity | Type)
## plot with a regression line
## first define a regression line drawing function
> plot.regression = function(x,y) {
                                    + panel.xyplot(x,y)
                                    + panel.abline(lm(y~x))
                                    + }
> trellis.device(bg="white") # set background to white.
> xyplot(MPG.highway ~ Fuel.tank.capacity | Type, panel = plot.regression)
```



Сега можем да анализираме зависимостите на по-горната графика. С нарастването на колите, наклонът на линията на регресия намалява. Т.е. според данните от извадката, с нарастването на обемът на резервоара, разходът на гориво намалява в рамките на една група. Функцията ***trellis.device*** е използвана за да направи фона на графиките бял. По подразбиране той е сив. Този пакет има още много възможности за разкрасяване на графиките.