

# Программирование и статистический анализ данных на языке R

Лекция 1 (Основы программирования на языке R)



Петровский Михаил (ВМК МГУ), [michael@cs.msu.su](mailto:michael@cs.msu.su)

# История и особенности языка R

- R бесплатный диалект и наследник **S** (Bell Labs 1980e)
- Альма-матер - Оклендский университет, Ross Ihaka и Robert Gentleman (1990e)
- Основное назначение – **статистическое программирование** (в меньшей степени ML и DM, т.к. не очень развиты средства обработки данных)
- Лицензия **GNU GPL**, мультиплатформенность
- **Интерпретируемый регистрозависимый** язык командной строки или интерактивной GUI среды (Rstudio, Jupyter блокноты) или внутри «внешних» сред Weka, RapidMiner, Oracle, Microsoft и др.
- **Объектный** подход, простые примитивы и структуры данных, простые управляющие конструкции, пользовательские функции
- **In-memory** – по умолчанию все хранится в памяти 😊
- Расширяемость за счет внешних **пакетов**
- Мощная подсистема графики и **визуализации**
- Установка <http://www.r-project.org>

# Интерфейс R Studio

**Меню консоли**

**Консоль**

**Графический вывод (с «листалкой»)**

The screenshot displays the RStudio interface. At the top is the menu bar with options like File, Edit, View, Misc, Packages, Windows, and Help. Below the menu is a toolbar with various icons. The main area contains three panes: 1) The R Console pane shows the R startup message and a command > demo(graphics). 2) The Plot pane shows a scatter plot titled "Simple Use of Color In a Plot" with green points connected by a dashed line. 3) The History pane on the right shows a list of commands, with the last two highlighted in blue: > y <- c(0, cumsum(rnorm(n))) and > xx <- c(0:n, n:0). A red box highlights the word "Recording" in the History pane's dropdown menu.

# Получение помощи – поиск в справке help

Функция	Результат
help.start()	Глобальная справка
help("xxx")	Справка про xxx
help.search("xxx") или ??xxx	Поиск в справке строк xxx

The screenshot shows the RGui interface with a search results window and a R console window.

**Search Results:**

- Vignettes:
  - [lmtest::lmtest-intro](#) Diagnostic Checking in Regression Relationships [PDF](#) [source](#) [R code](#)
- Code demonstrations:
  - [stats::glm.vr](#) Some glm() examples from V&R with several predictors [\(Run demo\)](#)
  - [stats::lm.glm](#) Some linear and generalized linear modelling examples from 'An Introduction to Statistical Modelling' by Annette Dobson [\(Run demo\)](#)
  - [stats::nlm](#) Nonlinear least-squares using nlm() [\(Run demo\)](#)
- Help pages:
  - [lmtest::coeftest](#) Inference for Estimated Coefficients

**R Gui (64-bit) - [R Console]**

File Edit View Misc Packages Windows

> > ??lm  
starting httpd help server ... done  
> |

# Получение помощи – поиск по загруженным пакетам vignette() или vignette("xxx")

The screenshot shows the RGui (64-bit) application window. In the top-left pane, the R Console displays the following R session:

```
> > ??lm
starting httpd help server ... done
> vignette()
> vignette("adj")
Warning message:
vignette 'adj' not found
> vignette("adjcurve")
> |
```

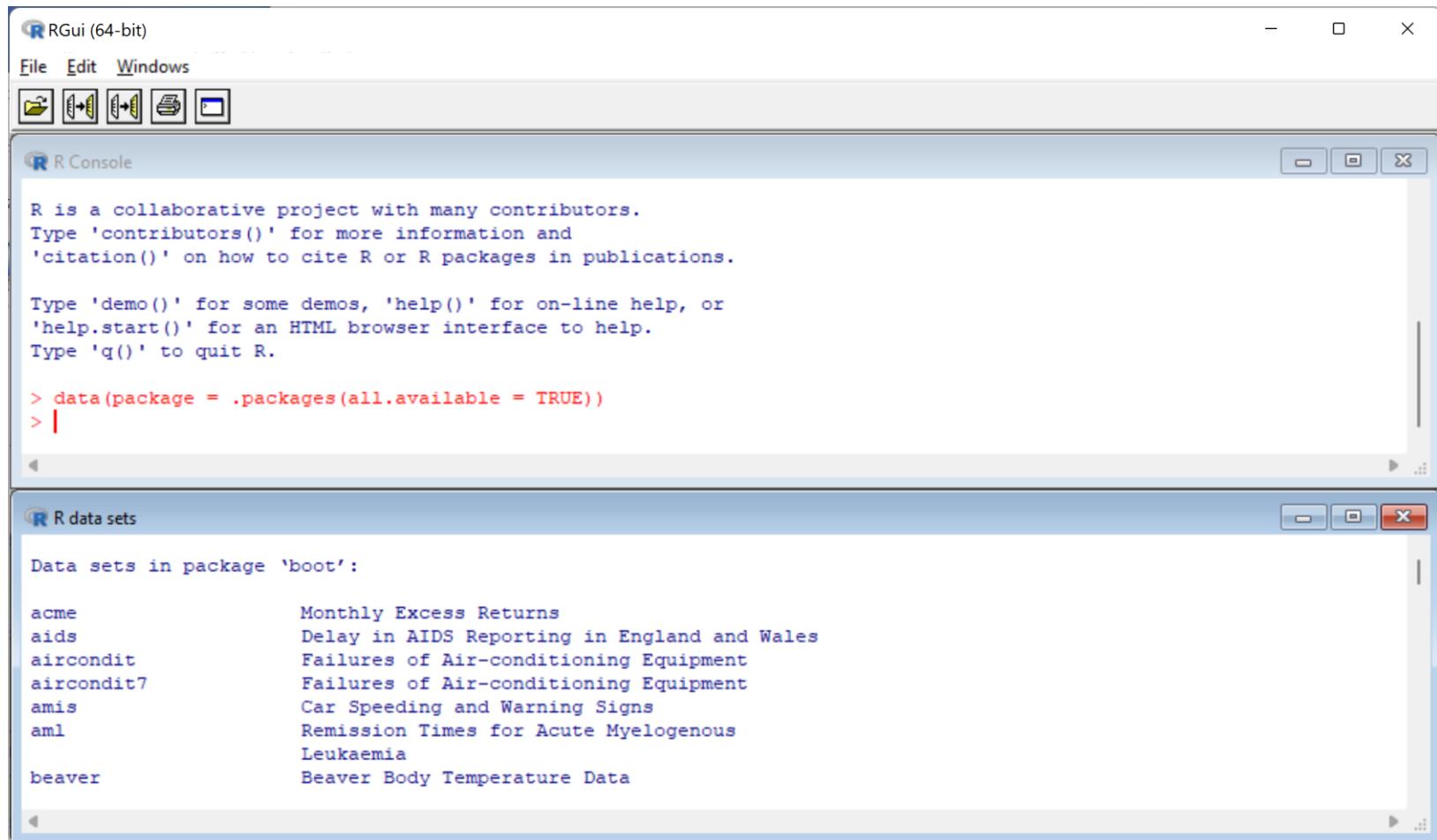
In the bottom-left pane, the R vignettes pane lists the available vignettes in the 'survival' package:

Vignettes in package 'survival':

adjcurve	Adjusted Survival Curves (pdf)
approximate	Approximating a Cox Model (source, pdf)
concordance	Concordance (source, pdf)
discrim	Discrimination and Calibration (pdf)
compete	Multi-state models and competing risks (source, pdf)

A Microsoft Edge browser window is open in the background, displaying the 'adjcurve.pdf' vignette from the 'survival' package. The page title is 'Adjusted Survival Curves' by Terry M Therneau, Cynthia S Crowson, Elizabeth J Atkinson, dated Jan 2015. The content includes an introduction section and a note about the ideal way to test for factor influences on survival.

# Получение помощи – доступные наборы данных data(), data (“xxx”)



# Получение помощи – поиск по имени функций `apropos()`

RGui (64-bit) - [R Console]

File Edit View Misc Packages Windows Help

<--> Stop

```
> apropos("reg")
[1] ".standard-regexp"
[4] "aggregate.ts"
[7] "getDLLRegisteredRoutines.character"
[10] "gregexpr"
[13] "readRegistry"
[16] "regexp"
[19] "registerS3methods"
[22] "state.region"
[25] "reg"
[28] "regexec"
[31] "regexec"
[34] "regexec"
[37] "regexec"
[40] "regexec"
[43] "regexec"
[46] "regexec"
[49] "regexec"
[52] "regexec"
[55] "regexec"
[58] "regexec"
[61] "regexec"
[64] "regexec"
[67] "regexec"
[70] "regexec"
[73] "regexec"
[76] "regexec"
[79] "regexec"
[82] "regexec"
[85] "regexec"
[88] "regexec"
[91] "regexec"
[94] "regexec"
[97] "regexec"
[100] "regexec"
[103] "regexec"
[106] "regexec"
[109] "regexec"
[112] "regexec"
[115] "regexec"
[118] "regexec"
[121] "regexec"
[124] "regexec"
[127] "regexec"
[130] "regexec"
[133] "regexec"
[136] "regexec"
[139] "regexec"
[142] "regexec"
[145] "regexec"
[148] "regexec"
[151] "regexec"
[154] "regexec"
[157] "regexec"
[160] "regexec"
[163] "regexec"
[166] "regexec"
[169] "regexec"
[172] "regexec"
[175] "regexec"
[178] "regexec"
[181] "regexec"
[184] "regexec"
[187] "regexec"
[190] "regexec"
[193] "regexec"
[196] "regexec"
[199] "regexec"
[202] "regexec"
[205] "regexec"
[208] "regexec"
[211] "regexec"
[214] "regexec"
[217] "regexec"
[220] "regexec"
[223] "regexec"
[226] "regexec"
[229] "regexec"
[232] "regexec"
[235] "regexec"
[238] "regexec"
[241] "regexec"
[244] "regexec"
[247] "regexec"
[250] "regexec"
[253] "regexec"
[256] "regexec"
[259] "regexec"
[262] "regexec"
[265] "regexec"
[268] "regexec"
[271] "regexec"
[274] "regexec"
[277] "regexec"
[280] "regexec"
[283] "regexec"
[286] "regexec"
[289] "regexec"
[292] "regexec"
[295] "regexec"
[298] "regexec"
[301] "regexec"
[304] "regexec"
[307] "regexec"
[310] "regexec"
[313] "regexec"
[316] "regexec"
[319] "regexec"
[322] "regexec"
[325] "regexec"
[328] "regexec"
[331] "regexec"
[334] "regexec"
[337] "regexec"
[340] "regexec"
[343] "regexec"
[346] "regexec"
[349] "regexec"
[352] "regexec"
[355] "regexec"
[358] "regexec"
[361] "regexec"
[364] "regexec"
[367] "regexec"
[370] "regexec"
[373] "regexec"
[376] "regexec"
[379] "regexec"
[382] "regexec"
[385] "regexec"
[388] "regexec"
[391] "regexec"
[394] "regexec"
[397] "regexec"
[400] "regexec"
[403] "regexec"
[406] "regexec"
[409] "regexec"
[412] "regexec"
[415] "regexec"
[418] "regexec"
[421] "regexec"
[424] "regexec"
[427] "regexec"
[430] "regexec"
[433] "regexec"
[436] "regexec"
[439] "regexec"
[442] "regexec"
[445] "regexec"
[448] "regexec"
[451] "regexec"
[454] "regexec"
[457] "regexec"
[460] "regexec"
[463] "regexec"
[466] "regexec"
[469] "regexec"
[472] "regexec"
[475] "regexec"
[478] "regexec"
[481] "regexec"
[484] "regexec"
[487] "regexec"
[490] "regexec"
[493] "regexec"
[496] "regexec"
[499] "regexec"
[502] "regexec"
[505] "regexec"
[508] "regexec"
[511] "regexec"
[514] "regexec"
[517] "regexec"
[520] "regexec"
[523] "regexec"
[526] "regexec"
[529] "regexec"
[532] "regexec"
[535] "regexec"
[538] "regexec"
[541] "regexec"
[544] "regexec"
[547] "regexec"
[550] "regexec"
[553] "regexec"
[556] "regexec"
[559] "regexec"
[562] "regexec"
[565] "regexec"
[568] "regexec"
[571] "regexec"
[574] "regexec"
[577] "regexec"
[580] "regexec"
[583] "regexec"
[586] "regexec"
[589] "regexec"
[592] "regexec"
[595] "regexec"
[598] "regexec"
[601] "regexec"
[604] "regexec"
[607] "regexec"
[610] "regexec"
[613] "regexec"
[616] "regexec"
[619] "regexec"
[622] "regexec"
[625] "regexec"
[628] "regexec"
[631] "regexec"
[634] "regexec"
[637] "regexec"
[640] "regexec"
[643] "regexec"
[646] "regexec"
[649] "regexec"
[652] "regexec"
[655] "regexec"
[658] "regexec"
[661] "regexec"
[664] "regexec"
[667] "regexec"
[670] "regexec"
[673] "regexec"
[676] "regexec"
[679] "regexec"
[682] "regexec"
[685] "regexec"
[688] "regexec"
[691] "regexec"
[694] "regexec"
[697] "regexec"
[700] "regexec"
[703] "regexec"
[706] "regexec"
[709] "regexec"
[712] "regexec"
[715] "regexec"
[718] "regexec"
[721] "regexec"
[724] "regexec"
[727] "regexec"
[730] "regexec"
[733] "regexec"
[736] "regexec"
[739] "regexec"
[742] "regexec"
[745] "regexec"
[748] "regexec"
[751] "regexec"
[754] "regexec"
[757] "regexec"
[760] "regexec"
[763] "regexec"
[766] "regexec"
[769] "regexec"
[772] "regexec"
[775] "regexec"
[778] "regexec"
[781] "regexec"
[784] "regexec"
[787] "regexec"
[790] "regexec"
[793] "regexec"
[796] "regexec"
[799] "regexec"
[802] "regexec"
[805] "regexec"
[808] "regexec"
[811] "regexec"
[814] "regexec"
[817] "regexec"
[820] "regexec"
[823] "regexec"
[826] "regexec"
[829] "regexec"
[832] "regexec"
[835] "regexec"
[838] "regexec"
[841] "regexec"
[844] "regexec"
[847] "regexec"
[850] "regexec"
[853] "regexec"
[856] "regexec"
[859] "regexec"
[862] "regexec"
[865] "regexec"
[868] "regexec"
[871] "regexec"
[874] "regexec"
[877] "regexec"
[880] "regexec"
[883] "regexec"
[886] "regexec"
[889] "regexec"
[892] "regexec"
[895] "regexec"
[898] "regexec"
[901] "regexec"
[904] "regexec"
[907] "regexec"
[910] "regexec"
[913] "regexec"
[916] "regexec"
[919] "regexec"
[922] "regexec"
[925] "regexec"
[928] "regexec"
[931] "regexec"
[934] "regexec"
[937] "regexec"
[940] "regexec"
[943] "regexec"
[946] "regexec"
[949] "regexec"
[952] "regexec"
[955] "regexec"
[958] "regexec"
[961] "regexec"
[964] "regexec"
[967] "regexec"
[970] "regexec"
[973] "regexec"
[976] "regexec"
[979] "regexec"
[982] "regexec"
[985] "regexec"
[988] "regexec"
[991] "regexec"
[994] "regexec"
[997] "regexec"
[1000] "regexec"
```

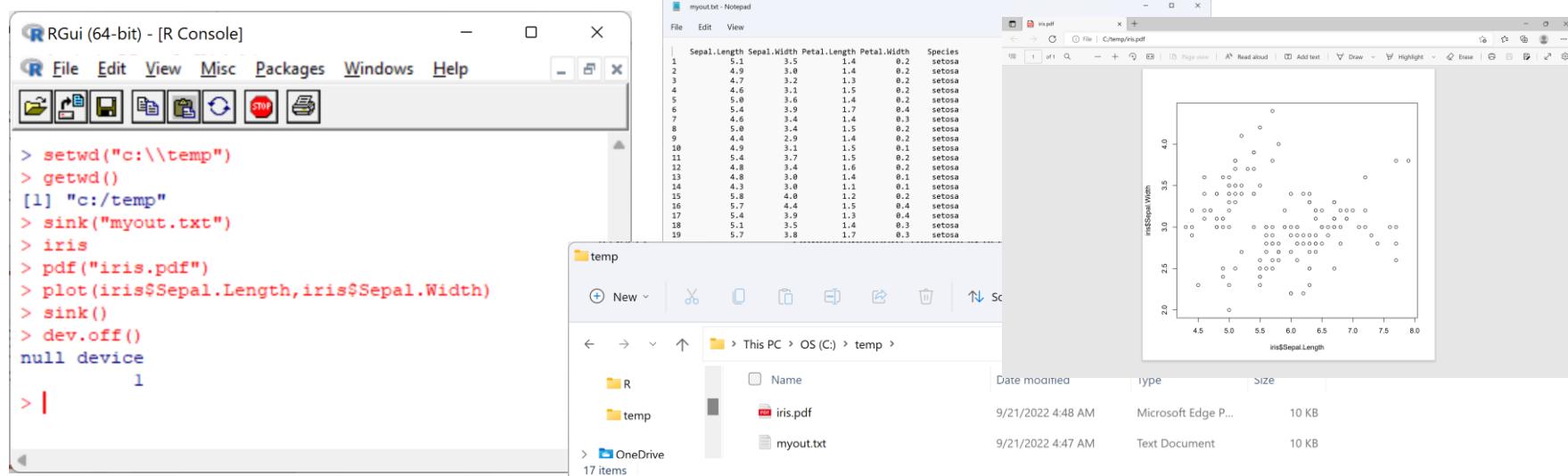
# Рабочее пространство

- По сути это ваш «проект»/«сессия», который содержит историю команд, созданные объекты, загруженные пакеты, построенные графики и так далее
- Можно записать и считать через меню (save/load workspace) или с помощью команд

Функция	Результат
getwd() / setwd()	Узнать/поменять рабочую директорию
ls()	Список объектов рабочего пространства
q()	Выход из сессии
rm("xxx")	Удалить объекты “xxx” из рабочего пространства
history(x)	Показать (x команд), записать, считать историю из файла “file”
savehistory("file")	
loadhistory("file")	
source("script.R")	Запуск скрипта из файла “script.R”
save("xxx", file="file"), load("file")	Записать/считать объекты “xxx” в файл “file”

# Перенаправление вывода

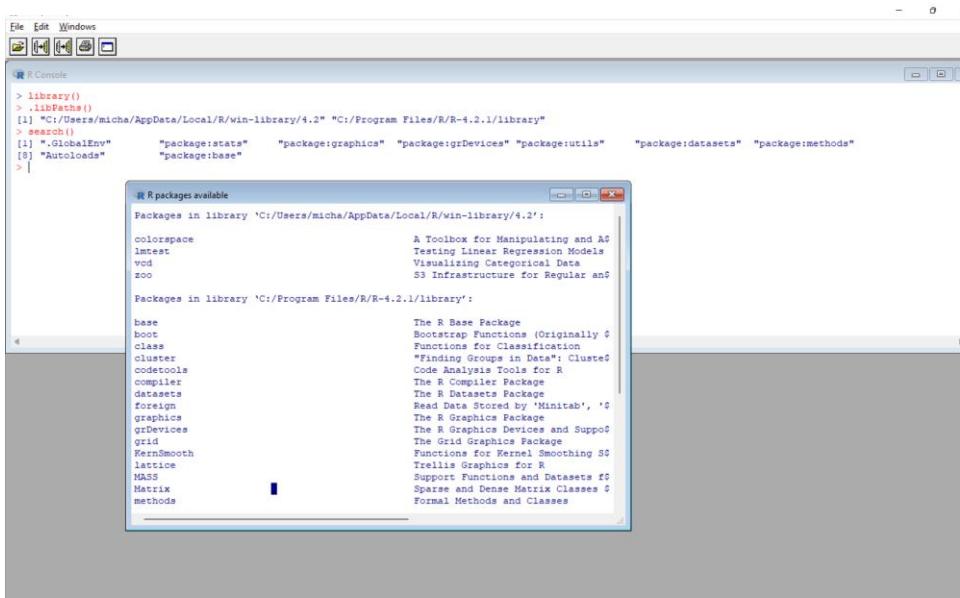
Функция	Результат
sink("file")	Перенаправляет текстовый вывод из консоли в файл "file"
pdf("file")	Перенаправляет графический вывод из консоли в pdf файл "file"
jpeg("file")	Перенаправляет графический вывод из консоли в jpeg файл "file"
png("file")	Перенаправляет графический вывод из консоли в png файл "file"
...	...
sink()	Перенаправляет текстовый вывод обратно в консоль
dev.off()	Перенаправляет графический вывод обратно на экран



# Пакеты

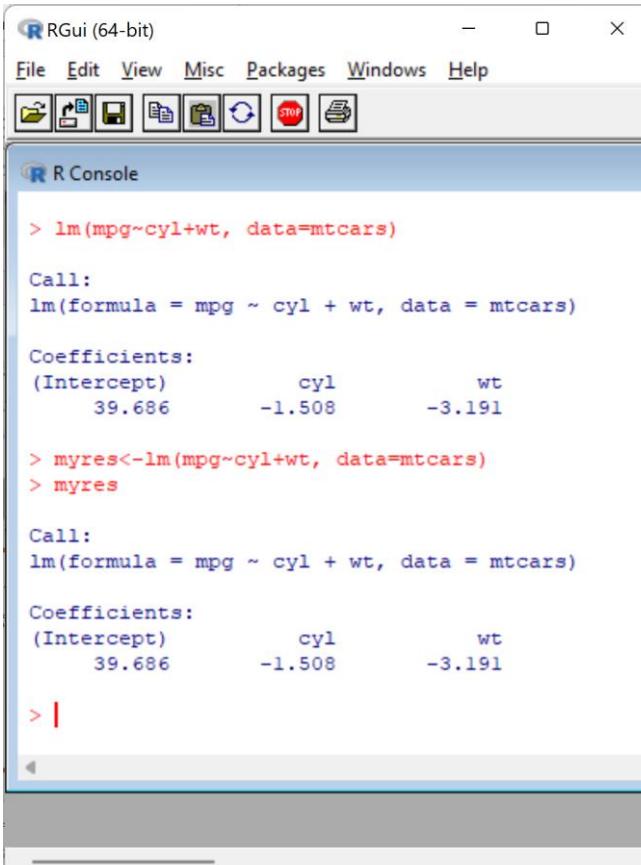
- Основной инструмент расширения функционала

Функция	Результат
library(xxx)	Загрузка установленного пакета xxx
libPath()	Путь до библиотеки
search()	Загруженные пакеты
install.packages("xxx")	Установить пакет "xxx"
installed.packages()	Информация по установленным пакетам
update.package()	Обновление пакетов



# Объекты, переменные, присваивание

- Вывод команды можно сохранить в объект с помощью операции -> или <- (или <<-, ->) для перехода в другую область видимости)



R Gui (64-bit)

File Edit View Misc Packages Windows Help

R Console

```
> lm(mpg~cyl+wt, data=mtcars)
Call:
lm(formula = mpg ~ cyl + wt, data = mtcars)

Coefficients:
(Intercept)      cyl          wt      
 39.686        -1.508       -3.191 

> myres<-lm(mpg~cyl+wt, data=mtcars)
> myres

Call:
lm(formula = mpg ~ cyl + wt, data = mtcars)

Coefficients:
(Intercept)      cyl          wt      
 39.686        -1.508       -3.191 

> |
```

Функция	Результат
length	Длина объекта
dim	Размерность
str	Структура
class	Класс
names	Имена элементов
mode	Режим хранения
cbind(..., ..., ...)	Объединение объектов по столбцам или в строкам
head/tail	Начало/конец объекта
ИМЯ	Печать содержимого
c(..., ..., ...)	Вектор объектов
summary	Описание
fix, edit	Редактирование

# Объекты, переменные, присваивание

```
R Console
[1] "coefficients"   "residuals"      "effects"        "rank"          "fitted.v$"
[10] "call"           "terms"          "model"
> length(myres)
[1] 12
> dim(myres)
NULL
> summary(myres)

Call:
lm(formula = mpg ~ cyl + wt, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.2893 -1.5512 -0.4684  1.5743  6.1004 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 39.6863    1.7150  23.141 < 2e-16 ***
cyl         -1.5078    0.4147  -3.636 0.001064 ** 
wt          -3.1910    0.7569  -4.216 0.000222 *** 
---
Signif. codes:  0 '****' 0.001 '***' 0.01 '**' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.568 on 29 degrees of freedom
Multiple R-squared:  0.8302,    Adjusted R-squared:  0.8185 
F-statistic: 70.91 on 2 and 29 DF,  p-value: 6.809e-12

> |
```

# Объекты – структура и редактирование

The screenshot shows the RGui interface with two windows open:

- R Gui (64-bit) - [R Console]**: This window contains the R command-line interface. The user has run several commands to inspect the structure of the 'myres' object, including `mode(myres)`, `fix(myres)`, and `str(myres)`. The output shows that 'myres' is a list containing various components like coefficients, residuals, effects, rank, fitted.values, assign, qr, and qr values.
- R Gui (64-bit) - [myres - R Editor]**: This window shows the detailed structure of the 'myres' object. It lists all the elements of the list, including their names and values. For example, it shows the 'coefficients' component as a Named num vector of length 39, and the 'residuals' component as a Named num vector of length 32.

The code in the R Console window is as follows:

```
> mode(myres)
[1] "list"
> fix(myres)
Warning message:
In edit.default(get(subx, envir = parent.frame(), deparse = TRUE))
  deparse may be incomplete
>
> str(myres)
List of 12
 $ coefficients : Named num [1:3] 39
   ..- attr(*, "names")= chr [1:3] "(Intercept)", "cyl", "wt"
 $ residuals      : Named num [1:32] -1.5077949682598, -3.19097213898374, ...
   ..- attr(*, "names")= chr [1:32] "Mazda RX4", "Mazda RX4 Wag", "Datsun 710", ...
 $ effects        : Named num [1:32] -1.27914466655683, -0.465446771115904, ...
   ..- attr(*, "names")= chr [1:32] "Valiant", "Duster 360", "Merc 240D", ...
 $ rank           : int 3
 $ fitted.values: Named num [1:32] 22
   ..- attr(*, "names")= chr [1:32] "2.00907843520029", "2.835551160338908, ...
 $ assign          : int [1:3] 0 1 2
 $ qr              :List of 5
 ...$ qr    : num [1:32, 1:3] -5.657
   ...- attr(*, "dimnames")=List of 2
     ...$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
     ...$ : chr [1:3] "(Intercept)" "cyl" "wt"
 ...- attr(*, "assign")= int [1:3] 0 1 2
 ...$ qraux: num [1:3] 1.18 1.02 1.05
```

# Полезные функции обработки сложных объектов

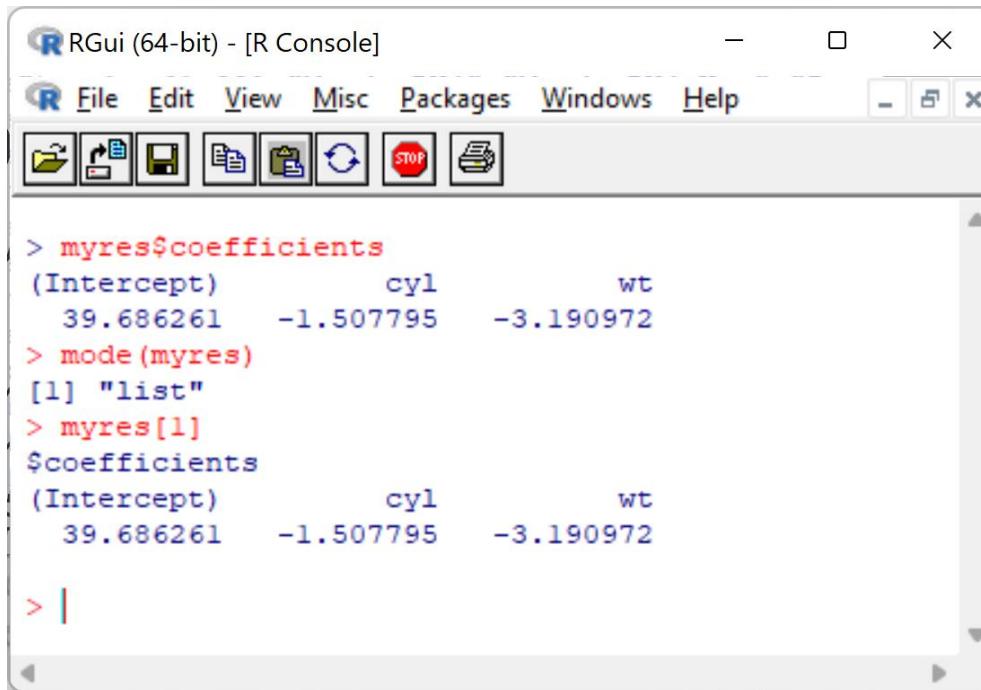
Функция	Результат
length	Длина
seq(from, to, by)	Генерация последовательности от from до to с шагом by
rep(x, n)	Дублировать x n раз
cut(x, n)	Генерация из непрерывной x фактор с n уровнями
cat(,,file=)	Объединяет объекты и записывает их в файл, если задан

R Console window showing R code and its output:

```
> seq(3,10,2)
[1] 3 5 7 9
> rep(c(1,3),2)
[1] 1 3 1 3
> cut(3:10,3)
[1] (2.99,5.33] (2.99,5.33] (2.99,5.33]$
Levels: (2.99,5.33] (5.33,7.67] (7.67,1$
> cat(c(1,2,3),c(4,5,6,7))
1 2 3 4 5 6 7> |
```

# Объекты – именование и доступ к элементам

- Имена цифры и буквы регистрозависимые, с цифр нельзя начинать имя, . и \_ как обычные буквы
- \$ для доступа к элементам объекта
- Либо через соответствующие механизмы в зависимости от типа хранения



```
RGui (64-bit) - [R Console]
File Edit View Misc Packages Windows Help
[Icons: File, Edit, View, Misc, Packages, Windows, Help, Stop]

> myres$coefficients
(Intercept)      cyl          wt
 39.686261   -1.507795   -3.190972
> mode(myres)
[1] "list"
> myres[1]
$coefficients
(Intercept)      cyl          wt
 39.686261   -1.507795   -3.190972

> |
```

# Пользовательские функции как объекты

- Составной оператор { *оператор;оператор;...; оператор* }
- Функции пользователя:

*myfunction<-function (аргумент, аргумент,... аргумент)*

{

*оператор;*

*оператор;*

**return(объект);**

}

- Позиционная, по имени, по умолчанию

```
cnk - R Editor
function (n, k=2)
{
  return(fct(n)*fct(k)/fct(n-k));
}
```

R Console

```
+ if (n>1) return (n*fct(n-1)) else return(1);
+ }
> fct(3)
[1] 6
> fct(5)->a
> a
[1] 120
> fix(fct)
```

fct - R Editor

```
function(n) {
  if (n>1) return (n*fct(n-1)) else return(1);
}
```

R Console

```
> cnk(4,3)
[1] 144
> cnk(n=4,k=3)
[1] 144
> cnk(4)
[1] 24
> fix(cnk)
```

# Управляющие конструкции

- Циклы:

**for (переменная in диапазон) оператор**

**while (условие) оператор**

- Ветвления:

**if (условие) оператор else оператор**

**ifelse (условие, оператор, оператор)**

**switch(выражение, список ...)** – выбор из списка в зависимости от значения выражения

```
R Console
> for (a in 10:12) print(a*10)
[1] 100
[1] 110
[1] 120
> while (a>0) {a=a-5; print(a)}
[1] 7
[1] 2
[1] -3
> |
```

```
R Console
> a<-1
> b<-2
> if (a>b) print(a) else b*1000
[1] 2000
> ifelse (a>b,a,b*1000)
[1] 2000
> switch(b,"один","два","три")
[1] "два"
> |
```

# Структуры и типы данных

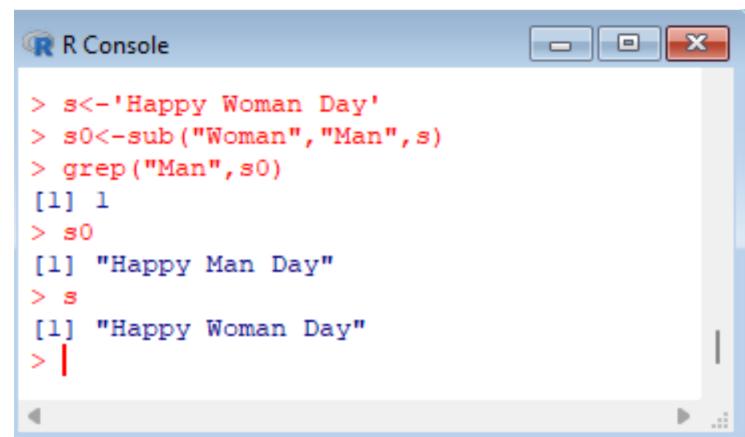
- Простые типы:
  - Числовые с плавающей и фиксированной точкой 1.11, 1E-10
  - Целочисленные – суффикс L, пример 100L
  - Логические – TRUE/FALSE
  - Символьные – в кавычках двойных или одинарных
  - Комплексные 1+5i
- Сложные структуры данных
  - Векторы (однородные, одномерные, скаляров нет)
  - Матрицы (однородные, двумерные)
  - Массивы (однородные, многомерные)
  - Списки (разнородные, одномерные), в том числе составные, например, список списков, список матриц и т.д.
  - Наборы данных (разнородные, двумерные)
  - Факторы (категориальные)

# Важные числовые операции и функции

Операция	Результат	Мат. функция	Результат
+	Сложение	abs	Модуль
-	Вычитание	sqrt	Корень
*	Умножение	round	Округление до заданного числа цифр
** или ^	Возведение в степень	signif	Округление до заданного числа значащих цифр
%%	Остаток от деления	floor	Наибольшее целое не больше чем аргумент
%/%	Целая часть от деления	trunc	Округление к нулю
		ceiling	Наименьшее целое не меньше чем аргумент
Стат. функция	Результат		
mean, median	Положение	cos, sin, tan, acos, asin, atan, acosh, asinh, atanh	Тригонометрия и гиперболические тригонометрические функции
sd, var, range	Разброс	log, exp	Логарифм по разным основаниям и экспонента
min, max	Минимум и максимум		
diff	Разности ряда с лагом		
scale	Центрирование и стандартизация		

# Текстовые и логические функции и операции

Текстовая функция	Результат
nchar	Считает длину строки (length не подходит)
substr	Выделение подстроки
toupper, tolower	Регистр
grep, sub	Поиск и замена по регулярным выражениям
strsplit, paste	Токенизация и склейка



R Console window showing R code execution:

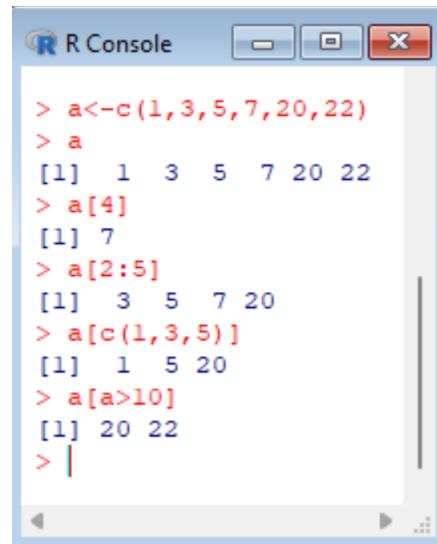
```
> s<-'Happy Woman Day'
> s0<-sub ("Woman", "Man", s)
> grep("Man", s0)
[1] 1
> s0
[1] "Happy Man Day"
> s
[1] "Happy Woman Day"
> |
```

Стандартный набор для сравнения: >, <, <=, >=, ==, !=

Логические операции: ИЛИ x|y, И x&y, НЕ !x, проверка выражения на истинность isTRUE(x)

# Вектора

- Формирование вектора:  
**c (элемент, элемент,..., элемент)**
- Доступ к элементу по индексу [], начиная с 1, можно использовать как фильтр по диапазону или условию



The screenshot shows the R Console window with the following session history:

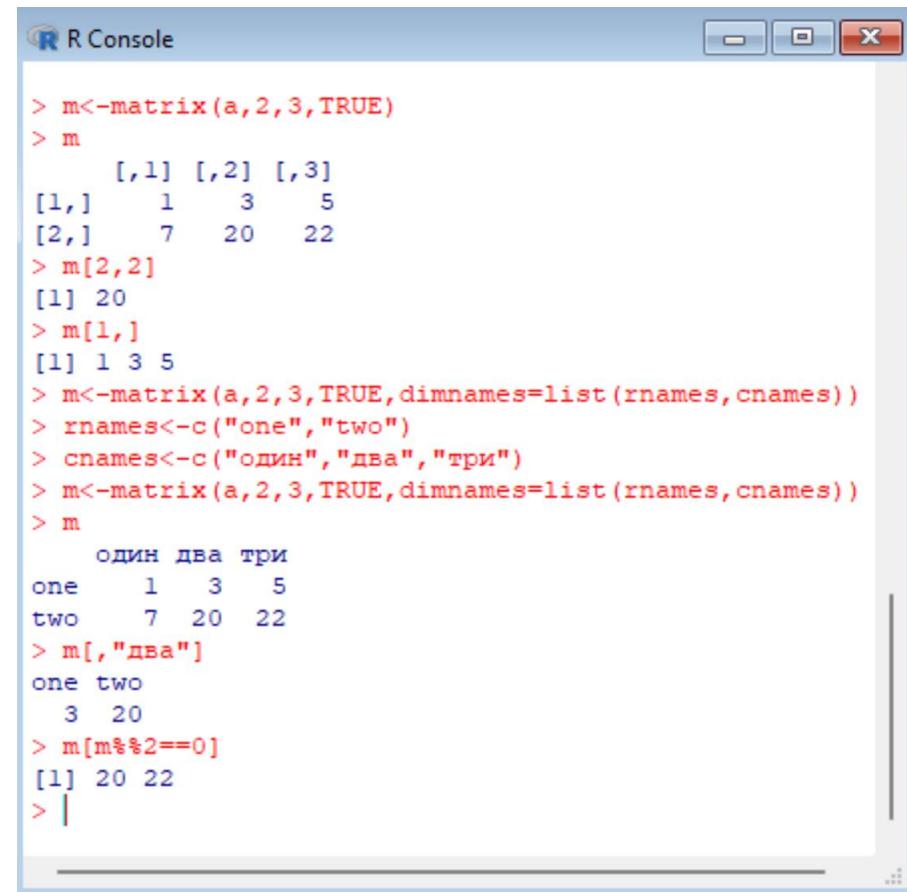
```
> a<-c(1,3,5,7,20,22)
> a
[1]  1  3  5  7 20 22
> a[4]
[1] 7
> a[2:5]
[1]  3  5  7 20
> a[c(1,3,5)]
[1]  1  5 20
> a[a>10]
[1] 20 22
>
```

# Матрицы

- Формирование матрицы из вектора:

**matrix (вектор, nrow, ncol, byrow, dimnames=список)**

- Доступ к элементу по двойному, можно именованному индексу [ , ], начиная с 1,
- можно использовать как фильтр по диапазону, условию или пустой индекс (значит все)



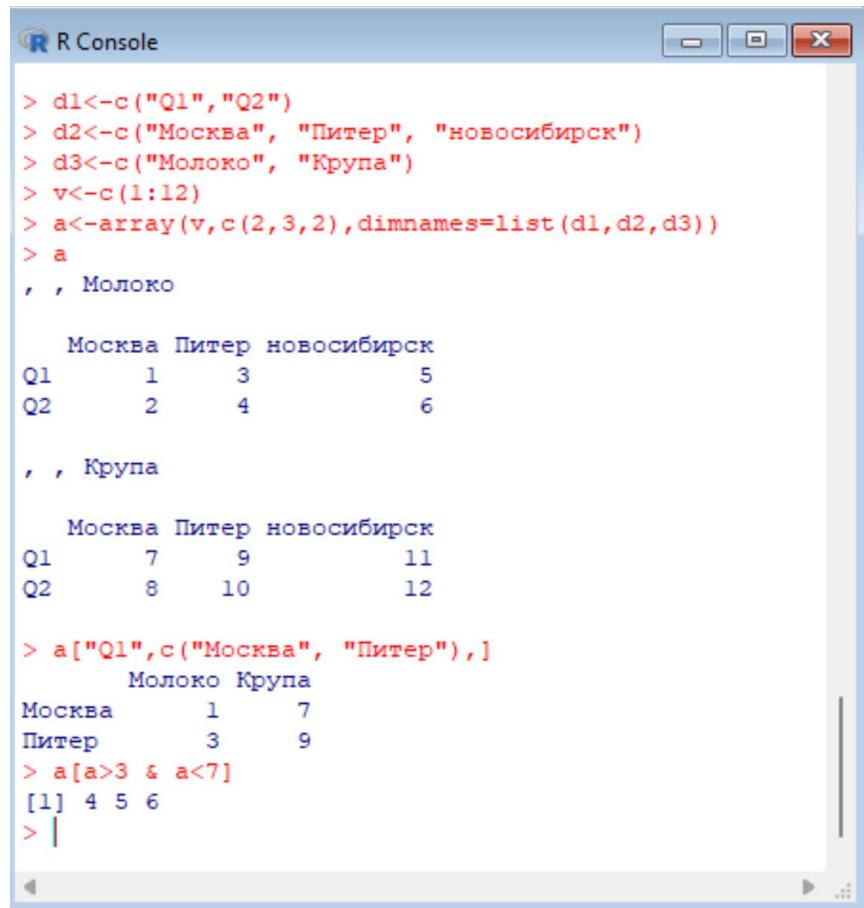
R Console window showing R code and its output. The code demonstrates creating a matrix from a vector, indexing it, and applying row and column names.

```
> m<-matrix(a,2,3,TRUE)
> m
      [,1]  [,2]  [,3]
[1,]    1     3     5
[2,]    7    20    22
> m[2,2]
[1] 20
> m[1,]
[1] 1 3 5
> m<-matrix(a,2,3,TRUE,dimnames=list(rnames,cnames))
> rnames<-c("one","two")
> cnames<-c("один","два","три")
> m<-matrix(a,2,3,TRUE,dimnames=list(rnames,cnames))
> m
      один  два  три
one    1    3    5
two    7   20   22
> m[, "два"]
one two
      3   20
> m[m%%2==0]
[1] 20 22
> |
```

# Массивы

- Формирование массива размерности dimensions с именами измерений dimnames из одномерного вектора vector:  
**array (vector, dimensions, dimnames)**

- Как и в матрицах доступ к элементу по многомерному индексу [,,,], начиная с 1, можно использовать как фильтр по диапазону или условию



```
R Console <--> R

> d1<-c("Q1","Q2")
> d2<-c("Москва", "Питер", "новосибирск")
> d3<-c("Молоко", "Крупа")
> v<-c(1:12)
> a<-array(v,c(2,3,2),dimnames=list(d1,d2,d3))
> a
, , Молоко

    Москва Питер новосибирск
Q1      1      3      5
Q2      2      4      6

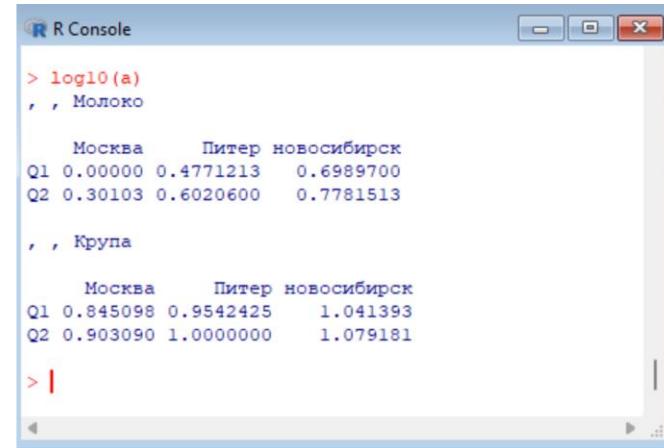
, , Крупа

    Москва Питер новосибирск
Q1      7      9     11
Q2      8     10     12

> a["Q1",c("Москва", "Питер"),]
    Молоко Крупа
Москва      1      7
Питер      3      9
> a[a>3 & a<7]
[1] 4 5 6
>
```

# Применение функций к сложным структурам (векторам, матрицам, массивам)

- Многие стандартные математические, статистические и логические функции применяются **поэлементно** или ко всему набору



```
R Console
> log10(a)
, , Молоко

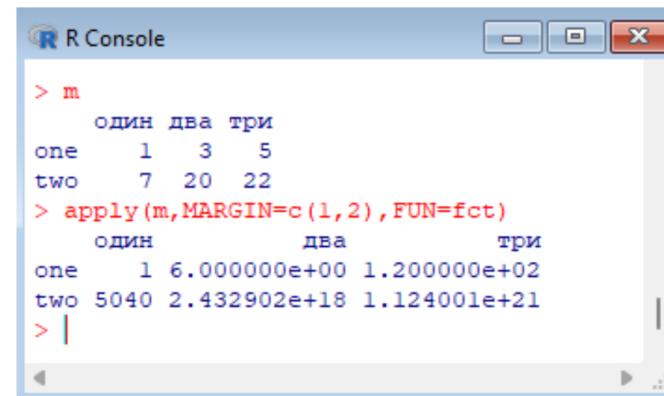
    Москва     Питер    новосибирск
Q1 0.00000 0.4771213 0.6989700
Q2 0.30103 0.6020600 0.7781513

, , Крупа

    Москва     Питер    новосибирск
Q1 0.845098 0.9542425 1.041393
Q2 0.903090 1.0000000 1.079181

> |
```

- Есть механизм применения и пользовательской функции FUN к объекту x (матрица, векторам) в заданном диапазоне **MARGIN** (1: rows, 2: columns, c(1, 2): rows и columns):  
**apply(x, MARGIN, FUN)**



```
R Console
> m
    один   два   три
one     1     3     5
two     7    20    22
> apply(m,MARGIN=c(1,2),FUN=fct)
    один           два           три
one     1 6.000000e+00 1.200000e+02
two 5040 2.432902e+18 1.124001e+21
> |
```

# Наборы (таблицы) данных

- Аналог картежей в реляционных СУБД или datasets в SAS
- Основной тип для хранения и обработки данных
- Состоит из именованных колонок разного типа, по сути выборка, строки – наблюдения, колонки – признаки
- Имеет развитый инструментарий для горизонтальной и вертикальной фильтрации, горизонтального (union) и вертикального (join) объединений
- Формирование:
  - Программное или «ручное» из векторов
  - Импорт (самый часто используемый путь)
  - На основе манипуляций с другими таблицами

# Построение таблиц и обращение к элементам

- Построение кодом:

**data.frame(column, column, ..., column)**

- «Ручное» построение:

**edit()**

- Доступ к элементам и фильтрация через \$

The screenshot shows the RStudio interface with two main windows: the Data Editor and the R Console.

**Data Editor:** A grid-based editor showing a data frame with columns: cust\_id, age, gender, var4, var5, var6, and var7. Rows 1 through 4 have data: (1, 20, m), (2, 35, f), (3, 18, f), (4, 33, m). Rows 5 and 6 are empty.

**R Console:** The command history and output window.

```
> cust_id<-c(1,2,3,4)
> age<-c(20,35,18,33)
> gender<-c("m","f","f","m")
> dt<-data.frame(cust_id,age,gender)
Error: unexpected symbol in "dt<-data.f$"
> dt<-data.frame(cust_id,age,gender)
> dt
  cust_id age gender
1       1   20     m
2       2   35     f
3       3   18     f
4       4   33     m
> edit(dt)
```

# Области видимости attach, detach, with

- attach/detach добавляет все колонки указанной таблицы в область видимости, чтобы можно было работать с полями без \$
- Функция with создает область видимости «внутри» объекта, а присваивать одноименные объекты из более общей области видимости можно через <<- и ->>

```
R Console
> attach(dt)
The following objects are masked _by_ .$  
  age, cust_id, gender  

> age
[1] 20 35 18 33
> dt$age
[1] 20 35 18 33
> age[2]<-55
> age
[1] 20 55 18 33
> dt$age
[1] 20 35 18 33
> detach(dt)
> |
```

```
R Console
> with(dt, {status<- (age>25); status})
[1] FALSE TRUE FALSE TRUE
> with(dt, {status<<-(age<25); status})
[1] TRUE FALSE TRUE FALSE
> status
[1] TRUE FALSE TRUE FALSE
> |
```

# Категориальные признаки в факторах и списках

- Функция **factor** кодирует последовательностью числовых значений список категориальных значений из *вектор*:

**factor** (*вектор*, ORDER, levels, labels)

- Самая сложная и гибкая структура данных за счет разнородности и возможности делать «список списков»

**list** (*объект, имя=объект, ...*)

- Доступ по индексу через **[ ]** или по имени через **\$**

```
R Console
> ff<-factor(dt$gender,labels=c("f","m"), order=TRUE)
> ff
[1] m f f m
Levels: f < m
> ff<-factor(dt$gender,levels=c("f","m"), labels=c(10,20), order=TRUE)
> ff
[1] 20 10 10 20
Levels: 10 < 20
> |
```

```
R Console
> l<-list("Всем Привет!", age,gender, status=c(1,2), TRUE)
> l
[[1]]
[1] "Всем Привет!"

[[2]]
[1] 20 55 18 33

[[3]]
[1] "m" "f" "f" "m"

$status
[1] 1 2

[[5]]
[1] TRUE

> l[3]
[[1]]
[1] "m" "f" "f" "m"

> l$status
[1] 1 2
> |
```

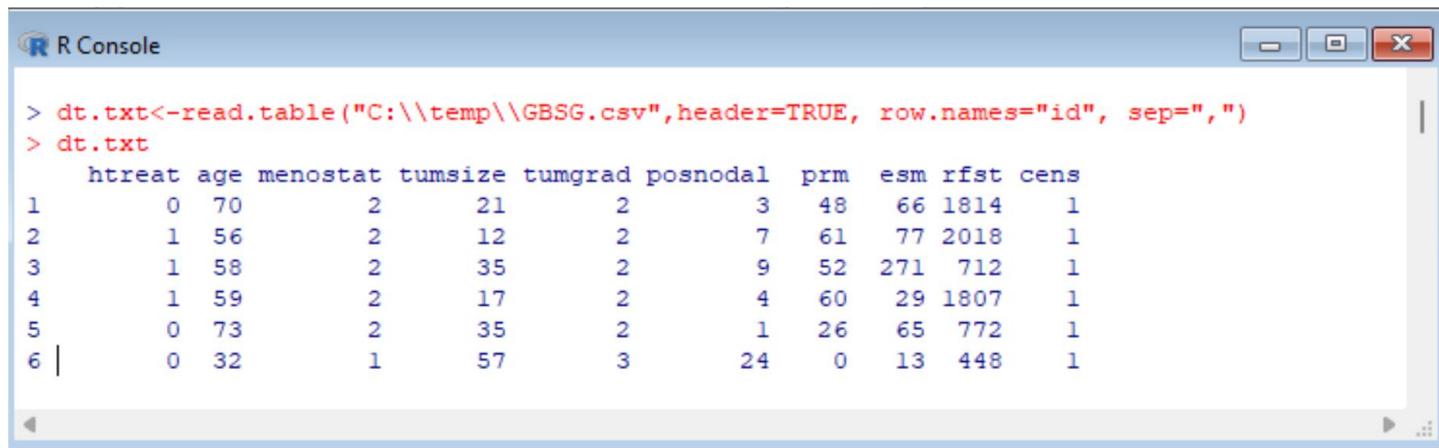
# Import/export табличных данных

- Поддерживается много форматов: csv, xls, SAS, SPSS, XML, ODBC ...
- Для импорта большинства форматов кроме текстовых необходимо подключить соответствующую библиотеку
- Импорт ASCII файла file, с признаком есть ли заголовочная строка header, указанным разделителем sep, и опционально указан столбец с именами строк

**read.table(file, header, sep, row.names)**

**write.table(объект, file, sep, row.names)**

**write.csv(объект, file, sep, row.names)**



R Console window showing the output of the R code:

```
> dt.txt<-read.table("C:\\\\temp\\\\GBSG.csv",header=TRUE, row.names="id", sep=",")  
> dt.txt  
   htreat age menostat tumsize tumgrad posnodal   prm    esm rfst cens  
 1      0   70        2     21      2       3    48     66 1814    1  
 2      1   56        2     12      2       7    61     77 2018    1  
 3      1   58        2     35      2       9    52    271  712    1  
 4      1   59        2     17      2       4    60     29 1807    1  
 5      0   73        2     35      2       1    26     65  772    1  
 6      0   32        1     57      3      24     0    13  448    1
```

# Простая обработка табличных данных

- Перекодирование значений
- Переименование переменных
- Работа с пропусками
- Календарные даты и время
- Проверка и преобразование типов
- Сортировка
- Отбор переменных и фильтрация наблюдений
- Горизонтальное и вертикальное объединение наборов
- Subset
- SQL возможности в R

# Изменение значений и имен переменных

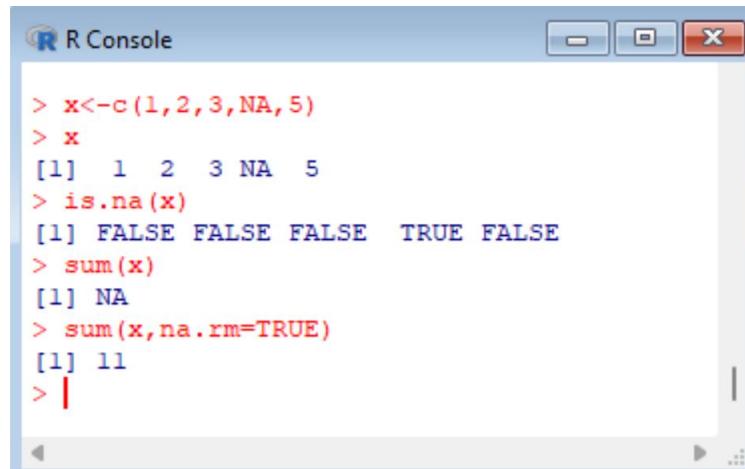
- «Вручную» fix
- Изменение значений с доступом через \$ или attach/with по индексу или по фильтру
  - таблица\$переменная [условие]<-выражение*
- Изменение имен через структуру объекта с функцией names

```
> dt.txt$age<-c(dt.txt$age>25) & (dt.txt$age<65)]<-"Mid"
> dt.txt$age<-c(dt.txt$age>=65)<-"Old"
> dt.txt
   htreat age menostat tumsize tumgrad posnodal  prm  esm rfst cens agect age.ct
1      0    70          2       21      2        3    48    66 1814     1   Old     Old
2      1    56          2       12      2        7    61    77 2018     1   Mid    <NA>
3      1    58          2       35      2        9    52   271  712     1   Mid    <NA>
4      1    59          2       17      2        4    60    29 1807     1   Mid    <NA>

5
6
7 > names(dt.txt)
8 [1] "htreat"    "age"        "menostat"   "tumsize"    "tumgrad"    "posnodal"   "prm"        "esm"        "rfst"        "$
9 > names(dt.txt)[2]
[1] "age"
> names(dt.txt)[2]<-"Возраст"
> names(dt.txt)
[1] "htreat"    "Возраст"    "menostat"   "tumsize"    "tumgrad"    "posnodal"   "prm"        "esm"        "rfst"        "$
```

# Пропущенные значения

- Семантика – не определено или неизвестно
- Причины – пропуски в данных, замена ошибок, артефактов и выбросов, объединение с источниками, где отсутствуют объекты
- Специальное значение `NA`, проверка на пропуск `is.na`
- Многие функции имеют optionalный параметр, считать пропуски или нет, например `na.rm`

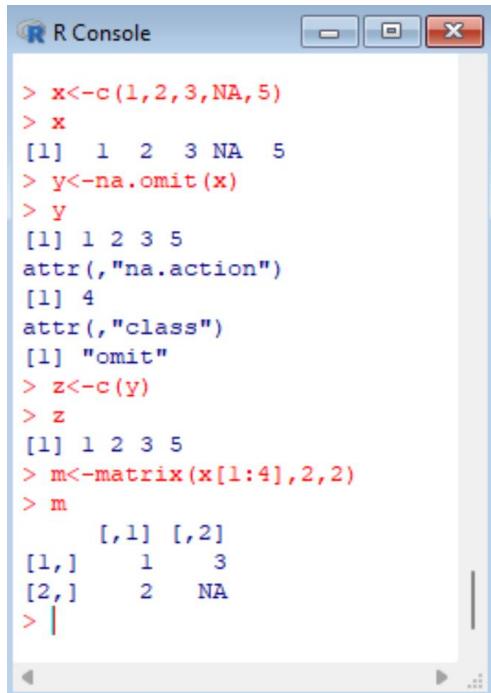


R Console window showing R code execution:

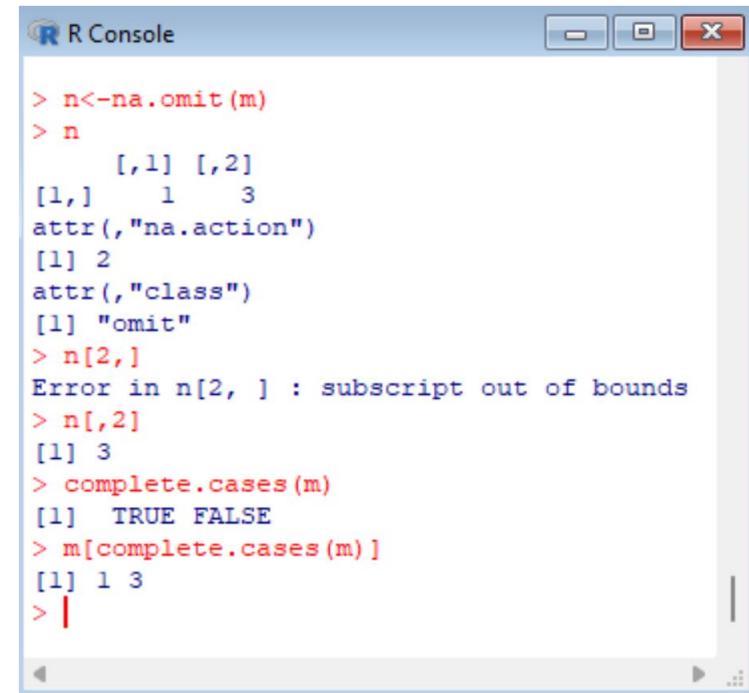
```
> x<-c(1,2,3,NA,5)
> x
[1] 1 2 3 NA 5
> is.na(x)
[1] FALSE FALSE FALSE TRUE FALSE
> sum(x)
[1] NA
> sum(x,na.rm=TRUE)
[1] 11
> |
```

# «Ручная» обработка пропущенных значений

- Оставить только записи без пропусков  
**na.omit(объект, ...)**
- Проверка наблюдений на заполненность  
**complete.cases(объект)**



```
R Console
> x<-c(1,2,3,NA,5)
> x
[1] 1 2 3 NA 5
> y<-na.omit(x)
> y
[1] 1 2 3 5
attr("na.action")
[1] 4
attr("class")
[1] "omit"
> z<-c(y)
> z
[1] 1 2 3 5
> m<-matrix(x[1:4],2,2)
> m
     [,1] [,2]
[1,]     1     3
[2,]     2     NA
> |
```



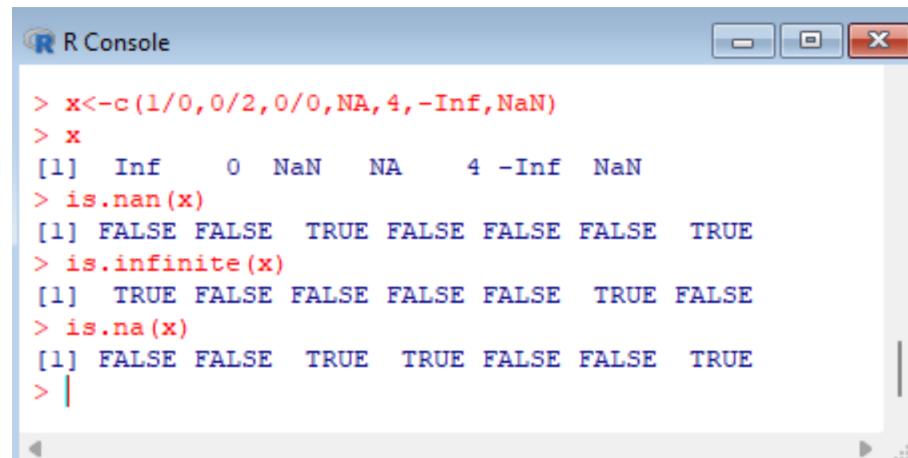
```
R Console
> n<-na.omit(m)
> n
     [,1] [,2]
[1,]     1     3
attr("na.action")
[1] 2
attr("class")
[1] "omit"
> n[2,]
Error in n[2, ] : subscript out of bounds
> n[,2]
[1] 3
> complete.cases(m)
[1] TRUE FALSE
> m[complete.cases(m)]
[1] 1 3
> |
```

# Другие виды неопределенности значений

- Семантика неопределенности

- NA=«не знаю»
- NaN=«не число»
- Inf и -Inf = «бесконечности»

x	is.na	is.nan	is.infinite
x<-NA	TRUE	FALSE	FALSE
x<-1/0	TRUE	TRUE	FALSE
x<-0/0	FALSE	FALSE	TRUE



R Console

```
> x<-c(1/0,0/2,0/0,NA,4,-Inf,NaN)
> x
[1] Inf    0   NaN   NA    4   -Inf   NaN
> is.nan(x)
[1] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE
> is.infinite(x)
[1] TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
> is.na(x)
[1] FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
> |
```

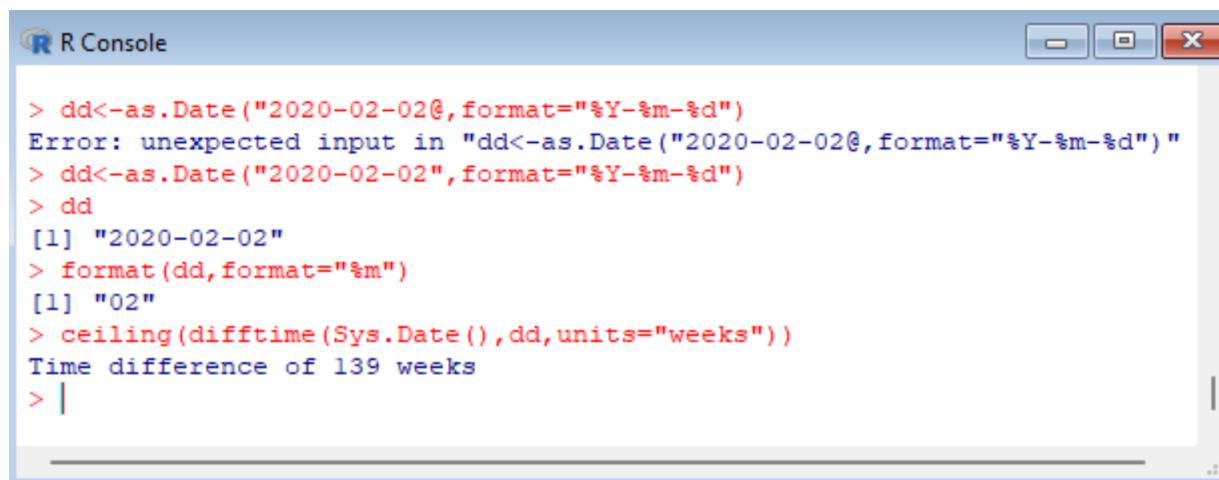
# Календарные даты

- Читаются и пишутся текстом в определенном формате, хранятся в числовом виде, преобразуются с помощью

**as.Date(дата, формат)**

**format(дата, формат)**

Функции	значение	символ	значение
date	сегодня	%d	День месяца
difftime(data1, data2, units)	корректный расчет разности дат	%a, %A %b, %B %m, %M %y, %Y	День недели Месяц Номер месяца год



R Console window showing R code execution:

```
> dd<-as.Date("2020-02-02@,format="%Y-%m-%d")
Error: unexpected input in "dd<-as.Date("2020-02-02@,format="%Y-%m-%d")"
> dd<-as.Date("2020-02-02",format="%Y-%m-%d")
> dd
[1] "2020-02-02"
> format(dd,format="%m")
[1] "02"
> ceiling(difftime(Sys.Date(),dd,units="weeks"))
Time difference of 139 weeks
> |
```

# Проверка и преобразование базовых типов

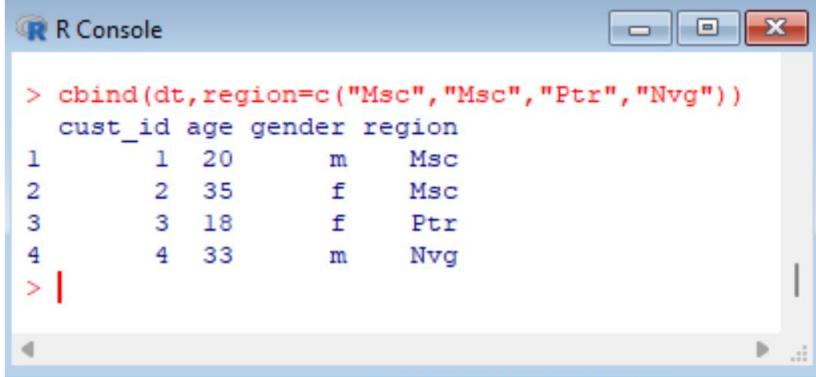
Проверка	Преобразование
is.numeric	as.numeric
is.character	as.character
is.vector	as.vector
is.matrix	as.matrix
is.data.frame	as.data.frame
is.factor	as.factor
is.logical	as.logical

R Console window showing R code and its output:

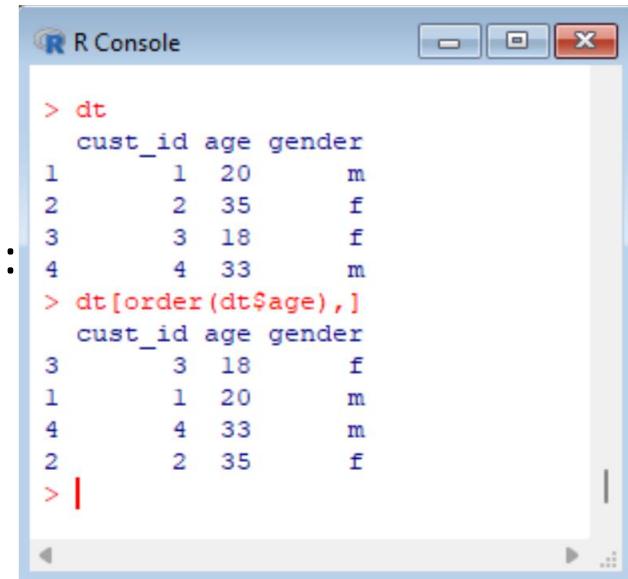
```
> a  
, , Молоко  
  
Москва Питер новосибирск  
Q1      1      3      5  
Q2      2      4      6  
  
, , Крупа  
  
Москва Питер новосибирск  
Q1      7      9     11  
Q2      8     10     12  
  
> is.numeric(a)  
[1] TRUE  
> is.character(a)  
[1] FALSE  
> is.matrix(a)  
[1] FALSE  
> is.data.frame(a)  
[1] FALSE  
> is.array(a)  
[1] TRUE  
> b<-as.character(a)  
> b  
[1] "1"  "2"  "3"  "4"  "5"  "6"  
> is.numeric(b)  
[1] FALSE  
> is.character(b)  
[1] TRUE  
> is.matrix(b)  
[1] FALSE  
> is.array(b)  
[1] FALSE  
> is.vector(b)  
[1] TRUE  
> |
```

# Сортировка и расширение наборов

- Сортировка осуществляется функцией:  
**order(поле, поле, поле)**
- «Горизонтальное» добавление столбцов:  
**cbind(data.A, data.B)**

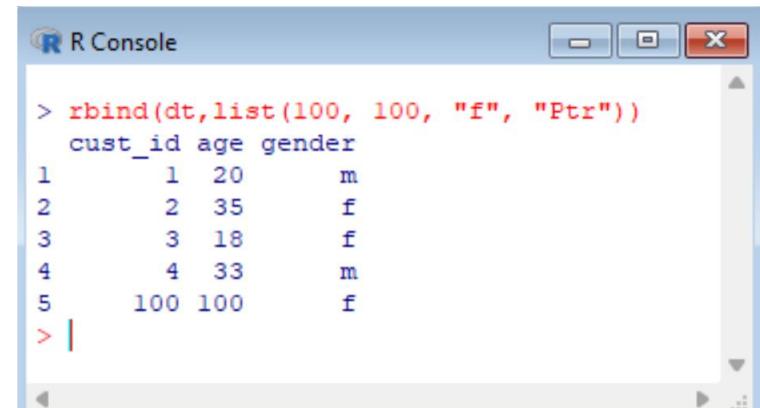


```
R Console
> cbind(dt,region=c("Msc","Msc","Ptr","Nvg"))
   cust_id age gender region
1       1   20      m     Msc
2       2   35      f     Msc
3       3   18      f      Ptr
4       4   33      m      Nvg
> |
```



```
R Console
> dt
   cust_id age gender
1       1   20      m
2       2   35      f
3       3   18      f
4       4   33      m
> dt[order(dt$age),]
   cust_id age gender
3       3   18      f
1       1   20      m
4       4   33      m
2       2   35      f
> |
```

- «Вертикальное» добавление строк:  
**rbind(data.A, data.B)**



```
R Console
> rbind(dt,list(100, 100, "f", "Ptr"))
   cust_id age gender
1       1   20      m
2       2   35      f
3       3   18      f
4       4   33      m
5    100 100      f
> |
```

# Отбор переменных в выборке

- Изменение структуры объекта через `names`
- Варианты удаления колонок: список имен, индекс, запись `NULL`

```
R Console
> dt
  cust_id age gender
1      1    20     m
2      2    35     f
3      3    18     f
4      4    33     m
> rmvars<-names(dt) %in% c("gender")
> rmvars
[1] FALSE FALSE  TRUE
> dt[!rmvars]
  cust_id age
1      1    20
2      2    35
3      3    18
4      4    33
> |
```

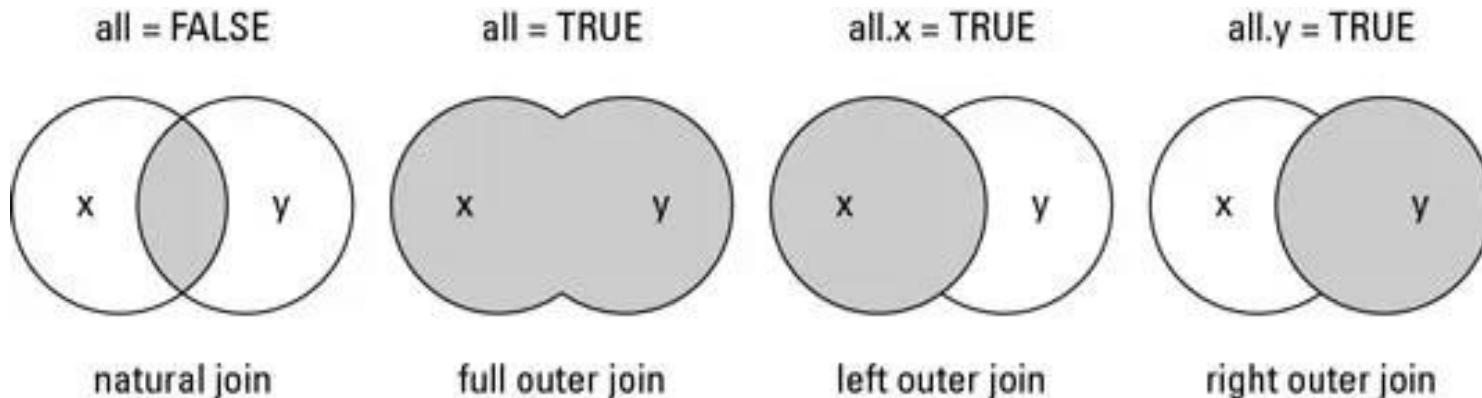
```
R Console
> dt[-2]
  cust_id gender
1      1     m
2      2     f
3      3     f
4      4     m
> |
```

```
R Console
> dt$cust_id<-NULL
> dt
  age gender
1   20     m
2   35     f
3   18     f
4   33     m
> |
```

# Горизонтальное объединение наборов

- Аналог внутреннего и внешнего join в SQL, две таблицы, по ключу (можно составному, тогда список), all отвечает за тип join, внутренний (по умолчанию) или внешний левый, правый:

`merge(data.A, data.B, by="поле", all, all.x, all.y, by.x, by.y)`



# Пример горизонтального объединения наборов

```
R Console

> df0
  cust_id prof salary gender
1       1     m     100     m
2       2     w      50     f
3       3     e     200     f
4       4     w      75     m
5       5     w     150     f
6       6     m     120     f
> df1<-df0[df0$cust_id %% 2 ==0,]
> df1$prof<-NULL
> df2<-df0[1:4,]
> df2$gender<-NULL
> df1
  cust_id salary gender
2       2      50     f
4       4      75     m
6       6     120     f
> df2
  cust_id prof salary
1       1     m     100
2       2     w      50
3       3     e     200
4       4     w      75
> |
```

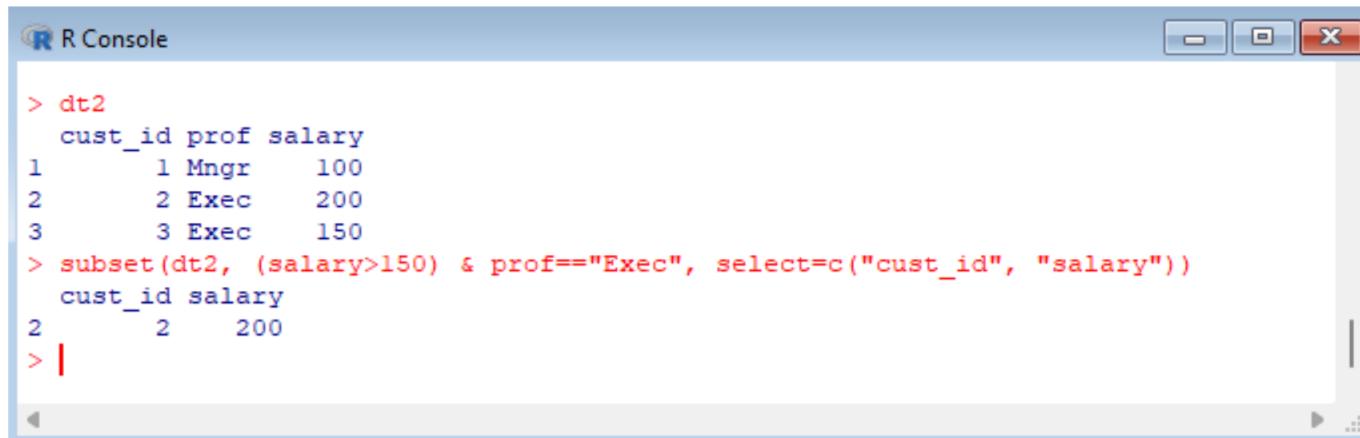
```
R Console

> merge(df1,df2)
  cust_id salary gender prof
1       2      50     f     w
2       4      75     m     w
> merge(df1,df2,all=TRUE)
  cust_id salary gender prof
1       1     100    <NA>   m
2       2      50     f     w
3       3     200    <NA>   e
4       4      75     m     w
5       6     120     f    <NA>
> merge(df1,df2,all.x=TRUE)
  cust_id salary gender prof
1       2      50     f     w
2       4      75     m     w
3       6     120     f    <NA>
> merge(df1,df2,all.y=TRUE)
  cust_id salary gender prof
1       1     100    <NA>   m
2       2      50     f     w
3       3     200    <NA>   e
4       4      75     m     w
> |
```

# ФУНКЦИЯ subset

- Subset - аналог запросу select в SQL, но без join и агрегаций.  
Выбирает переменные из списка *select* при условии *subset*

**subset(data, subset, select)**



The screenshot shows an R console window titled "R Console". The code entered is:

```
> dt2
  cust_id prof salary
1      1 Mngr    100
2      2 Exec    200
3      3 Exec    150
> subset(dt2, (salary>150) & prof=="Exec", select=c("cust_id", "salary"))
  cust_id salary
2      2     200
> |
```

The output shows the filtered data frame with only rows where salary is greater than 150 and prof is "Exec", and only selecting the columns "cust\_id" and "salary".

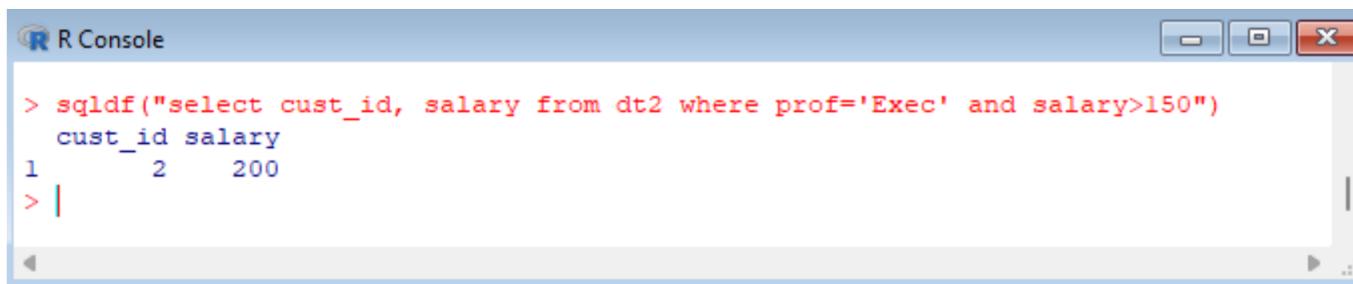
# Работа с данными через ODBC и SQL

- install.packages ("RODBC")
- library (RODBC)

Функция	Результат
odbcConnect	Установить соединение
sqlFetch	Открыть таблицу в соединении
sqlSave	Сохраняет набор в таблице базы
sqlDrop	Удалить таблицу
odbcClose	Закрыть соединение

- install.packages ("sqldf")
- library (sqldf)

Функция	Результат
sqldf	Выполнение sql запроса над R набором данных



R Console

```
> sqldf("select cust_id, salary from dt2 where prof='Exec' and salary>150")
  cust_id salary
1          2     200
> |
```

# Агрегирование и транспонирование

- Агрегирование набора *data* произвольной функцией *FUN* с группировкой по *by*:

**aggregate(*data*, *by*, *FUN*)**

- Транспонирование матрицы или таблицы: **t(*data*)**

```
R Console
> dt
  age gender
1 20   m
2 35   f
3 18   f
4 33   m
> aggregate(dt$age, by=list(gender), FUN=mean, na.rm=TRUE)
  Group.1    x
1      f 26.5
2      m 26.5
> |
```

```
R Console
> dt3<-t(dt)
> dt3
     [,1] [,2] [,3] [,4]
age    "20" "35" "18" "33"
gender "m"  "f"  "f"  "m"
> class(dt3)
[1] "matrix" "array"
> |
```

# Простой ввод/вывод текстовых данных

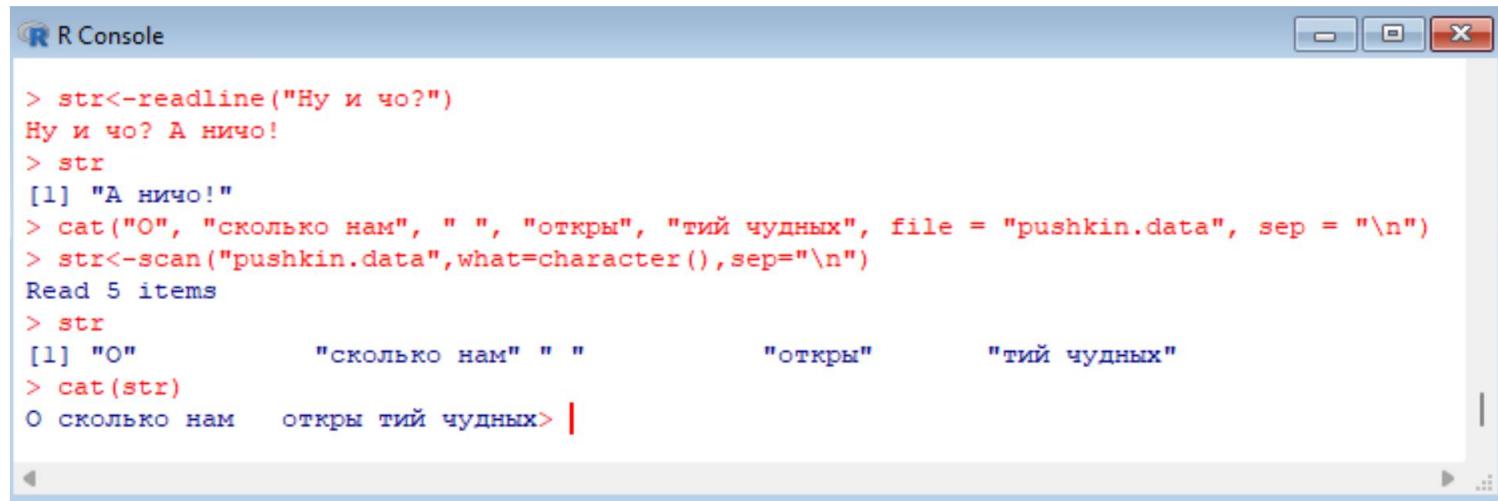
- Чтение «сырых» данных типа `what` (или клавиатурный ввод, если не указан файл `file`) с разделителем `sep` в виде списка:

**`scan(file=,what=,sep=)`**

- Клавиатурный ввод текстовой строки:

**`readline()`**

- Запись последовательности объектов в файл через `cat`



The screenshot shows the R Console window with the following interaction:

```
R Console
> str<-readline("Ну и чо?")
Ну и чо? А ничо!
> str
[1] "А ничо!"
> cat("О", "сколько нам", " ", "откры", "тий чудных", file = "pushkin.data", sep = "\n")
> str<-scan("pushkin.data",what=character(),sep="\n")
Read 5 items
> str
[1] "О"           "сколько нам" " "           "откры"      "тий чудных"
> cat(str)
О сколько нам  откры тий чудных> |
```

The console shows the user inputting text via `readline`, reading it into `str`, then writing it back to the console using `cat` with a separator of a new line character (`\n`). The resulting output is a vector of characters containing the original text.

# “C style” ввод/вывод через соединения

- «Знакомые» методы работы с потоками данных, включая файлы, архивы, пайпы, сокеты, url и т.д., задается источник, тип доступа, кодировка и многие другие

Функция	Результат
open, file, gzfile, url, pipe, socketConnection, ...	Открытие источника
close, unlink	Закрытие источника
flush	Сбрасывает буфер
isOpen	Удалить таблицу
seek	Перестановка указателя (где поддерживается)
readBin, writeBin	Запись и чтение бинарных данных
readLines, writeLines	Запись и чтение текстовых данных
readChar, writeChar	Запись и чтение символьных данных

# Примеры ввода данных через соединение



```
R Console
> cn
A connection with
  description "https://www.wikipedia.org/"
  class      "url-libcurl"
  mode       "r"
  text       "text"
  opened     "closed"
  can read   "yes"
  can write  "no"
> rl<-readLines(cn, n=10)
> rl
[1] "<!DOCTYPE html>"
[2] "<html lang=\"en\" class=\"no-js\">"
[3] "<head>"
[4] "<meta charset=\"utf-8\">"
[5] "<title>Wikipedia</title>"
[6] "<meta name=\"description\" content=\"Wikipe$"
[7] "<script>"
[8] "document.documentElement.className = document$"
[9] "</script>"
[10] "<meta name=\"viewport\" content=\"initial-sca$"
> class(rl)
[1] "character"
> mode(rl)
[1] "character"
> class(cn)
[1] "url"        "connection"
> mode(cn)
[1] "numeric"
> |
```