

Anayeli Cocoltzi

The screenshot shows the RStudio IDE interface. The main editor window displays an R script with the following code:

```
1 #rm(lm) con la una función entonces rm la borra de la memoria
2 #funcion de r (regresión lineal)
3 search()
4 #te muestra todos lo paquetes que tienes, y busca funciones de manera de jerarqu
5
6 - hacer.potencia <- function(n){
7   potencia <- function(x){
8     x^n
9   }
10  potencia
11 }
12 cubica <- hacer.potencia(3)
13 cuadrada <- hacer.potencia(2)
14
15 cubica(3)
16 cuadrada(3)
17 ls(environment(cubica))
18 get("n",environment(cubica))
19
20
```

The Environment pane on the right shows the Global Environment with the following objects:

Object	Value
moneda	1L
name	"foo"
old.dir	"C:/Users/naye_coco/Documents"
s	List of 5
u	log1 [1:6] FALSE FALSE TRUE TRUE FALSE FALSE
x	9

The Console pane at the bottom shows the following output:

```
100%
|
| In this lesson, you learned how to examine your R workspace and work with the file
| system
| of your machine from within R. Thanks for playing!
...
100%
|
| would you like to receive credit for completing this course on coursera.org?
1: No
2: Yes
```

The R Documentation pane on the right shows the documentation for the `files2` function, titled "Manipulation of Directories and File Permissions".

The screenshot displays the RStudio interface with the following components:

- Source Editor (Top Left):** Contains an R script named `funciones(apply).R`. The script includes:


```
##split
library(datasets)
head(airquality)
print(airquality)
f <- airquality[,5]
split(airquality,f)

#o bien lo podemos ver:
s <- split(airquality,airquality$month)
s
lapply(s, function(x) colMeans(x[1:3],na.rm = T))
#pero como todos los resultados son del mismo tamaño podemos aplicar sapply
sapply(s,function(x) colMeans(x[,1:4],na.rm = T))
```
- Console (Bottom Left):** Shows the execution output:


```
[1] 3.478505 3.181981 2.146460
```

 It also displays a progress bar at 100% and a message: "You've reached the end of this lesson! Return to the main menu."
- Environment Pane (Top Right):** Lists the objects in the Global Environment:

Object	Class	Value
<code>my_d1v</code>	num	[1:3] 3.48 3.18 2.15
<code>my_sqrt</code>	num	[1:3] 0.316 2.828 1.463
<code>name</code>	chr	"foo"
<code>s</code>	List of 5	
<code>u</code>	logi	[1:6] FALSE FALSE TRUE TRUE FALSE FALSE
<code>x</code>	int	12
- Documentation Pane (Bottom Right):** Displays the documentation for the `c()` function, titled "Combine Values into a Vector or List".

Description: This is a generic function which combines its arguments.

The default method combines its arguments to form a vector. All arguments are coerced to a common type which is the type of the returned value, and all attributes except names are removed.

Usage: `c(..., recursive = FALSE)`

Arguments: `...` objects to be concatenated.

`recursive` logical. If `recursive = TRUE`, the function recursively descends through lists (and parlists) combining all their elements into a vector.

Anayeli Cocoltzi

The screenshot shows the RStudio IDE interface. The main editor window displays an R script with the following code:

```
1 #rm(lm) con lm una función entonces rm la borra de la memoria
2 #función de r (regresión lineal)
3 search()
4 #te muestra todos los paquetes que tienes, y busca funciones de manera de jerarquía
5
6 hacer.potencia <- function(n){
7   potencia <- function(x){
8     x^n
9   }
10  potencia
11 }
12 cubica <- hacer.potencia(3)
13 cuadrada <- hacer.potencia(2)
14
15 cubica(3)
16 cuadrada(3)
17 ls(environment(cubica))
18 get("n", environment(cubica))
19
20
```

The console window shows the following output:

```
| All that hard work is paying off!
|-----| 96%
| Finally, let's say that rather than repeating the vector (0, 1, 2) over and
| over again, we want our vector to contain 10 zeros, then 10 ones, then 10 twos.
| we can do this with the 'each' argument. Try rep(c(0, 1, 2), each = 10).
> rep(c(0,1,2),each=10)
[1] 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
| Excellent work!
|-----| 100%
| would you like to receive credit for completing this course on Coursera.org?
```

The R Documentation window shows the "Colon Operator" page, which includes the following information:

- Description:** Generate regular sequences.
- Usage:**

```
from:to
a:b
```
- Arguments:**
 - `from`: starting value of sequence.
 - `to`: (maximal) end value of the sequence.
 - `a`, `b`: [factors](#) of the same length.

File Edit Code View Plots Session Build Debug Tools Help

Project: (None)

Global Environment

Object	Class	Attributes
my_char	chr [1:3]	"My" "name" "is"
my_name	chr [1:4]	"My" "name" "is" "Naye"
my_seq	num [1:30]	5 5.17 5.34 5.52 5.69 ...
name	chr [1:1]	"foo"
num_vect	num [1:4]	0.5 55 -10 6

Files Plots Packages Help Viewer

R Colon Operator

Colon Operator

Description

Generate regular sequences.

Usage

from:to
a:b

Arguments

from: starting value of sequence.
to: (maximal) end value of the sequence.
a, b: **factors** of the same length.

```

1 #rm(lm) con lm una función entonces rm la borra de la memoria
2 #funcion de r (regresión lineal)
3 search()
4 #te muestra todos lo paquetes que tienes, y busca funciones de manera de jerarquía
5
6 - hacer.potencia <- function(n){
7   potencia <- function(x){
8     x^n
9   }
10  potencia
11 }
12 cubica <- hacer.potencia(3)
13 cuadrada <- hacer.potencia(2)
14
15 cubica(3)
16 cuadrada(3)
17 ls(environment(cubica))
18 get("n",environment(cubica))
19
20

```

Console

```

> 3, and 4 in the output above are no longer numbers to R, but rather characters "3",
"4",
"3", and "4".
...
==> 100%
| would you like to receive credit for completing this course on Coursera.org?
1: No
2: Yes
Selection:

```

Windows Taskbar: 10:00 p.m. 03/10/2016

File Edit Code View Plots Session Build Debug Tools Help

Project: (None)

Global Environment

Object	Class	Attributes
a	dbl	48L
b	dbl	28
c	dbl	1
completos	logi [1:153]	TRUE TRUE TRUE TRUE FALSE FALSE ...
con	Classes 'url', 'connection' atomic [1:1]	

Files Plots Packages Help Viewer

R: Finite, Infinite and NaN Numbers

Finite, Infinite and NaN Numbers

Description

is.finite and is.infinite return a vector of the same length as x, indicating which elements are finite (not infinite and not missing) or infinite.

Inf and -Inf are positive and negative infinity whereas NA means 'Not a Number'. (These apply to numeric values and real and imaginary parts of complex values but not to values of integer vectors.) Inf and NA are **reserved** words in the R language.

Usage

is.finite(x)
is.infinite(x)
is.nan(x)

Inf
NA

```

1 #rm(lm) con lm una función entonces rm la borra de la memoria
2 #funcion de r (regresión lineal)
3 search()
4 #te muestra todos lo paquetes que tienes, y busca funciones de manera de jerarquía
5
6 - hacer.potencia <- function(n){
7   potencia <- function(x){
8     x^n
9   }
10  potencia
11 }
12 cubica <- hacer.potencia(3)
13 cuadrada <- hacer.potencia(2)
14
15 cubica(3)
16 cuadrada(3)
17 ls(environment(cubica))
18 get("n",environment(cubica))
19
20

```

Console

```

> 7Inf
> Inf- Inf
[1] NaN
| keep working like that and you'll get there!
==> 100%
| would you like to receive credit for completing this course on Coursera.org?
1: No
2: Yes
Selection:

```

Windows Taskbar: 10:16 p.m. 03/10/2016

File Edit Code View Plots Session Build Debug Tools Help

Project: (None)

funciones1.R

```
1 #rm(lm) con lm una función entonces rm la borra de la memoria
2 #función de r (regresión lineal)
3 search()
4 #te muestra todos los paquetes que tienes, y busca funciones de manera de jerarquía
5
6 hacer.potencia <- function(n){
7   potencia <- function(x){
8     x^n
9   }
10  potencia
11 }
12 cubica <- hacer.potencia(3)
13 cuadrada <- hacer.potencia(2)
14
15 cubica(3)
16 cuadrada(3)
17 ls(environment(cubica))
18 get("n", environment(cubica))
19
20 <
```

Console

```
| Now you know all four methods of subsetting data from vectors. Different approaches
| are
| best in different scenarios and when in doubt, try it out!
...
=====
--> 100%

| Would you like to receive credit for completing this course on Coursera.org?
1: No
2: Yes
Selection: |
```

Environment

Global Environment	
moneda	1L
name	"foo"
s	List of 5
u	logi [1:6] FALSE FALSE TRUE TRUE FALSE FALSE
vect	Named num [1:3] 11 2 NA
vect2	Named num [1:3] 11 2 NA

Files Plots Packages Help Viewer

R: Finite, Infinite and NaN Numbers

Finite, Infinite and NaN Numbers

Description

`is.finite` and `is.infinite` return a vector of the same length as `x`, indicating which elements are finite (not infinite and not missing) or infinite.

`Inf` and `-Inf` are positive and negative infinity whereas `NaN` means 'Not a Number'. (These apply to numeric values and real and imaginary parts of complex values but not to values of integer vectors.) `Inf` and `NaN` are [reserved](#) words in the R language.

Usage

```
is.finite(x)
is.infinite(x)
is.nan(x)
```

Inf
NaN

06:31 p.m. 06/10/2016

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Project: (None)

funciones(appl).R x Clase Agosto_29.R x

Source on Save Go to file/function Addins

Run Source

```
1 # Clase de 25/08/2016
2
3 #Crear vectores
4 x <- vector(mode = "numeric", length = 5L)
5 x
6
7 #Crear vectores con la función c
8 x <- c(.05,.06)
9 x
10 class(x)
11
12 x <- c(TRUE, FALSE,T,F)
13 x
14 class(x)
15
16 x <- 10:0
17 x
18 class(x)
19
20 <
```

Environment History

Global Environment

my_data 4 obs. of 6 variables
my_matr1 int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
my_matrix int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
my_matrix2 int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
my_vector int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
nueva.y 1 obs. of 2 variables

Files Plots Packages Help Viewer

R Matrices Find in Topic

matrix (base) R Documentation

Matrices

Description

matrix creates a matrix from the given set of values.
as.matrix attempts to turn its argument into a matrix.
is.matrix tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, mcol = 1, byrow = FALSE,  
        dimnames = NULL)  
  
as.matrix(x, ...)  
## S3 method for class 'data.frame'  
as.matrix(x, rownames.force = NA, ...)
```

Anayeli Cocoltzi

Console

...
==> 100%
...
| would you like to receive credit for completing this course on coursera.org?
1: No
2: Yes
selection:
Enter an item from the menu, or 0 to exit
Selection: |

Windows taskbar: 01:54 p.m. 09/10/2016

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Project: (None)

funciones(appl).R x Clase Agosto_29.R x

Source on Save Go to file/function Addins

Run Source

```
1 # Clase de 25/08/2016
2
3 #Crear vectores
4 x <- vector(mode = "numeric", length = 5L)
5 x
6
7 #Crear vectores con la función c
8 x <- c(.05,.06)
9 x
10 class(x)
11
12 x <- c(TRUE, FALSE,T,F)
13 x
14 class(x)
15
16 x <- 10:0
17 x
18 class(x)
19
20 <
```

Environment History

Global Environment

my_data 4 obs. of 6 variables
my_matr1 int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
my_matrix int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
my_matrix2 int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
my_vector int [1:4, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
nueva.y 1 obs. of 2 variables

Files Plots Packages Help Viewer

R Matrices Find in Topic

matrix (base) R Documentation

Matrices

Description

matrix creates a matrix from the given set of values.
as.matrix attempts to turn its argument into a matrix.
is.matrix tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, mcol = 1, byrow = FALSE,  
        dimnames = NULL)  
  
as.matrix(x, ...)  
## S3 method for class 'data.frame'  
as.matrix(x, rownames.force = NA, ...)
```

Anayeli Cocoltzi

Console

...
==> 100%
...
| would you like to receive credit for completing this course on coursera.org?
1: No
2: Yes
selection:
Enter an item from the menu, or 0 to exit
Selection: |

Windows taskbar: 01:54 p.m. 09/10/2016

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Project: (None)

Global Environment

Variable	Value
firts	firt [1:10] 3 6 7 5 4 9 2 1 10 8
letra	"d"
moneda	1L
my_vector	firt [1:20] 1 2 3 4 5 6 7 8 9 10 ...
name	"foo"
norma11	num [1:10000] -0.82 0.487 0.738 0.576 -0.305 ...

R Matrices

matrix (base)

R Documentation

Matrices

Description

matrix creates a matrix from the given set of values.

as.matrix attempts to turn its argument into a matrix.

is.matrix tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, mcol = 1, byrow = FALSE,
        dimnames = NULL)

as.matrix(x, ...)
## S3 method for class 'data.frame'
as.matrix(x, rownames.force = NA, ...)
```

Console

```
1 # You're about to write your first function! Just like you would assign a value
2 # to a variable with the assignment operator, you assign functions in the follow
3 # way:
4 #
5 # function_name <- function(arg1, arg2){
6 #   # Manipulate arguments in some way
7 #   # Return a value
8 # }
9 #
10 # The "variable name" you assign will become the name of your function. arg1 and
11 # arg2 represent the arguments of your function. You can manipulate the argument
12 # you specify within the function. After sourcing the function, you can use the
13 # function by typing:
14 #
15 # function_name(value1, value2)
16 #
17 # Below we will create a function called boring_function. This function takes
18 # the argument 'x' as input, and returns the value of x without modifying it.
19 # Delete the pound sign in front of the x to make the function work! Be sure to
20 #
21 #
22 #
23 #
24 #
25 #
26 #
27 #
28 #
29 #
30 #
31 #
32 #
33 #
34 #
35 #
36 #
37 #
38 #
39 #
40 #
41 #
42 #
43 #
44 #
45 #
46 #
47 #
48 #
49 #
50 #
51 #
52 #
53 #
54 #
55 #
56 #
57 #
58 #
59 #
60 #
61 #
62 #
63 #
64 #
65 #
66 #
67 #
68 #
69 #
70 #
71 #
72 #
73 #
74 #
75 #
76 #
77 #
78 #
79 #
80 #
81 #
82 #
83 #
84 #
85 #
86 #
87 #
88 #
89 #
90 #
91 #
92 #
93 #
94 #
95 #
96 #
97 #
98 #
99 #
100 #
101 #
102 #
103 #
104 #
105 #
106 #
107 #
108 #
109 #
110 #
111 #
112 #
113 #
114 #
115 #
116 #
117 #
118 #
119 #
120 #
121 #
122 #
123 #
124 #
125 #
126 #
127 #
128 #
129 #
130 #
131 #
132 #
133 #
134 #
135 #
136 #
137 #
138 #
139 #
140 #
141 #
142 #
143 #
144 #
145 #
146 #
147 #
148 #
149 #
150 #
151 #
152 #
153 #
154 #
155 #
156 #
157 #
158 #
159 #
160 #
161 #
162 #
163 #
164 #
165 #
166 #
167 #
168 #
169 #
170 #
171 #
172 #
173 #
174 #
175 #
176 #
177 #
178 #
179 #
180 #
181 #
182 #
183 #
184 #
185 #
186 #
187 #
188 #
189 #
190 #
191 #
192 #
193 #
194 #
195 #
196 #
197 #
198 #
199 #
200 #
201 #
202 #
203 #
204 #
205 #
206 #
207 #
208 #
209 #
210 #
211 #
212 #
213 #
214 #
215 #
216 #
217 #
218 #
219 #
220 #
221 #
222 #
223 #
224 #
225 #
226 #
227 #
228 #
229 #
230 #
231 #
232 #
233 #
234 #
235 #
236 #
237 #
238 #
239 #
240 #
241 #
242 #
243 #
244 #
245 #
246 #
247 #
248 #
249 #
250 #
251 #
252 #
253 #
254 #
255 #
256 #
257 #
258 #
259 #
260 #
261 #
262 #
263 #
264 #
265 #
266 #
267 #
268 #
269 #
270 #
271 #
272 #
273 #
274 #
275 #
276 #
277 #
278 #
279 #
280 #
281 #
282 #
283 #
284 #
285 #
286 #
287 #
288 #
289 #
290 #
291 #
292 #
293 #
294 #
295 #
296 #
297 #
298 #
299 #
300 #
301 #
302 #
303 #
304 #
305 #
306 #
307 #
308 #
309 #
310 #
311 #
312 #
313 #
314 #
315 #
316 #
317 #
318 #
319 #
320 #
321 #
322 #
323 #
324 #
325 #
326 #
327 #
328 #
329 #
330 #
331 #
332 #
333 #
334 #
335 #
336 #
337 #
338 #
339 #
340 #
341 #
342 #
343 #
344 #
345 #
346 #
347 #
348 #
349 #
350 #
351 #
352 #
353 #
354 #
355 #
356 #
357 #
358 #
359 #
360 #
361 #
362 #
363 #
364 #
365 #
366 #
367 #
368 #
369 #
370 #
371 #
372 #
373 #
374 #
375 #
376 #
377 #
378 #
379 #
380 #
381 #
382 #
383 #
384 #
385 #
386 #
387 #
388 #
389 #
390 #
391 #
392 #
393 #
394 #
395 #
396 #
397 #
398 #
399 #
400 #
401 #
402 #
403 #
404 #
405 #
406 #
407 #
408 #
409 #
410 #
411 #
412 #
413 #
414 #
415 #
416 #
417 #
418 #
419 #
420 #
421 #
422 #
423 #
424 #
425 #
426 #
427 #
428 #
429 #
430 #
431 #
432 #
433 #
434 #
435 #
436 #
437 #
438 #
439 #
440 #
441 #
442 #
443 #
444 #
445 #
446 #
447 #
448 #
449 #
450 #
451 #
452 #
453 #
454 #
455 #
456 #
457 #
458 #
459 #
460 #
461 #
462 #
463 #
464 #
465 #
466 #
467 #
468 #
469 #
470 #
471 #
472 #
473 #
474 #
475 #
476 #
477 #
478 #
479 #
480 #
481 #
482 #
483 #
484 #
485 #
486 #
487 #
488 #
489 #
490 #
491 #
492 #
493 #
494 #
495 #
496 #
497 #
498 #
499 #
500 #
501 #
502 #
503 #
504 #
505 #
506 #
507 #
508 #
509 #
510 #
511 #
512 #
513 #
514 #
515 #
516 #
517 #
518 #
519 #
520 #
521 #
522 #
523 #
524 #
525 #
526 #
527 #
528 #
529 #
530 #
531 #
532 #
533 #
534 #
535 #
536 #
537 #
538 #
539 #
540 #
541 #
542 #
543 #
544 #
545 #
546 #
547 #
548 #
549 #
550 #
551 #
552 #
553 #
554 #
555 #
556 #
557 #
558 #
559 #
560 #
561 #
562 #
563 #
564 #
565 #
566 #
567 #
568 #
569 #
570 #
571 #
572 #
573 #
574 #
575 #
576 #
577 #
578 #
579 #
580 #
581 #
582 #
583 #
584 #
585 #
586 #
587 #
588 #
589 #
590 #
591 #
592 #
593 #
594 #
595 #
596 #
597 #
598 #
599 #
600 #
601 #
602 #
603 #
604 #
605 #
606 #
607 #
608 #
609 #
610 #
611 #
612 #
613 #
614 #
615 #
616 #
617 #
618 #
619 #
620 #
621 #
622 #
623 #
624 #
625 #
626 #
627 #
628 #
629 #
630 #
631 #
632 #
633 #
634 #
635 #
636 #
637 #
638 #
639 #
640 #
641 #
642 #
643 #
644 #
645 #
646 #
647 #
648 #
649 #
650 #
651 #
652 #
653 #
654 #
655 #
656 #
657 #
658 #
659 #
660 #
661 #
662 #
663 #
664 #
665 #
666 #
667 #
668 #
669 #
670 #
671 #
672 #
673 #
674 #
675 #
676 #
677 #
678 #
679 #
680 #
681 #
682 #
683 #
684 #
685 #
686 #
687 #
688 #
689 #
690 #
691 #
692 #
693 #
694 #
695 #
696 #
697 #
698 #
699 #
700 #
701 #
702 #
703 #
704 #
705 #
706 #
707 #
708 #
709 #
710 #
711 #
712 #
713 #
714 #
715 #
716 #
717 #
718 #
719 #
720 #
721 #
722 #
723 #
724 #
725 #
726 #
727 #
728 #
729 #
730 #
731 #
732 #
733 #
734 #
735 #
736 #
737 #
738 #
739 #
740 #
741 #
742 #
743 #
744 #
745 #
746 #
747 #
748 #
749 #
750 #
751 #
752 #
753 #
754 #
755 #
756 #
757 #
758 #
759 #
760 #
761 #
762 #
763 #
764 #
765 #
766 #
767 #
768 #
769 #
770 #
771 #
772 #
773 #
774 #
775 #
776 #
777 #
778 #
779 #
780 #
781 #
782 #
783 #
784 #
785 #
786 #
787 #
788 #
789 #
790 #
791 #
792 #
793 #
794 #
795 #
796 #
797 #
798 #
799 #
800 #
801 #
802 #
803 #
804 #
805 #
806 #
807 #
808 #
809 #
810 #
811 #
812 #
813 #
814 #
815 #
816 #
817 #
818 #
819 #
820 #
821 #
822 #
823 #
824 #
825 #
826 #
827 #
828 #
829 #
830 #
831 #
832 #
833 #
834 #
835 #
836 #
837 #
838 #
839 #
840 #
841 #
842 #
843 #
844 #
845 #
846 #
847 #
848 #
849 #
850 #
851 #
852 #
853 #
854 #
855 #
856 #
857 #
858 #
859 #
860 #
861 #
862 #
863 #
864 #
865 #
866 #
867 #
868 #
869 #
870 #
871 #
872 #
873 #
874 #
875 #
876 #
877 #
878 #
879 #
880 #
881 #
882 #
883 #
884 #
885 #
886 #
887 #
888 #
889 #
890 #
891 #
892 #
893 #
894 #
895 #
896 #
897 #
898 #
899 #
900 #
901 #
902 #
903 #
904 #
905 #
906 #
907 #
908 #
909 #
910 #
911 #
912 #
913 #
914 #
915 #
916 #
917 #
918 #
919 #
920 #
921 #
922 #
923 #
924 #
925 #
926 #
927 #
928 #
929 #
930 #
931 #
932 #
933 #
934 #
935 #
936 #
937 #
938 #
939 #
940 #
941 #
942 #
943 #
944 #
945 #
946 #
947 #
948 #
949 #
950 #
951 #
952 #
953 #
954 #
955 #
956 #
957 #
958 #
959 #
960 #
961 #
962 #
963 #
964 #
965 #
966 #
967 #
968 #
969 #
970 #
971 #
972 #
973 #
974 #
975 #
976 #
977 #
978 #
979 #
980 #
981 #
982 #
983 #
984 #
985 #
986 #
987 #
988 #
989 #
990 #
991 #
992 #
993 #
994 #
995 #
996 #
997 #
998 #
999 #
1000 #
1001 #
1002 #
1003 #
1004 #
1005 #
1006 #
1007 #
1008 #
1009 #
1010 #
1011 #
1012 #
1013 #
1014 #
1015 #
1016 #
1017 #
1018 #
1019 #
1020 #
1021 #
1022 #
1023 #
1024 #
1025 #
1026 #
1027 #
1028 #
1029 #
1030 #
1031 #
1032 #
1033 #
1034 #
1035 #
1036 #
1037 #
1038 #
1039 #
1040 #
1041 #
1042 #
1043 #
1044 #
1045 #
1046 #
1047 #
1048 #
1049 #
1050 #
1051 #
1052 #
1053 #
1054 #
1055 #
1056 #
1057 #
1058 #
1059 #
1060 #
1061 #
1062 #
1063 #
1064 #
1065 #
1066 #
1067 #
1068 #
1069 #
1070 #
1071 #
1072 #
1073 #
1074 #
1075 #
1076 #
1077 #
1078 #
1079 #
1080 #
1081 #
1082 #
1083 #
1084 #
1085 #
1086 #
1087 #
1088 #
1089 #
1090 #
1091 #
1092 #
1093 #
1094 #
1095 #
1096 #
1097 #
1098 #
1099 #
1100 #
1101 #
1102 #
1103 #
1104 #
1105 #
1106 #
1107 #
1108 #
1109 #
1110 #
1111 #
1112 #
1113 #
1114 #
1115 #
1116 #
1117 #
1118 #
1119 #
1120 #
1121 #
1122 #
1123 #
1124 #
1125 #
1126 #
1127 #
1128 #
1129 #
1130 #
1131 #
1132 #
1133 #
1134 #
1135 #
1136 #
1137 #
1138 #
1139 #
1140 #
1141 #
1142 #
1143 #
1144 #
1145 #
1146 #
1147 #
1148 #
1149 #
1150 #
1151 #
1152 #
1153 #
1154 #
1155 #
1156 #
1157 #
1158 #
1159 #
1160 #
1161 #
1162 #
1163 #
1164 #
1165 #
1166 #
1167 #
1168 #
1169 #
1170 #
1171 #
1172 #
1173 #
1174 #
1175 #
1176 #
1177 #
1178 #
1179 #
1180 #
1181 #
1182 #
1183 #
1184 #
1185 #
1186 #
1187 #
1188 #
1189 #
1190 #
1191 #
1192 #
1193 #
1194 #
1195 #
1196 #
1197 #
1198 #
1199 #
1200 #
1201 #
1202 #
1203 #
1204 #
1205 #
1206 #
1207 #
1208 #
1209 #
1210 #
1211 #
1212 #
1213 #
1214 #
1215 #
1216 #
1217 #
1218 #
1219 #
1220 #
1221 #
1222 #
1223 #
1224 #
1225 #
1226 #
1227 #
1228 #
1229 #
1230 #
1231 #
1232 #
1233 #
1234 #
1235 #
1236 #
1237 #
1238 #
1239 #
1240 #
1241 #
1242 #
1243 #
1244 #
1245 #
1246 #
1247 #
1248 #
1249 #
1250 #
1251 #
1252 #
1253 #
1254 #
1255 #
1256 #
1257 #
1258 #
1259 #
1260 #
1261 #
1262 #
1263 #
1264 #
1265 #
1266 #
1267 #
1268 #
1269 #
1270 #
1271 #
1272 #
1273 #
1274 #
1275 #
1276 #
1277 #
1278 #
1279 #
1280 #
1281 #
1282 #
1283 #
1284 #
1285 #
1286 #
1287 #
1288 #
1289 #
1290 #
1291 #
1292 #
1293 #
1294 #
1295 #
1296 #
1297 #
1298 #
1299 #
1300 #
1301 #
1302 #
1303 #
1304 #
1305 #
1306 #
1307 #
1308 #
1309 #
1310 #
1311 #
1312 #
1313 #
1314 #
1315 #
1316 #
1317 #
1318 #
1319 #
1320 #
1321 #
1322 #
1323 #
1324 #
1325 #
1326 #
1327 #
1328 #
1329 #
1330 #
1331 #
1332 #
1333 #
1334 #
1335 #
1336 #
1337 #
1338 #
1339 #
1340 #
1341 #
1342 #
1343 #
1344 #
1345 #
1346 #
1347 #
1348 #
1349 #
1350 #
1351 #
1352 #
1353 #
1354 #
1355 #
1356 #
1357 #
1358 #
1359 #
1360 #
1361 #
1362 #
1363 #
1364 #
1365 #
1366 #
1367 #
1368 #
1369 #
1370 #
1371 #
1372 #
1373 #
1374 #
1375 #
1376 #
1377 #
1378 #
1379 #
1380 #
1381 #
1382 #
1383 #
1384 #
1385 #
1386 #
1387 #
1388 #
1389 #
1390 #
1391 #
1392 #
1393 #
1394 #
1395 #
1396 #
1397 #
1398 #
1399 #
1400 #
1401 #
1402 #
1403 #
1404 #
1405 #
1406 #
1407 #
1408 #
1409 #
1410 #
1411 #
1412 #
1413 #
1414 #
1415 #
1416 #
1417 #
1418 #
1419 #
1420 #
1421 #
1422 #
1423 #
1424 #
1425 #
1426 #
1427 #
1428 #
1429 #
1430 #
1431 #
1432 #
1433 #
1434 #
1435 #
1436 #
1437 #
1438 #
1439 #
1440 #
1441 #
1442 #
1443 #
1444 #
1445 #
1446 #
1447 #
1448 #
1449 #
1450 #
1451 #
1452 #
1453 #
1454 #
1455 #
1456 #
1457 #
1458 #
1459 #
1460 #
1461 #
1462 #
1463 #
1464 #
1465 #
1466 #
1467 #
1468 #
1469 #
1470 #
1471 #
1472 #
1473 #
1474 #
1475 #
1476 #
1477 #
1478 #
1479 #
1480 #
1481 #
1482 #
1483 #
1484 #
1485 #
1486 #
1487 #
1488 #
1489 #
1490 #
1491 #
1492 #
1493 #
1494 #
1495 #
1496 #
1497 #
1498 #
1499 #
1500 #
1501 #
1502 #
1503 #
1504 #
1505 #
1506 #
1507 #
1508 #
1509 #
1510 #
1511 #
1512 #
1513 #
1514 #
1515 #
1516 #
1517 #
1518 #
1519 #
1520 #
1521 #
1522 #
1523 #
1524 #
1525 #
1526 #
1527 #
1528 #
1529 #
1530 #
1531 #
1532 #
1533 #
1534 #
1535 #
1536 #
1537 #
1538 #
1539 #
1540 #
1541 #
1542 #
1543 #
1544 #
1545 #
1546 #
1547 #
1548 #
1549 #
1550 #
1551 #
1552 #
1553 #
1554 #
1555 #
1556 #
1557 #
1558 #
1559 #
1560 #
1561 #
1562 #
1563 #
1564 #
1565 #
1566 #
1567 #
1568 #
1569 #
1570 #
1571 #
1572 #
1573 #
1574 #
1575 #
1576 #
1577 #
1578 #
1579 #
1580 #
1581 #
1582 #
1583 #
1584 #
1585 #
1586 #
1587 #
1588 #
1589 #
1590 #
1591 #
1592 #
1593 #
1594 #
1595 #
1596 #
1597 #
1598 #
1599 #
1600 #
1601 #
1602 #
1603 #
1604 #
1605 #
1606 #
1607 #
1608 #
1609 #
1610 #
1611 #
1612 #
1613 #
1614 #
1615 #
1616 #
1617 #
1618 #
1619 #
1620 #
1621 #
1622 #
1623 #
1624 #
1625 #
1626 #
1627 #
1628 #
1629 #
1630 #
1631 #
1632 #
1633 #
1634 #
1635 #
1636 #
1637 #
1638 #
1639 #
1640 #
1641 #
1642 #
1643 #
1644 #
1645 #
1646 #
1647 #
1648 #
1649 #
1650 #
1651 #
1652 #
1653 #
1654 #
1655 #
1656 #
1657 #
1658 #
1659 #
1660 #
1661 #
1662 #
1663 #
1664 #
1665 #
1666 #
1667 #
1668 #
1669 #
1670 #
1671 #
1672 #
1673 #
1674 #
1675 #
1676 #
1677 #
1678 #
1679 #
1680 #
1681 #
1682 #
1683 #
1684 #
1685 #
1686 #
1687 #
1688 #
1689 #
1690 #
1691 #
1692 #
1693 #
1694 #
1695 #
1696 #
1697 #
1698 #
1699 #
1700 #
1701 #
1702 #
1703 #
1704 #
1705 #
1706 #
1707 #
1708 #
1709 #
1710 #
1711 #
1712 #
1713 #
1714 #
1715 #
1716 #
1717 #
1718 #
1719 #
1720 #
1721 #
1722 #
1723 #
1724 #
1725 #
1726 #
1727 #
1728 #
1729 #
1730 #
1731 #
1732 #
1733 #
1734 #
1735 #
1736 #
1737 #
1738 #
1739 #
1740 #
1741 #
1742 #
1743 #
1744 #
1745 #
1746 #
1747 #
1748 #
1749 #
1750 #
1751 #
1752 #
1753 #
1754 #
1755 #
1756 #
1757 #
1758 #
1759 #
1760 #
1761 #
1762 #
1763 #
1764 #
1765 #
1766 #
1767 #
1768 #
1769 #
1770 #
1771 #
1772 #
1773 #
1774 #
1775 #
1776 #
1777 #
1778 #
1779 #
1780 #
1781 #
1782 #
1783 #
1784 #
1785 #
1786 #
1787 #
1788 #
1789 #
1790 #
1791 #
1792 #
1793 #
1794 #
1795 #
1796 #
1797 #
1798 #
1799 #
1800 #
1801 #
1802 #
1803 #
1804 #
1805 #
1806 #
1807 #
1808 #
1809 #
1810 #
1811 #
1812 #
1813 #
1814 #
1815 #
1816 #
1817 #
1818 #
1819 #
1820 #
1821 #
1822 #
1823 #
1824 #
1825 #
1826 #
1827 #
1828 #
1829 #
1830 #
1831 #
1832 #
1833 #
1834 #
1835 #
1836 #
1837 #
1838 #
1839 #
1840 #
1841 #
1842 #
1843 #
1844 #
1845 #
1846 #
1847 #
1848 #
1849 #
1850 #
1851 #
1852 #
1853 #
1854 #
1855 #
1856 #
1857 #
1858 #
1859 #
1860 #
1861 #
1862 #
1863 #
1864 #
1865 #
1866 #
1867 #
1868 #
1869 #
1870 #
1871 #
1872 #
1873 #
1874 #
1875 #
1876 #
1877 #
1878 #
1879 #
1880 #
1881 #
1882 #
1883 #
1884 #
1885 #
1886 #
1887 #
1888 #
1889 #
1890 #
1891 #
1892 #
1893 #
1894 #
1895 #
1896 #
1897 #
1898 #
1899 #
1900 #
1901 #
1902 #
1903 #
1904 #
1905 #
1906 #
1907 #
1908 #
1909 #
1910 #
1911 #
1912 #
1913 #
1914 #
1915 #
1916 #
1917 #
1918 #
1919 #
1920 #
1921 #
1922 #
1923 #
1924 #
1925 #
1926 #
1927 #
1928 #
1929 #
1930 #
1931 #
1932 #
1933 #
1934 #
1935 #
1936 #
1937 #
1938 #
1939 #
1940 #
1941 #
1942 #
1943 #
1944 #
1945 #
1946 #
1947 #
1948 #
1949 #
1950 #
1951 #
1952 #
1953 #
1954 #
1955 #
1956 #
1957 #
1958 #
1959 #
1960 #
1961 #
1962 #
1963 #
1964 #
1965 #
1966 #
1967 #
1968 #
1969 #
1970 #
1971 #
1972 #
1973 #
1974 #
1975 #
1976 #
1977 #
1978 #
1979 #
1980 #
1981 #
1982 #
1983 #
1984 #
1985 #
1986 #
1987 #
1988 #
1989 #
1990 #
1991 #
1992 #
1993 #
1994 #
1995 #
1996 #
1997 #
1998 #
1999 #
2000 #
2001 #
2002 #
2003 #
2004 #
2005 #
2006 #
2007 #
2008 #
2009 #
2010 #
2011 #
2012 #
2013 #
2014 #
2015 #
2016 #
2017 #
2018 #
2019 #
2020 #
2021 #
2022 #
2023 #
2024 #
2025 #
2026 #
2027 #
2028 #
2029 #
2030 #
2031 #
2032 #
2033 #
2034 #
2035 #
2036 #
2037 #
2038 #
2039 #
2040 #
2041 #
2042 #
2043 #
2044 #
2045 #
2046 #
2047 #
2048 #
2049 #
2050 #
2051 #
2052 #
2053 #
2054 #
2055 #
2056 #
2057 #
2058 #
2059 #
2060 #
2061 #
2062 #
2063 #
2064 #
2065 #
2066 #
2067 #
2068 #
2069 #
2070 #
2071 #
2072 #
2073 #
2074 #
2075 #
2076 #
2077 #
2078 #
2079 #
2080 #
2081 #
2082 #
2083 #
2084 #
2085 #
2086 #
2087 #
2088 #
2089 #
2090 #
2091 #
2092 #
2093 #
2094 #
2095 #
2096 #
2097 #
2098 #
2099 #
2100 #
2101 #
2102 #
2103 #
2104 #
2105 #
2106 #
2107 #
2108 #
2109 #
2110 #
2111 #
2112 #
2113 #
2114 #
2115 #
2116 #
2117 #
2118 #
2119 #
2120 #
2121 #
2122 #
2123 #
2124 #
2125 #
2126 #
2127 #
2128 #
2129 #
2130 #
2131 #
2132 #
2133 #
2134 #
2135 #
2136 #
2137 #
2138 #
2139 #
2140 #
2141 #
2142 #
2143 #
2144 #
2145 #
```

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Project: (None)

Global Environment

Functions

- boring_functi... function (x)
- mayor10 function (x)
- mayorque function (x, n)
- Mediacontamin... function (directorio, contaminante, f...
- ok function ()

Files Plots Packages Help Viewer

R Apply a Function Over a Ragged Array

Find in Topic

Apply a Function Over a Ragged Array

Description

Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

Usage

tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)

Arguments

X an atomic object, typically a vector.

INDEX list of one or more factors, each of same length as X. The elements are coerced to factors by [as.factor](#).

Console

In this lesson, you learned how to use vapply() as a safer alternative to sapply(), which is most helpful when writing your own functions. You also learned how to use tapply() to split your data into groups based on the value of some variable, then apply a function to each group. These functions will come in handy on your quest to become a better data analyst.

100%

would you like to receive credit for completing this course on coursera.org?

1: No
2: Yes

Selection: |

Anayeli Cocolletzi

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Project: (None)

Global Environment

Variables

- plants 5166 obs. of 10 variables
- shape_mat fnc [1:2, 1:5] 0 4 0 2 0 1 0 4 0 50

Values

- a num [1:11] 13.5 40.6 67.7 85.7 94.7 ...
- b 99.9991691775631
- b0 0.5

Files Plots Packages Help Viewer

R Apply a Function Over a Ragged Array

Find in Topic

Apply a Function Over a Ragged Array

Description

Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

Usage

tapply(X, INDEX, FUN = NULL, ..., simplify = TRUE)

Arguments

X an atomic object, typically a vector.

INDEX list of one or more factors, each of same length as X. The elements are coerced to factors by [as.factor](#).

Console

In this lesson, you learned how to get a feel for the structure and contents of a new dataset using a collection of simple and useful functions. Taking the time to do this upfront can save you time and frustration later on in your analysis.

100%

would you like to receive credit for completing this course on coursera.org?

1: No
2: Yes

Selection: |

Anayeli Cocolletzi

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Go to file/function Addins

Funciones(appli.R)*

```
232 plot(x,y,main = "Modelo Poisson",col="forestgreen")
233
234
235 x <- sample(1:20)
236 x
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
```

Environment History

Global Environment

b0	0.5
b1	2
c	900.000160568198
c1s_list	List of 30
c1s_vect	Named chr [1:30] "factor" "integer" "in...
cm	num [1:100] 9.6 9.8 10.4 10.6 8.6 10.6 ...

Files Plots Packages Help Viewer

R: The Normal Distribution Find in Topic

The Normal Distribution

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.

Usage

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Arguments

x: vector of quantiles

Selection:

1: Yes
2: No

Simulation is practically a field of its own and we've only skimmed the surface of what's possible. I encourage you to explore these and other functions further on your own.

100%

R Documentation

06:28 p.m.
11/10/2016

RStudio

File Edit Code View Plots Session Build Debug Tools Help

Go to File/Function Addins

Project: (None)

funciones(appl).R

```
232 plot(x,y,main = "Modelo Poisson",col="forestgreen")
233
234
235 x <- sample(1:20)
236 x
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
```

Environment History

Global Environment

Object	Value
poisson2	100000
s	List of 5
t1	2016-10-11 19:53:44
t2	2016-10-11 19:55:54
t3	"October 17, 1986 08:24"
t4	List of 1

Files Plots Packages Help Viewer

R: The Normal Distribution

Normal (stats)

R Documentation

The Normal Distribution

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.

Usage

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Arguments

x: vector of quantiles

Console

```
1: In this lesson, you learned how to work with dates and times in R. while it is important to
2: understand the basics, if you find yourself working with dates and times often, you may want to
3: check out the lubridate package by hadley wickham.
4:
5: would you like to receive credit for completing this course on coursera.org?
6:
7: 1: YES
8: 2: NO
9: Selection: |
```

08:07 p.m. 11/10/2016

