



Занятие 4

if, for, apply, merge

тесты ассоциации

6 марта 2013

Биологическая задача

- Найти аллельные варианты, которые отличают популяцию европейцев и африканцев
- Для каждого человека можем узнать:
 - К какой расе принадлежит?
 - Присутствует ли данный аллельный вариант* в данном месте последовательности ДНК?

A

C

A

G

T

A



присутствие аллельного варианта
(«мутация»)

*в гетеро- или гомозиготном состоянии²

Биологическая задача

- Задача: какие генетические варианты (мутации) отличают популяцию европейцев и африканцев
- Исходные данные:

> `variant_counts`

Кол-во людей
без альт. варианта

Кол-во людей с
альтернативным
вариантом

Хромосома	Координата	EA_ref	EA_alt	AA_ref	AA_alt
chr6	292452	4028	27	2089	12
chr6	292461	4100	1	2117	0

...

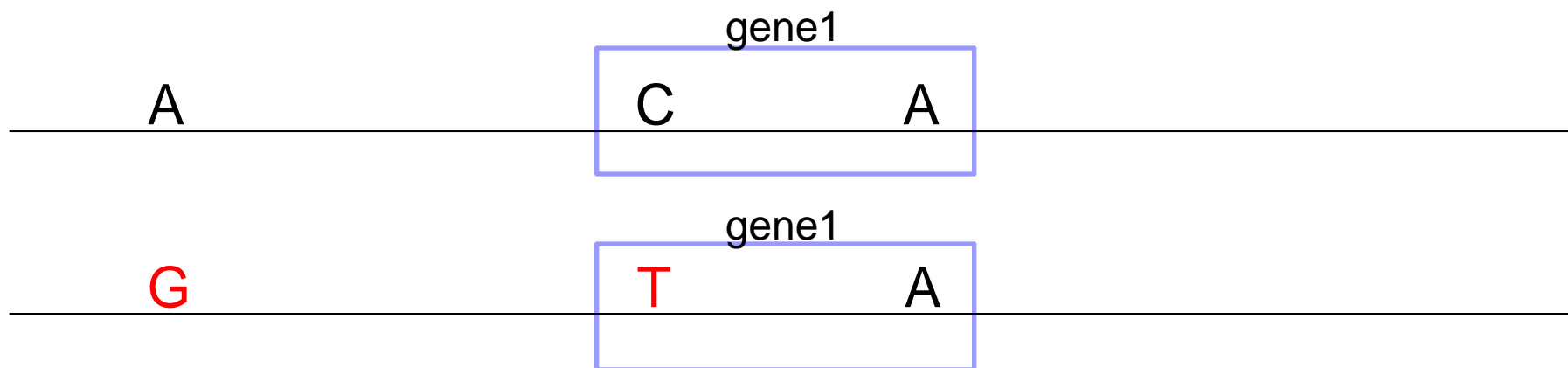
`variant_counts.csv`

Европейцы

Африканцы

Биологическая задача

- Второй шаг – понять, в каких генах эти расположены наиболее значимые мутации



>variant_genes

Хромосома	Координата	Гены
chr	pos	genes
chr6	292452	DUSP22
chr6	292461	DUSP22

...

variant_genes.csv

Что нужно для решения такой задачи?

- Какая нужна статистика?
 - Проверка гипотезы об ассоциации. Точный тест Фишера, χ^2
- Как посчитать что-то для всех строк таблицы?
 - Условия и циклы (*if, for*)
 - Функции, применение их к элементам вектора, рядам и колонкам матрицы
- Как объединить две таблицы?
 - SQL-подобные операции с таблицами: merge, aggregate



Точный тест Фишера

χ^2

ПРОВЕРКА ГИПОТЕЗЫ ОБ АССОЦИАЦИИ

Тесты ассоциации

У некоторых объектов есть два свойства, про которые, в простейшем случае, можно сказать «да» или «нет», например, больной – здоровый, носитель мутации – не носитель, курит – не курит. Связаны ли эти свойства?

Примеры задач:

- Ассоциирована ли определенная мутация с болезнью
- Верно ли, что на биофаке значимо большая доля девушек, чем на ВМК?

Тесты ассоциации. Мутация, ассоциированная с болезнью

```
> gwas=matrix(c(3324,1896,2676,2104), nrow=2, ncol=2)
```

```
> gwas
```

```
      [,1] [,2]  
[1,] 3324 2676  
[2,] 1896 2104
```

```
> colnames(gwas)=c("reference", "mutant")
```

```
> rownames(gwas)=c("Healthy", "Diseased")
```

```
> gwas
```

	reference	mutant	
Healthy	3324	2676	← Здоровый (контроль)
Diseased	1896	2104	← Больной

Таблица сопряженности

↑
Не-носитель мутации

↑
Носитель мутации*

* Пояснение «для биологов»:
подразумеваем, например, что носитель мутации = носитель болезни **8**
(т.е. мутантный аллель - доминантный)

Тест Фишера и χ^2

> `fisher.test(gwas)`

Fisher's Exact Test for Count Data

data: gwas

p-value = 4.874e-15

alternative hypothesis: true odds ratio is not equal to 1

95 percent confidence interval:

1.271049 1.494857

sample estimates:

odds ratio

1.378354

> `chisq.test(gwas)`

Pearson's Chi-squared test with Yates' continuity correction

data: gwas

X-squared = 61.239, df = 1, **p-value = 5.055e-15**

Тест Фишера и χ^2

- Другой (иногда более удобный) способ – передавать не четырехпольные таблицы, а два логических вектора, соответствующие двум свойствам, между которыми ищем ассоциацию

```
> people=data.frame(mutant=c(T,F,F,F,T,T,F,T),  
diseased=c(T,F,F,F,F,T,T,T))
```

```
> fisher.test(people$mutant, people$diseased)
```

- Удобный способ получить 4-польную таблицу:

```
> xtabs(~ mutant + diseased, data=people)
```

	diseased	
Mutant	FALSE	TRUE
FALSE	3	1
TRUE	1	3

Точный тест Фишера

	не мутант	мутант	ВСЕГО
здоровый	a	b	a+b
больной	c	d	c+d
ВСЕГО	a+c	b+d	n

Отношение рисков (odds ratio)

$$OR = \frac{P(\text{больной}|\text{мутант})}{P(\text{здоровый}|\text{мутант})} \bigg/ \frac{P(\text{больной}|\text{не мутант})}{P(\text{здоровый}|\text{не мутант})} = \frac{d}{b} \bigg/ \frac{c}{a} = \frac{da}{bc}$$

Точный тест Фишера (Fisher exact test):

Сколько способов разложить n независимых элементов по 4 ячейкам с данным количеством элементов? p-value – сумма вероятностей таблиц с не меньшим OR.

Тест Хи² (Chi-squared)

- Не рекомендуется использовать, если хотя бы в одной ячейке меньше 5-7 образцов
- **H₀**: Предположение независимости. Например, $p(\text{больной} \ \& \ \text{мутант}) = p(\text{больной}) * p(\text{мутант})$

Тест Хи²:

Посчитаем значение в каждой ячейке в предположении независимости, это будет ожидаемое значение.

Далее по всем ячейкам суммируем

$$\chi^2 = \sum \frac{(O - E)^2}{E} \sim \chi^2_{\text{d.f.}=(nrow-1)(ncol-1)}$$



матрицы

списки

СТРУКТУРЫ ДАННЫХ (КОНТЕЙНЕРЫ)

Матрицы

- На вид трудно отличить от data frame
- «Двумерный вектор»
- Все элементы одного типа

Матрицы заполняются по колонкам:

```
> m <- matrix(c(1:6), nrow=2, ncol=3)
```

```
> m
```

```
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Или по рядам (аргумент **byrow=T**)

```
> m <- matrix(c(1:6), nrow=2, ncol=3, byrow=T)
```

```
> m
```

```
      [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

```
> as.matrix(1:3)
```

```
      [,1]
[1,]    1
[2,]    2
[3,]    3
```

Матрицы

- Обращение к элементам – как в data frame

```
> m[2, 3]
```

```
[1] 6
```

```
> m[2, ]
```

```
[1] 2 4 6
```

```
> m[, 3]
```

```
[1] 5 6
```

```
> m[2, 3] = 1
```

```
> m
```

	[,1]	[,2]	[,3]
[1,]	1	3	5
[2,]	2	4	6

Матрицы

```
> m <- matrix(nrow=2, ncol=3)
```

```
> m
```

```
      [,1] [,2] [,3]
[1,]  NA  NA  NA
[2,]  NA  NA  NA
```

```
> dim(m) ← Атрибут dim – размер матрицы
[1] 2 3      См. также nrow, ncol
```

```
> attributes(m)
```

```
$dim
```

```
[1] 2 3
```


Матрицы

Матрица – это вектор с атрибутом *dim*

```
> v <- 1:6
```

```
> dim(v)      количество строк
```

```
NULL
```

← количество столбцов

```
> dim(v) <- c(2, 3)
```

```
> v
```

	[, 1]	[, 2]	[, 3]
[1,]	1	3	5
[2,]	2	4	6

```
> is.matrix(v)
```

```
[1] TRUE
```

Матрицы

```
> a <- c(1:3)
```

```
> b <- c(11:13)
```

```
> cbind(a, b) # “column bind”
```

```
      a  b  
[1,] 1 11  
[2,] 2 12  
[3,] 3 13
```

```
> rbind(a, b) # “row bind”
```

```
  [,1] [,2] [,3]  
a     1     2     3  
b    11    12    13
```

Матрицы

Присваивание имен столбцам и колонкам:

```
> m <- matrix(c(1:6), nrow=2, ncol=3)
```

```
> m
```

```
      [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> rownames(m) <- c("r1", "r2")
```

```
> colnames(m) <- c("c1", "c2", "c3")
```

```
> m
```

```
      c1 c2 c3
r1     1  3  5
r2     2  4  6
```

Списки

- Список, в отличие от вектора, может хранить элементы разного типа
 - В частности, элементом может быть список

```
> L = list("A", c(1,2), 30)
```

```
Сравните: > v=c(v, 4)
```

```
> L1 = list(L, 40)
```

```
#->list(list("A",c(1,2),30),40)
```

```
> L [[ 4 ]] = 4
```

```
> L [[ length( L ) + 1 ]] = 4
```

```
> L
```

```
[[1]]
```

```
[1] "A"
```

```
[[2]]
```

```
[1] 1 2
```

```
[[3]]
```

```
[1] 30
```

список
из 3
элементов

Каждый элемент –
вектор (в R 1 элемент
– вектор из самого себя)

СПИСКИ

- Как достать элемент списка

```
> L [ 3 ] #-> список (из 1 элемента)
```

```
[[1]]  
[1] 30
```

Правило: одинарные скобки []
возвращают элемент того же типа,

```
> L [[ 3 ]]
```

что и контейнер

```
#элемент (= вектор из 1 элемента)
```

```
[1] 30
```

- Как достать элемент из списка списков?

```
> L=list(10,list(11,12,13), 20, list(30))
```

```
#L[[2]] - тоже список
```

```
> L[[2]][[3]]
```

```
[1] 13
```

СПИСКИ

Сравните с выдачей любого теста, например t.test

■ Именованные элементы

```
> L=list(10,20)
```

```
> L$abc=123
```

```
> L
```

```
[[1]]
```

```
[1] 10
```

```
[[2]]
```

```
[1] 20
```

```
$abc
```

```
[1] 123
```

```
> names(L)
```

```
[1] "" "" "abc"
```

```
> L[[3]]
```

```
[1] 123
```

```
> L$abc=123
```

```
[1] 123
```

```
> L$abc
```

```
[1] 123
```

```
> L[["abc"]]
```

```
[1] 123
```

Повторение: векторы, списки, матрицы и data frames

	элементы одного типа	элементы разных типов
1D	<p>Вектор</p> <pre>>v = c(1,2,3) >v = c(v, 4) >v [2] #-> вектор (из 1 элемента)</pre>	<p>Список</p> <pre>> L = list(1,2,3) > L1 = list(L, 4) #-> list(list(1,2,3), 4) > L [[length(L) + 1]] = 4 > L [2] #-> список (из 1 элемента) > L [[2]] #элемент (=вектор из 1 элемента)</pre>
2D	<p>Матрица “2D вектор”</p> <pre>>m=matrix(c(1,2,3,4), nrow=2) >m[,1] #колонка матрицы - вектор >m[2,] #ряд матрицы – вектор</pre>	<p>data frame “список векторов (разных типов)”</p> <pre>>df=data.frame(name=c(“A”, “B”, “C”), grade=c(5,5,4)) >df[,1] #колонка d.f- вектор >df[1,] #ряд d.f- тоже data frame (как и d.f, является списком). Почему не вектор?</pre>
nD	<p>array “n-мерный вектор”</p>	



Условия и циклы

Функции и apply

«ПРОГРАММИРОВАНИЕ»

Цикл *for*

- Пример из домашнего задания.

```
N_iter=20
```

```
y=rep(NA, N_iter)
```

```
for(i in 1:N_iter){
```

```
  rands=rnorm(2^i)
```

```
  y[i]=max(rands)
```

```
}
```

```
plot(2^(1:N_iter), y, type="l")
```

Вектор (в данном случае от 1 до N_iter)
i проходит все элементы вектора
Для каждого i выполняется код в { }

- Совет: многострочные исходные коды удобно сохранять в файл, читать и выполнять из R:

```
> source("my_program.R")
```

for. Пример для таблицы

```
> head(grades)
```

```
   id write math science socst
1  70   52  41    47    57
2 121   59  53    63    61
3  86   33  54    58    31
```

#посчитаем средний балл каждого студента и средний балл за каждый экзамен

```
> matrix_grades=as.matrix(grades[,2:ncol(grades)])
```

```
> meangrades=c()
```

```
> for(i in 1:nrow(grades)){
```

```
+   meangrades=c(meangrades, mean(matrix_grades[i,]))
```

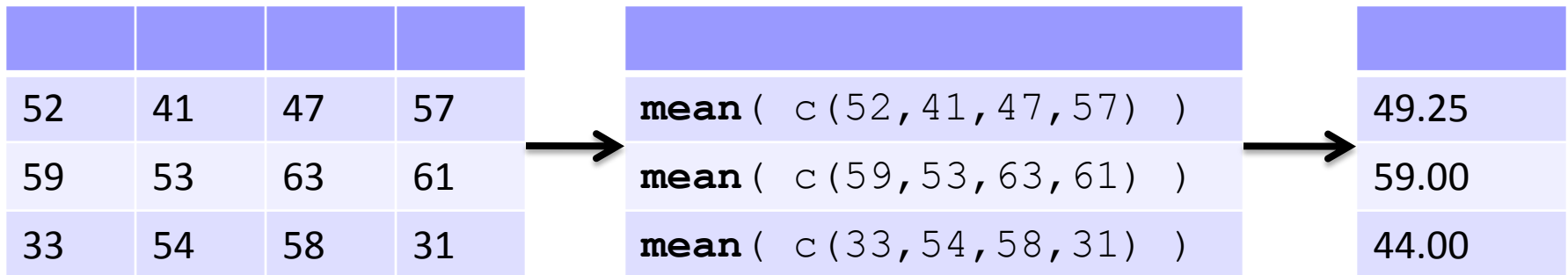
```
+ }
```

```
> head(meangrades)
```

```
[1] 49.25 59.00 44.00 50.00 55.75 56.75
```

apply. Пример для таблицы

Применить функцию к каждому ряду или колонке data frame или элементу вектора.



Матрица или data frame

Функция

#средний балл каждого студента

```
> meangrades=apply(grades[,2:ncol(grades)], 1, mean)
```

```
> head(meangrades)
```

```
[1] 49.25 59.00 44.00 50.00 55.75 56.75
```

Измерение (1-строки или 2-столбцы)

#средний балл по каждому экзамену

```
> examgrades=apply(grades[,2:ncol(grades)], 2, mean)
```

```
> head(examgrades)
```

```
write    math science    socst
52.775   52.645   51.850   52.405
```

Функции – использование

- Имя функции со скобками – её вызов, в скобках аргументы. Функция может возвращать значение.

`tt=t.test(a, b, paired=T)`  Именованный аргумент

- Имя функции без скобок можно использовать как имя переменной – посмотреть содержимое или передать как аргумент другой функции

`> apply`

```
function (X, MARGIN, FUN, ...) {
```

...

- Помимо стандартных функций (например, *mean*, *all*, *any*) можно передать *apply* свою функцию

Функции – создание

Создадим функцию, которая берет на вход вектор из 4 чисел, создает четырехпольную таблицу, выполняет тест Фишера или χ^2

```
association.test <- function(v){  
  m=matrix(v, nrow=2, ncol=2) #создаем матрицу  
  res=fisher.test(m)  
  return(res$p.value) ИЛИ res$p.value  
}
```

обычное присваивание

Всегда возвращается последнее вычисленное значение, даже если не писать return

Функции – создание. *if* и аргумент по умолчанию

```
association.test <- function(v, test="fisher"){  
  m=matrix(v, nrow=2, ncol=2) #создаем матрицу  
  if (test=="fisher"){  
    res=fisher.test(m)  
  } else if(test=="chi2"){  
    res=chisq.test(m)  
  } else { return (NA) }  
  return(res$p.value) ИЛИ res$p.value  
}
```

Значение по умолчанию
↓

условие

Другие разновидности *apply*

- Если на входе список (list), то `lapply`, `sapply`

```
> l=list(1, c(1,2,3), c(2,1))
```

```
> lapply(l, sum)
```

```
[[1]]
```

```
[1] 1
```

#список, каждый элемент – результат работы функции

```
[[2]]
```

```
[1] 6
```

```
[[3]]
```

```
[1] 3
```

```
> sapply(l, sum)
```

```
[1] 1 6 3
```

См. также `unlist`

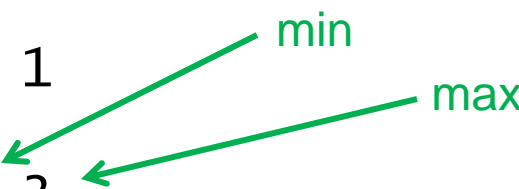
#Пытается «упростить» список. В данном случае свёл к вектору

Другие разновидности *apply*

- Более содержательные пример – а пусть функция выдает не один элемент, а вектор из двух

```
> lapply(1, function(x){return(c(min(x), max(x)))})
```


```
[[1]]  
[1] 1 1  
[[2]]  
[1] 1 3  
[[3]]  
[1] 1 2
```



The diagram shows two green arrows. One arrow points from the word 'min' to the second element '1' of the second row. The other arrow points from the word 'max' to the third element '3' of the second row.

```
> sapply(1, function(x){return(c(min(x), max(x)))})
```

```
      [,1] [,2] [,3]  
[1,]    1    1    1  
[2,]    1    3    2
```



The diagram shows two green arrows. One arrow points from the word 'min' to the second column of the matrix. The other arrow points from the word 'max' to the third column of the matrix.

#В данном случае «упростил»
до матрицы

Вернемся к мутациям

- Уже можем посчитать для каждой позиции (строки в таблице) p-value ассоциации соответствующего варианта с расой

```
apply(____, ____, association.test, test="fisher")
```

- Подсказка: все **аргументы**, которые apply не распознала как свои, передаются вызываемой функции
- Осталось понять, каким генам соответствуют мутации (сделать из двух таблиц одну)



merge

aggregate, tapply

SQL-ПОДОБНЫЕ ОПЕРАЦИИ С ТАБЛИЦАМИ

merge



Студент	Факультет	Курс	Пол	КР1	КР2	КР3
Иванов						

merge

- Пример: 2 таблицы
 - id -> оценки,
 - id -> метаданные студента
- Объединить таблицы в одну по идентификатору студента

```
> head(grades)
```

	id	write	math	science	socst
1	70	52	41	47	57
2	121	59	53	63	61
3	86	33	54	58	31
4	141	44	47	53	56
5	172	52	57	53	61
6	113	52	51	63	61

```
> head(metadata)
```

	id	female	race	schttyp	prog
1	1	1	1	1	3
2	2	1	1	1	3
3	3	0	1	1	2
4	4	1	1	1	2
5	5	0	1	1	2
6	6	1	1	1	2

merge

```
> meta_grades=merge(metadata, grades,  
  by.x=c(1), by.y=c(1))
```

```
> head(meta_grades)
```

	id	female	race	schtyp	prog	write	math	science	socst
1	1	1	1	1	3	44	40	39	41
2	2	1	1	1	3	41	33	42	41
3	3	0	1	1	2	65	48	63	56
4	4	1	1	1	2	50	41	39	51
5	5	0	1	1	2	40	43	45	31
6	6	1	1	1	2	41	46	40	41

```
> meta_grades=merge(metadata,  
  grades, by=c("id"))
```

```
> head(meta_grades)
```

	id	female	race	schtyp	prog	write	math	science	socst
1	1	1	1	1	3	44	40	39	41
2	2	1	1	1	3	41	33	42	41
3	3	0	1	1	2	65	48	63	56
4	4	1	1	1	2	50	41	39	51
5	5	0	1	1	2	40	43	45	31
6	6	1	1	1	2	41	46	40	41

```
> meta_grades=merge(metadata, grades,  
  by=c("id"), all.x=F, all.y=T)
```

by может содержать несколько ЭЛЕМЕНТОВ

```
> head(grades)
```

	id	write	math	science	socst
1	70	52	41	47	57
2	121	59	53	63	61
3	86	33	54	58	31
4	141	44	47	53	56
5	172	52	57	53	61
6	113	52	51	63	61

```
> head(metadata)
```

	id	female	race	schtyp	prog
1	1	1	1	1	3
2	2	1	1	1	3
3	3	0	1	1	2
4	4	1	1	1	2
5	5	0	1	1	2
6	6	1	1	1	2



aggregate

- Объединение рядов с одинаковыми значениями определенных столбцов

id студента	Факультет	Оценка
1	ФББ	5
2	ФББ	4
3	ВМК	5
4	ВМК	5
5	ВМК	4
6	Биофак	5
7	Биофак	4

↓
↑
↓
↑
↓
↑

mean(c(5,5,4))

Group	Оценка
ФББ	4.5
ВМК	4.67
Биофак	4.5

aggregate

```
> mean_race=aggregate(meta_grades[,6:9],  
  by=list(meta_grades$race), FUN=mean)
```

```
> mean_race
```

	Group.1	write	math	science	socst
1	1	46.45833	47.41667	45.37500	47.79167
2	2	58.00000	57.27273	51.45455	51.00000
3	3	48.20000	46.75000	42.80000	49.45000
4	4	54.05517	53.97241	54.20000	53.68276

```
> mean_gender=aggregate(meta_grades[,6:9],  
  by=list(meta_grades$female), FUN=mean)
```

```
> mean_gender
```

	Group.1	write	math	science	socst
1	0	50.12088	52.94505	53.23077	51.79121
2	1	54.99083	52.39450	50.69725	52.91743

aggregate

```
> mean_gender_race=aggregate(meta_grades[,6:9],  
  by=list(meta_grades$female,  
  meta_grades$race), FUN=mean)
```

```
> mean_gender_race
```

	Group.1	Group.2	write	math	science	socst
1	0	1	44.38462	49.23077	45.53846	45.69231
2	1	1	48.90909	45.27273	45.18182	50.27273
3	0	2	55.66667	58.66667	53.00000	47.66667
4	1	2	58.87500	56.75000	50.87500	52.25000
5	0	3	47.00000	45.28571	46.71429	49.00000
6	1	3	48.84615	47.53846	40.69231	49.69231
7	0	4	51.29412	54.19118	55.38235	53.42647
8	1	4	56.49351	53.77922	53.15584	53.90909

tapply

- Похожая идея, но для векторов
- На входе – **вектор** (а не таблица) и список (длины N) из векторов факторов той же длины
- На выходе – N-мерный вектор

```
> mean_race=tapply(meta_grades$math,  
list(meta_grades$race), FUN=mean)
```

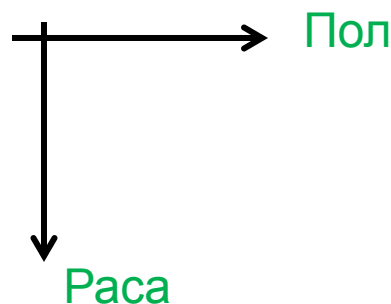
```
> mean_race
```

```
      1          2          3          4 ← раса  
47.41667 57.27273 46.75000 53.97241
```

```
> mean_race=tapply(meta_grades$math,  
list(meta_grades$race, meta_grades$female),  
FUN=mean)
```

```
> mean_race
```

```
      0          1  
1 49.23077 45.27273  
2 58.66667 56.75000  
3 45.28571 47.53846  
4 54.19118 53.77922
```



Ещё раз про домашнее задание

variant_counts.csv

Для каждой строки посчитать:
добавить как столбцы к data frame

chr	pos	EA_ref	EA_alt	AA_ref	AA_alt
chr6	292452	4028	27	2089	12
chr6	292461	4100	1	2117	0

P-value Fisher	P-value X2	Odds ratio

...

Объединить таблицы
по 2 общим полям

chr	pos	genes
chr6	292452	DUSP22
chr6	292461	DUSP22

... *variant_genes.csv*

...	gene	...	P-value
	DUSP22		1e-10
	DUSP22		1e-5
	HLA-A		1e-13
	HLA-A		1e-10

...	gene	...	P-value
	DUSP22		1e-10
	HLA-A		1e-13

Группировать
по генам, взяв
min p-value



```
> students[students$female==1,]
```

С 8 марта!

