

# **Comparando listas e rankings**

Fernando Náufel

13/02/2024 18:18

# Índice

<b>Apresentação</b>	<b>3</b>
<b>1 Listas e <i>rankings</i></b>	<b>4</b>
1.1 Problema . . . . .	4
1.2 Criando <i>rankings</i> . . . . .	4
1.2.1 Representação . . . . .	4
1.2.2 Quantidade de <i>rankings</i> . . . . .	5
1.2.3 Criando um <i>ranking</i> a partir de um vetor . . . . .	6
1.3 Outras funções . . . . .	7
1.3.1 Mostrando um <i>ranking</i> graficamente . . . . .	7
1.3.2 Criando uma <i>tibble</i> com todos os <i>rankings</i> . . . . .	9
<b>2 O <i>ranking</i> concorda com a lista? Posições</b>	<b>13</b>
2.1 Usando $p$ como medida de concordância . . . . .	13
2.2 Usando $p$ e as posições dos elementos da lista . . . . .	14
2.2.1 Contando posições - . . . . .	14
2.2.2 Comparando <i>rankings</i> com valores diferentes de $p$ . . . . .	19
<b>3 O <i>ranking</i> concorda com a lista? Correlação</b>	<b>24</b>
3.0.1 Comparando <i>rankings</i> com valores diferentes de $p$ . . . . .	30

# Apresentação

???

# 1 Listas e *rankings*

## 1.1 Problema

Vamos trabalhar com listas e *rankings* sujeitos às seguintes condições:

- A *lista* tem  $k$  elementos,  $k > 0$ , não ordenados.
- O *ranking* tem  $p$  elementos,  $p \geq k$ , ordenados, sem empates.
- Todos os elementos da lista também pertencem ao *ranking*.
- O último elemento do *ranking* sempre pertence à lista.
- As identidades dos elementos do *ranking* não importam — i.e., eles são indistinguíveis, a não ser por pertencerem ou não à lista (e pela ordem que ocupam no *ranking*, claro).

## 1.2 Criando *rankings*

### 1.2.1 Representação

Considere naturais  $k > 0$  e  $p \geq k$ .

Podemos representar um *ranking* através de um *string* contendo  $k$  caracteres “x” e  $p - k$  caracteres “-”.

“x” representa uma posição ocupada por um elemento da lista.

“-” representa uma posição ocupada por um elemento que não está na lista.

Você pode usar a função `rk()` para criar um *ranking*, passando um *string* da forma acima:

```
rk('xx--x')
```

ranking: [xx--x] (p = 5, k = 3)

R vai mostrar o *ranking* com os valores de  $k$  e  $p$ . Se quiser ver o *ranking* com caracteres Unicode, use a função `print` com o argumento `unicode = TRUE`:

```
print(rk('xx--x'), unicode = TRUE)
```

```
ranking: [ •• ] (p = 5, k = 3)
```

## 1.2.2 Quantidade de *rankings*

Dados  $k > 0$  e  $p \geq k$  fixos, quantos *rankings* existem?

Para montar um *ranking*:

1. Sabemos que a última posição é ocupada por alguém da lista.
2. Só resta escolher as posições dos  $k-1$  elementos restantes da lista dentre as  $p-1$  posições restantes no *ranking*, o que dá  $\binom{p-1}{k-1}$  escolhas.

Assim, a quantidade total de *rankings* para  $k$  e  $p$  dados é

$$\binom{p-1}{k-1}$$

Por exemplo, para  $k=3, p=5$ , os  $\binom{4}{2} = 6$  *rankings* possíveis são

- xx--x
- x-x-x
- x--xx
- -xx-x
- -x-xx
- --xxx

A tabela a seguir (na verdade, um pedaço do triângulo de Pascal) mostra as quantidades de *rankings* possíveis para alguns valores de  $k$  e  $p$ :

$p$	$k$									
	1	2	3	4	5	6	7	8	9	10
1	1									
2	1	1								
3	1	2	1							
4	1	3	3	1						
5	1	4	6	4	1					

6	1	5	10	10	5	1				
7	1	6	15	20	15	6	1			
8	1	7	21	35	35	21	7	1		
9	1	8	28	56	70	56	28	8	1	
10	1	9	36	84	126	126	84	36	9	1
11	1	10	45	120	210	252	210	120	45	10
12	1	11	55	165	330	462	462	330	165	55
13	1	12	66	220	495	792	924	792	495	220
14	1	13	78	286	715	1.287	1.716	1.716	1.287	715
15	1	14	91	364	1.001	2.002	3.003	3.432	3.003	2.002
16	1	15	105	455	1.365	3.003	5.005	6.435	6.435	5.005
17	1	16	120	560	1.820	4.368	8.008	11.440	12.870	11.440
18	1	17	136	680	2.380	6.188	12.376	19.448	24.310	24.310
19	1	18	153	816	3.060	8.568	18.564	31.824	43.758	48.620
20	1	19	171	969	3.876	11.628	27.132	50.388	75.582	92.378
21	1	20	190	1.140	4.845	15.504	38.760	77.520	125.970	167.960
22	1	21	210	1.330	5.985	20.349	54.264	116.280	203.490	293.930
23	1	22	231	1.540	7.315	26.334	74.613	170.544	319.770	497.420
24	1	23	253	1.771	8.855	33.649	100.947	245.157	490.314	817.190
25	1	24	276	2.024	10.626	42.504	134.596	346.104	735.471	1.307.504
26	1	25	300	2.300	12.650	53.130	177.100	480.700	1.081.575	2.042.975
27	1	26	325	2.600	14.950	65.780	230.230	657.800	1.562.275	3.124.550
28	1	27	351	2.925	17.550	80.730	296.010	888.030	2.220.075	4.686.825
29	1	28	378	3.276	20.475	98.280	376.740	1.184.040	3.108.105	6.906.900
30	1	29	406	3.654	23.751	118.755	475.020	1.560.780	4.292.145	10.015.005

### 1.2.3 Criando um *ranking* a partir de um vetor

Em vez de especificar as  $p$  posições do *ranking*, pode ser mais compacto especificar as  $k$  posições do *ranking* que são ocupadas por elementos da lista.

Para isso, a função `rk()` também aceita um vetor numérico com  $k$  elementos.

```
rk(c(1, 3, 5, 7))
```

```
ranking: [x-x-x-x] (p = 7, k = 4)
```

Observe que as posições não precisam ser passadas em ordem:

```
rk(c(3, 7, 5, 1))
```

ranking: [x-x-x-x] (p = 7, k = 4)

A função detecta vetores que não podem representar *rankings*:

```
rk(c(3, 7, 3, 1))
```

Error in validate\_rk(x):

Valores precisam ser inteiros positivos, sem repetições.

```
rk(c(5, 7, 3, 1.5))
```

Error in validate\_rk(x):

Valores precisam ser inteiros positivos, sem repetições.

```
rk(c(5, -7, 3, 1))
```

Error in validate\_rk(x):

Valores precisam ser inteiros positivos, sem repetições.

## 1.3 Outras funções

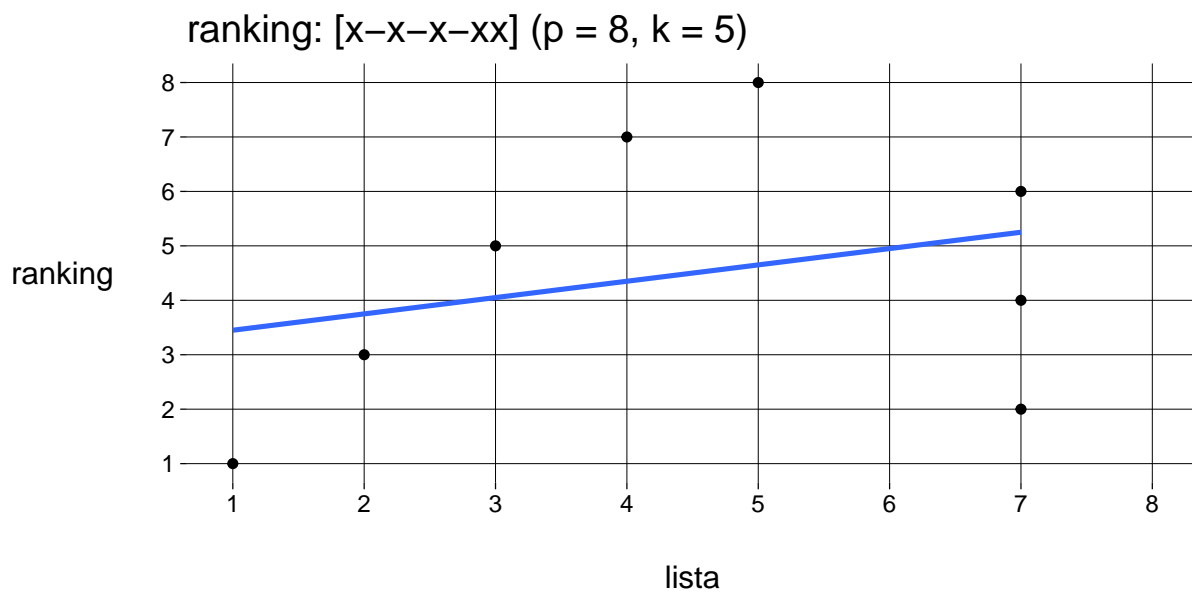
### 1.3.1 Mostrando um *ranking* graficamente

A função `plot` recebe um *ranking* e gera um gráfico de pontos, com um ponto para cada elemento.

No eixo *x*, a posição do elemento na lista.

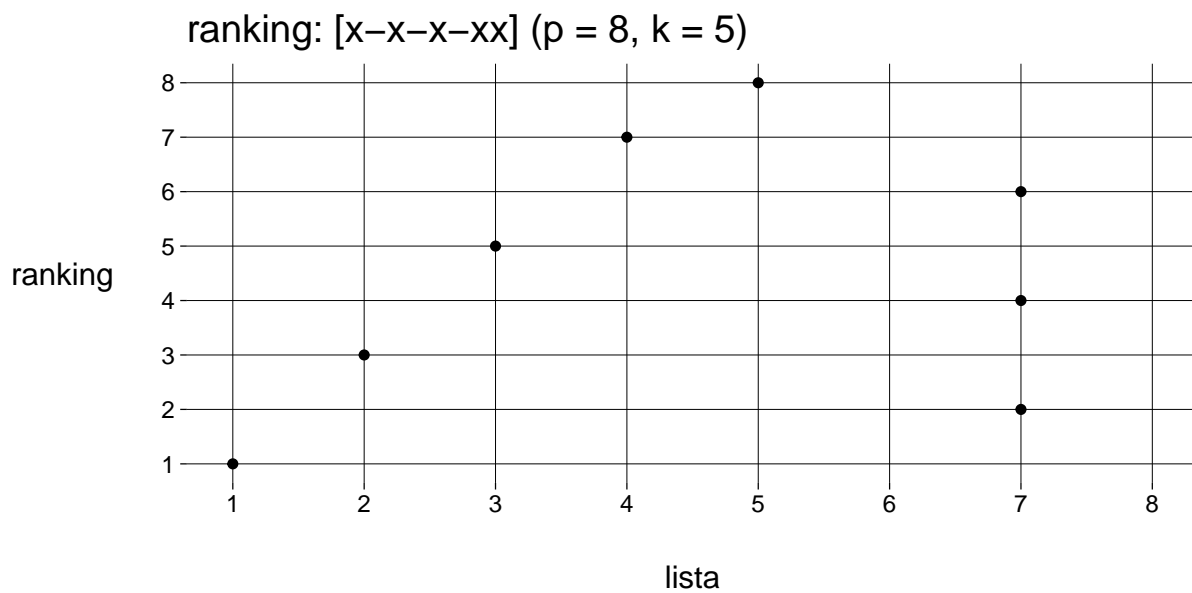
No eixo *y*, a posição do elemento no *ranking*.

```
r <- rk('x-x-x-xx')  
plot(r)
```



O argumento `reta`, opcional, especifica se deve ser incluída uma reta de regressão linear via mínimos quadrados. O *default* é `TRUE`.

```
plot(r, reta = FALSE)
```



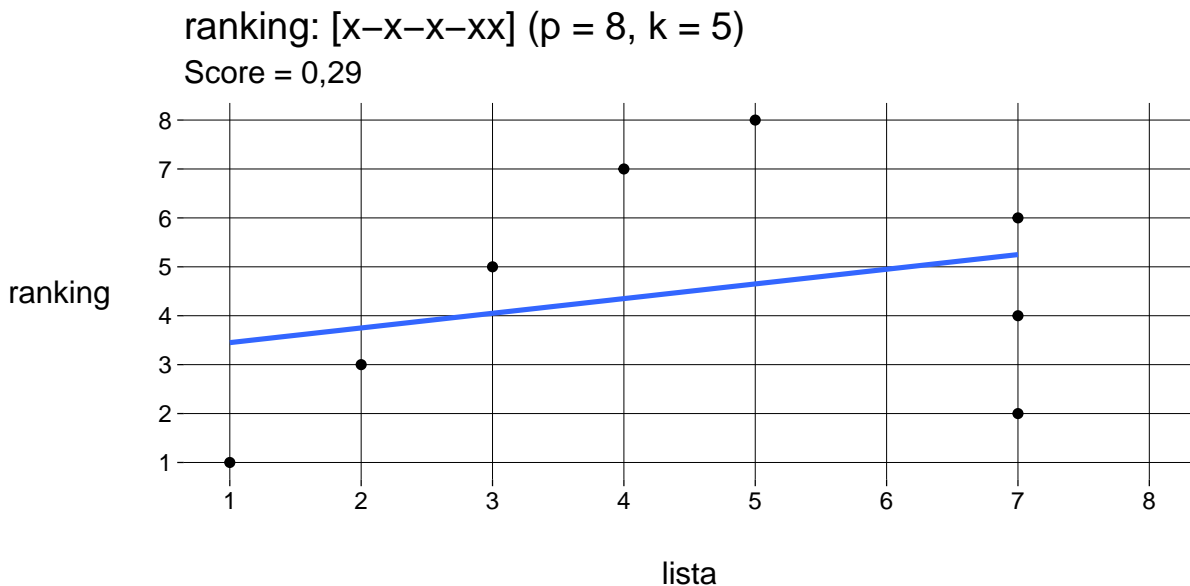
A função `plot` pode receber um argumento `fun`, opcional, especificando uma função para calcular o *score* deste *ranking* (i.e., alguma forma de correlação entre o *ranking* e a lista). O *score* vai ser mostrado no título do gráfico.



A função passada em `fun` deve receber, como único argumento, uma *tibble* com uma linha para cada elemento, e as colunas

- `nome`: um vetor de caracteres contendo 'x' e '-'.
- `pos_lista`: um vetor numérico com a posição de cada elemento na lista, sendo que elementos do *ranking* que não pertencem à lista são adicionados à lista todos empatados no final.
- `pos_ranking`: um vetor numérico com a posição de cada elemento no *ranking*.

```
plot(  
  r,  
  fun = \(df) {  
    cor(df$pos_lista, df$pos_ranking) %>% round(2)  
  }  
)
```



### 1.3.2 Criando uma *tibble* com todos os *rankings*

Dados valores de  $p$  e  $k$  (nesta ordem), a função `criar_df_rankings()` retorna uma *tibble* com todos os  $\binom{p-1}{k-1}$  *rankings* possíveis, como objetos (S3) e como *strings*.

Todos os *rankings* com  $p = 8$  e  $k = 5$ :

```
criar_df_rankings(8, 5)
```

```

# A tibble: 35 x 2
  ranking ranking_str
  <list>   <chr>
1 <rk>     xxxx---x
2 <rk>     xxx-x--x
3 <rk>     xxx--x-x
4 <rk>     xxx---xx
5 <rk>     xx-xx--x
6 <rk>     xx-x-x-x
7 <rk>     xx-x--xx
8 <rk>     xx--xx-x
9 <rk>     xx--x-xx
10 <rk>    xx---xxx
11 <rk>    x-xxx--x
12 <rk>    x-xx-x-x
13 <rk>    x-xx--xx
14 <rk>    x-x-xx-x
15 <rk>    x-x-x-xx
16 <rk>    x-x--xxx
17 <rk>    x--xxx-x
18 <rk>    x--xx-xx
19 <rk>    x--x-xxx
20 <rk>    x---xxxx
21 <rk>    -xxxx--x
22 <rk>    -xxx-x-x
23 <rk>    -xxx--xx
24 <rk>    -xx-xx-x
25 <rk>    -xx-x-xx
26 <rk>    -xx--xxx
27 <rk>    -x-xxx-x
28 <rk>    -x-xx-xx
29 <rk>    -x-x-xxx
30 <rk>    -x--xxxx
31 <rk>    --xxxx-x
32 <rk>    --xxx-xx
33 <rk>    --xx-xxx
34 <rk>    --x-xxxx
35 <rk>    ---xxxxx

```

Se for passado apenas o valor de  $p$ , a função retorna uma *tibble* com todos os *rankings* possíveis de comprimento  $p$  (com  $k$  variando de 1 até  $p$ ). Exercício: quantos são?

Todos os *rankings* com  $p = 5$ :

```
criar_df_rankings(5)
```

```
# A tibble: 16 x 2
  ranking ranking_str
  <list>   <chr>
1 <rk>     ----x
2 <rk>     x---x
3 <rk>     -x--x
4 <rk>     --x-x
5 <rk>     ---xx
6 <rk>     xx--x
7 <rk>     x-x-x
8 <rk>     x--xx
9 <rk>     -xx-x
10 <rk>    -x-xx
11 <rk>    --xxx
12 <rk>    xxx-x
13 <rk>    xx-xx
14 <rk>    x-xxx
15 <rk>    -xxxx
16 <rk>    xxxxx
```

Se você quiser a representação em *string* usando unicode, basta passar o argumento `unicode = TRUE`:

```
criar_df_rankings(5, unicode = TRUE)
```

```
# A tibble: 16 x 2
  ranking ranking_str
  <list>   <chr>
1 <rk>     ....
2 <rk>     ...
3 <rk>     ..
4 <rk>     .
5 <rk>     .
6 <rk>     .
7 <rk>     .
8 <rk>     .
9 <rk>     .
10 <rk>    .
11 <rk>    .
```

12	<rk>	•
13	<rk>	•
14	<rk>	•
15	<rk>	•
16	<rk>	

## 2 O *ranking* concorda com a lista? Posições

### 2.1 Usando $p$ como medida de concordância

Imagine que a lista de  $k$  elementos foi definida por uma autoridade, usando critérios que não conhecemos.

Em uma tentativa de descobrir esses critérios, construímos um modelo para avaliar todos os elementos da população (que incluem os  $k$  elementos da lista).

Nosso modelo produz um *ranking* de todos os elementos. Para facilitar, vamos supor que não há empates no *ranking*.

Uma pergunta natural sobre a qualidade do *ranking* produzido é

Quantas posições do *ranking* são necessárias para incluir todos os  $k$  elementos da lista?

A resposta é  $p$ , a posição, no *ranking*, do elemento da lista com pior classificação.

Aliás, é por isso que convencionamos, no capítulo anterior, que nossos *rankings* sempre terminam com um elemento da lista.

Um exemplo:

- A lista contém  $k = 5$  elementos.
- O *ranking*  $r_1$  é **xx-x-xx**, com  $p = 7$ .
- O *ranking*  $r_2$  é **-xxxxx**, com  $p = 6$ .

Segundo a medida proposta aqui,  $r_2$  é melhor que  $r_1$ .

Ou seja, quanto menor o valor de  $p$ , melhor o *ranking*.

Embora comparar *rankings* através de seus valores de  $p$  seja simples, podemos examinar medidas alternativas, que sejam mais finas que esta.

Por exemplo, é discutível se os dois rankings **xx---x** e **---xxx** devem ser considerados igualmente bons; no entanto, ambos têm  $p = 6$ .

## 2.2 Usando $p$ e as posições dos elementos da lista

### 2.2.1 Contando posições -

Dado um *ranking*  $r$  com  $k$  e  $p$ , queremos definir uma função  $s(r)$  —  $s$  de *score* — com as seguintes características:

- Se  $r$  não contiver “-”, então  $s(r) = 1$ . Neste caso,  $r$  é um *ranking* perfeito, que coincide com a lista (por exemplo, xxxxx). Em casos assim,  $k = p$ . Vamos definir  $s$  como sendo da forma

$$s(r) = \frac{k}{p} + \dots$$

onde as reticências representam uma parcela que ainda vamos definir. Se  $r$  for um *ranking* perfeito, a parcela  $k/p$  será 1, e vamos definir a parcela restante para que seja igual a zero.

- A parcela restante deve ter valor maior quanto melhor for o *ranking*. Quanto mais próximos do fim do *ranking* estiverem os caracteres “-”, melhor ele será. Uma quantidade natural seria

$$\frac{\text{soma\_}}{\sum_{i=1,p} i} = \frac{\text{soma\_}}{p(p+1)/2} = \frac{2 \text{soma\_}}{p(p+1)}$$

onde soma\_ é a soma das posições ocupadas por “-” em  $r$ .

Como queríamos, quando  $r$  for um *ranking* perfeito, soma\_ = 0, e então  $s(r) = 1$ .

- Mas também queremos que somente *rankings* perfeitos tenham  $s(r) = 1$ . Para isso, considere que um *ranking* mais próximo do perfeito é da forma

x...x-x

Ou seja,  $k = p - 1$  e soma\_ =  $p - 1$ .

Vamos multiplicar a segunda parcela por  $\alpha$  de forma que  $s(r) < 1$  para este *ranking* quase perfeito:

$$s(r) = \frac{p-1}{p} + \frac{2(p-1)}{p(p+1)} \cdot \alpha$$

Então

$$\begin{aligned}
s(r) < 1 &\Leftrightarrow \frac{2(p-1)}{p(p+1)} \cdot \alpha < \frac{1}{p} \\
&\Leftrightarrow 2\alpha(p-1) < p+1 \\
&\Leftrightarrow \alpha < \frac{1}{2} \cdot \frac{p+1}{p-1} \\
&\Leftrightarrow \alpha = \frac{1}{m} \cdot \frac{p+1}{p-1} \quad (m > 2)
\end{aligned}$$

o que dá

$$\begin{aligned}
s(r) &= \frac{k}{p} + \frac{2 \text{soma\_}}{p(p+1)} \cdot \alpha \\
&= \frac{k}{p} + \frac{2 \text{soma\_}}{p(p+1)} \cdot \frac{1}{m} \cdot \frac{p+1}{p-1} \quad (m > 2) \\
&= \frac{k}{p} + \frac{2 \text{soma\_}}{p(p-1)} \cdot \frac{1}{m} \quad (m > 2) \\
&= \frac{k}{p} + \frac{\text{soma\_}}{p(p-1)} \cdot \frac{2}{m} \quad (m > 2)
\end{aligned}$$

Dependendo do valor de  $m > 2$  escolhido, teremos medidas diferentes.

A função que implementamos usa o *default* de  $m = 10$ , mas valores diferentes podem ser passados.

```
r <- rk('xxx-x')
s(r)
```

```
[1] 0,84
```

Para  $p = 8$ , alguns exemplos:

```
s(
  list(
    rk('xxxxxxxx'),
    rk('xxxxxxx-x'),
    rk('-xxxxxxxx')
  )
)
```

```
[1] 1,0000000 0,9000000 0,8785714
```

Eis todos os *rankings* de comprimento 8, com suas pontuações:

```
# A tibble: 128 x 3
  ranking ranking_str      s
  <list>   <chr>         <dbl>
1 <rk>     xxxxxxxx         1
2 <rk>     xxxxxx-x         0.9
3 <rk>     xxxxxx-xx        0.896
4 <rk>     xxxxx-xxx        0.893
5 <rk>     xxx-xxxx         0.889
6 <rk>     xx-xxxxx         0.886
7 <rk>     x-xxxxxx         0.882
8 <rk>     -xxxxxxx         0.879
9 <rk>     xxxxxx--x        0.796
10 <rk>    xxxxx-x-x       0.793
11 <rk>    xxxx--xx       0.789
12 <rk>    xxx-xx-x       0.789
13 <rk>    xxx-x-xx       0.786
14 <rk>    xx-xxx-x       0.786
15 <rk>    xxx--xxx       0.782
16 <rk>    xx-xx-xx       0.782
17 <rk>    x-xxxx-x       0.782
18 <rk>    xx-x-xxx       0.779
19 <rk>    x-xxx-xx       0.779
20 <rk>    -xxxxx-x       0.779
21 <rk>    xx--xxxx       0.775
22 <rk>    x-xx-xxx       0.775
23 <rk>    -xxxx-xx       0.775
24 <rk>    x-x-xxxx       0.771
25 <rk>    -xxx-xxx       0.771
26 <rk>    x--xxxxx       0.768
27 <rk>    -xx-xxxx       0.768
28 <rk>    -x-xxxxx       0.764
29 <rk>    --xxxxxx       0.761
30 <rk>    xxxx---x       0.689
31 <rk>    xxx-x--x       0.686
32 <rk>    xxx--x-x       0.682
33 <rk>    xx-xx--x       0.682
34 <rk>    xxx---xx       0.679
35 <rk>    xx-x-x-x       0.679
36 <rk>    x-xxx--x       0.679
37 <rk>    xx-x--xx       0.675
```



38	<rk>	xx--xx-x	0.675
39	<rk>	x-xx-x-x	0.675
40	<rk>	-xxxx--x	0.675
41	<rk>	xx--x-xx	0.671
42	<rk>	x-xx--xx	0.671
43	<rk>	x-x-xx-x	0.671
44	<rk>	-xxx-x-x	0.671
45	<rk>	xx---xxx	0.668
46	<rk>	x-x-x-xx	0.668
47	<rk>	x--xxx-x	0.668
48	<rk>	-xxx--xx	0.668
49	<rk>	-xx-xx-x	0.668
50	<rk>	x-x--xxx	0.664
51	<rk>	x--xx-xx	0.664
52	<rk>	-xx-x-xx	0.664
53	<rk>	-x-xxx-x	0.664
54	<rk>	x--x-xxx	0.661
55	<rk>	-xx--xxx	0.661
56	<rk>	-x-xx-xx	0.661
57	<rk>	--xxxx-x	0.661
58	<rk>	x---xxxx	0.657
59	<rk>	-x-x-xxx	0.657
60	<rk>	--xxx-xx	0.657
61	<rk>	-x--xxxx	0.654
62	<rk>	--xx-xxx	0.654
63	<rk>	--x-xxxx	0.65
64	<rk>	---xxxxx	0.646
65	<rk>	xxx----x	0.579
66	<rk>	xx-x---x	0.575
67	<rk>	xx--x--x	0.571
68	<rk>	x-xx---x	0.571
69	<rk>	xx---x-x	0.568
70	<rk>	x-x-x--x	0.568
71	<rk>	-xxx---x	0.568
72	<rk>	xx----xx	0.564
73	<rk>	x-x--x-x	0.564
74	<rk>	x--xx--x	0.564
75	<rk>	-xx-x--x	0.564
76	<rk>	x-x---xx	0.561
77	<rk>	x--x-x-x	0.561
78	<rk>	-xx--x-x	0.561
79	<rk>	-x-xx--x	0.561
80	<rk>	x--x--xx	0.557

81	<rk>	x---xx-x	0.557
82	<rk>	-xx---xx	0.557
83	<rk>	-x-x-x-x	0.557
84	<rk>	--xxx--x	0.557
85	<rk>	x---x-xx	0.554
86	<rk>	-x-x--xx	0.554
87	<rk>	-x--xx-x	0.554
88	<rk>	--xx-x-x	0.554
89	<rk>	x----xxx	0.55
90	<rk>	-x--x-xx	0.55
91	<rk>	--xx--xx	0.55
92	<rk>	--x-xx-x	0.55
93	<rk>	-x---xxx	0.546
94	<rk>	--x-x-xx	0.546
95	<rk>	---xxx-x	0.546
96	<rk>	--x--xxx	0.543
97	<rk>	---xx-xx	0.543
98	<rk>	---x-xxx	0.539
99	<rk>	----xxxx	0.536
100	<rk>	xx-----x	0.464
101	<rk>	x-x-----x	0.461
102	<rk>	x--x---x	0.457
103	<rk>	-xx-----x	0.457
104	<rk>	x---x--x	0.454
105	<rk>	-x-x---x	0.454
106	<rk>	x-----x-x	0.45
107	<rk>	-x--x--x	0.45
108	<rk>	--xx---x	0.45
109	<rk>	x-----xx	0.446
110	<rk>	-x---x-x	0.446
111	<rk>	--x-x--x	0.446
112	<rk>	-x----xx	0.443
113	<rk>	--x--x-x	0.443
114	<rk>	---xx--x	0.443
115	<rk>	--x---xx	0.439
116	<rk>	---x-x-x	0.439
117	<rk>	---x--xx	0.436
118	<rk>	----xx-x	0.436
119	<rk>	----x-xx	0.432
120	<rk>	-----xxx	0.429
121	<rk>	x-----x	0.346
122	<rk>	-x-----x	0.343
123	<rk>	--x-----x	0.339

124	<rk>	---x---x	0.336
125	<rk>	----x--x	0.332
126	<rk>	-----x-x	0.329
127	<rk>	-----xx	0.325
128	<rk>	-----x	0.225

Perceba que pode haver empates: `xxxx--xx` e `xxx-xx-x` têm o mesmo valor de  $s$ . É razoável achar que estes dois *rankings* têm a mesma qualidade.

## 2.2.2 Comparando *rankings* com valores diferentes de $p$

Como a lista é dada e fixa, só faz sentido, na prática, comparar *rankings* com o mesmo valor de  $k$ .

Vamos examinar, para uma lista com  $k = 15$ , os *rankings* possíveis com  $p$  variando de 15 a 20.

São 15.504 *rankings*. Eis os 100 melhores:

```
# A tibble: 100 x 4
  ranking ranking_str      s      p
  <list>   <chr>         <dbl> <int>
1 <rk>     xxxxxxxxxxxxxxxx      1      15
2 <rk>     xxxxxxxxxxxxxxxx-x 0.95      16
3 <rk>     xxxxxxxxxxxxxxxx-xx 0.949      16
4 <rk>     xxxxxxxxxxxxxxxx-xxx 0.948      16
5 <rk>     xxxxxxxxxxxxxxxx-xxxx 0.948      16
6 <rk>     xxxxxxxxxxxxxx-xxxxx 0.947      16
7 <rk>     xxxxxxxxxxxx-xxxxxx 0.946      16
8 <rk>     xxxxxxxxx-xxxxxxx 0.945      16
9 <rk>     xxxxxxxx-xxxxxxxx 0.944      16
10 <rk>     xxxxxx-xxxxxxxxxx 0.943      16
11 <rk>     xxxxx-xxxxxxxxxxx 0.942      16
12 <rk>     xxxx-xxxxxxxxxxxx 0.942      16
13 <rk>     xxx-xxxxxxxxxxxxx 0.941      16
14 <rk>     xx-xxxxxxxxxxxxxx 0.94      16
15 <rk>     x-xxxxxxxxxxxxxxx 0.939      16
16 <rk>     -xxxxxxxxxxxxxxxx 0.938      16
17 <rk>     xxxxxxxxxxxxxxxx--x 0.905      17
18 <rk>     xxxxxxxxxxxxxxxx-x-x 0.904      17
19 <rk>     xxxxxxxxxxxxxxxx--xx 0.904      17
20 <rk>     xxxxxxxxxxxxxxxx-xx-x 0.904      17
```

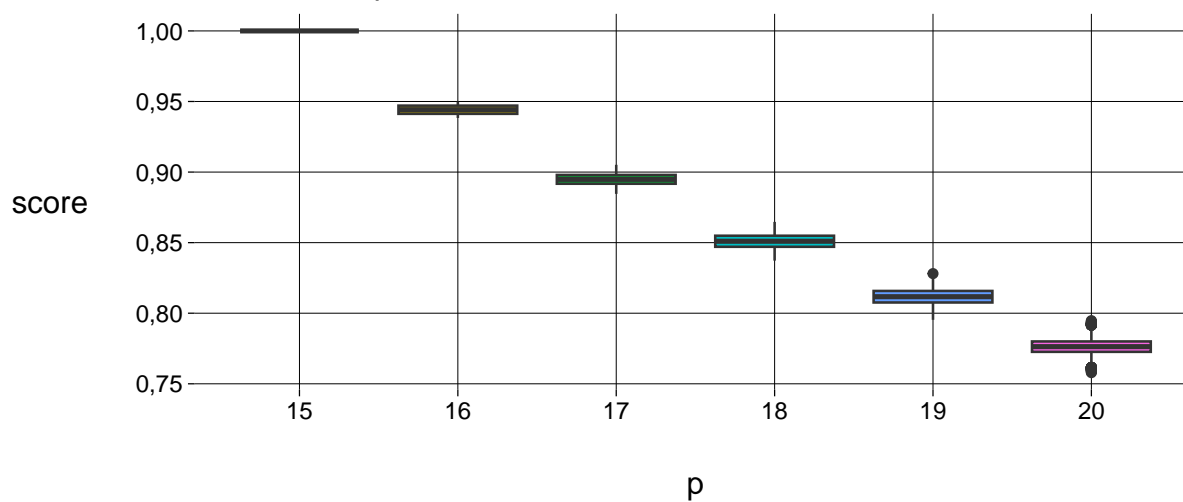
21	<rk>	xxxxxxxxxxxxx-x-xx	0.903	17
22	<rk>	xxxxxxxxxxxxx-xxx-x	0.903	17
23	<rk>	xxxxxxxxxxxxx--xxx	0.902	17
24	<rk>	xxxxxxxxxxxxx-xx-xx	0.902	17
25	<rk>	xxxxxxxxxxxxx-xxxx-x	0.902	17
26	<rk>	xxxxxxxxxxxxx-x-xxx	0.901	17
27	<rk>	xxxxxxxxxxxxx-xxx-xx	0.901	17
28	<rk>	xxxxxxxxxxxxx-xxxx-x	0.901	17
29	<rk>	xxxxxxxxxxxxx--xxxx	0.901	17
30	<rk>	xxxxxxxxxxxxx-xx-xxx	0.901	17
31	<rk>	xxxxxxxxxxxxx-xxxx-xx	0.901	17
32	<rk>	xxxxxxxxxxxxx-xxxxxx-x	0.901	17
33	<rk>	xxxxxxxxxxxxx-x-xxxx	0.9	17
34	<rk>	xxxxxxxxxxxxx-xxx-xxx	0.9	17
35	<rk>	xxxxxxxxxxxxx-xxxx-xx	0.9	17
36	<rk>	xxxxxxxxxxxxx-xxxxxxx-x	0.9	17
37	<rk>	xxxxxxxxxxxxx--xxxxxx	0.899	17
38	<rk>	xxxxxxxxxxxxx-xx-xxxx	0.899	17
39	<rk>	xxxxxxxxxxxxx-xxxx-xxx	0.899	17
40	<rk>	xxxxxxxxxxxxx-xxxxxx-xx	0.899	17
41	<rk>	xxxxxxxxxxxxx-xxxxxxx-x	0.899	17
42	<rk>	xxxxxxxxxxxxx-x-xxxxxx	0.899	17
43	<rk>	xxxxxxxxxxxxx-xxx-xxxx	0.899	17
44	<rk>	xxxxxxxxxxxxx-xxxxxx-xxx	0.899	17
45	<rk>	xxxxxxxxxxxxx-xxxxxxx-xx	0.899	17
46	<rk>	xxxxxx-xxxxxxxxxxx-x	0.899	17
47	<rk>	xxxxxxxxxxxxx--xxxxxxx	0.898	17
48	<rk>	xxxxxxxxxxxxx-xx-xxxxxx	0.898	17
49	<rk>	xxxxxxxxxxxxx-xxxx-xxxx	0.898	17
50	<rk>	xxxxxxxxxxxxx-xxxxxx-xxx	0.898	17
51	<rk>	xxxxxx-xxxxxxxxxxx-xx	0.898	17
52	<rk>	xxxxx-xxxxxxxxxxxxxx-x	0.898	17
53	<rk>	xxxxxxxxxxxxx-x-xxxxxxx	0.897	17
54	<rk>	xxxxxxxxxxxxx-xxx-xxxxxx	0.897	17
55	<rk>	xxxxxxx-xxxxxx-xxxx	0.897	17
56	<rk>	xxxxxx-xxxxxxxxxxx-xxx	0.897	17
57	<rk>	xxxxx-xxxxxxxxxxx-xx	0.897	17
58	<rk>	xxx-xxxxxxxxxxxxxx-x	0.897	17
59	<rk>	xxxxxxxxxxxxx--xxxxxxx	0.896	17
60	<rk>	xxxxxxxxxxxxx-xx-xxxxxx	0.896	17
61	<rk>	xxxxxxx-xxxx-xxxxxx	0.896	17
62	<rk>	xxxxxx-xxxxxx-xxxx	0.896	17
63	<rk>	xxxxx-xxxxxxxxxxx-xxx	0.896	17

64	<rk>	xxx-xxxxxxxxxxx-xx	0.896	17
65	<rk>	xx-xxxxxxxxxxxxx-x	0.896	17
66	<rk>	xxxxxxx-x-xxxxxxx	0.896	17
67	<rk>	xxxxxxx-xxx-xxxxxxx	0.896	17
68	<rk>	xxxxxx-xxxxx-xxxxxx	0.896	17
69	<rk>	xxxxx-xxxxxxx-xxxxx	0.896	17
70	<rk>	xxx-xxxxxxxxxxx-xxx	0.896	17
71	<rk>	xx-xxxxxxxxxxxxx-xx	0.896	17
72	<rk>	x-xxxxxxxxxxxxxxxxx-x	0.896	17
73	<rk>	xxxxxxx--xxxxxxxxxx	0.895	17
74	<rk>	xxxxxxx-xx-xxxxxxx	0.895	17
75	<rk>	xxxxxx-xxxxx-xxxxxxx	0.895	17
76	<rk>	xxxxx-xxxxxxx-xxxxxx	0.895	17
77	<rk>	xxx-xxxxxxxxxxx-xxxx	0.895	17
78	<rk>	xx-xxxxxxxxxxxxx-xxx	0.895	17
79	<rk>	x-xxxxxxxxxxxxxxxxx-xx	0.895	17
80	<rk>	-xxxxxxxxxxxxxxxxxx-x	0.895	17
81	<rk>	xxxxxxx-x-xxxxxxxxxx	0.894	17
82	<rk>	xxxxxx-xxx-xxxxxxx	0.894	17
83	<rk>	xxxxx-xxxxxx-xxxxxxx	0.894	17
84	<rk>	xxx-xxxxxxx-xxxxxx	0.894	17
85	<rk>	xx-xxxxxxxxxxx-xxxx	0.894	17
86	<rk>	x-xxxxxxxxxxxxx-xxx	0.894	17
87	<rk>	-xxxxxxxxxxxxxxxxxx-xx	0.894	17
88	<rk>	xxxxxxx--xxxxxxxxxx	0.893	17
89	<rk>	xxxxxx-xx-xxxxxxxxxx	0.893	17
90	<rk>	xxxxx-xxxxx-xxxxxxx	0.893	17
91	<rk>	xxx-xxxxxxx-xxxxxxx	0.893	17
92	<rk>	xx-xxxxxxxxxx-xxxxxx	0.893	17
93	<rk>	x-xxxxxxxxxxxxx-xxxx	0.893	17
94	<rk>	-xxxxxxxxxxxxxxxxxx-xxx	0.893	17
95	<rk>	xxxxxx-x-xxxxxxxxxxx	0.893	17
96	<rk>	xxxxx-xxx-xxxxxxxxxx	0.893	17
97	<rk>	xxx-xxxxxx-xxxxxxx	0.893	17
98	<rk>	xx-xxxxxxx-xxxxxxx	0.893	17
99	<rk>	x-xxxxxxxxxxx-xxxxxx	0.893	17
100	<rk>	-xxxxxxxxxxxxxxxxxx-xxxx	0.893	17

Os gráficos abaixo mostram os *scores* atribuídos para todos os *rankings* com  $k = 15$  e  $p$  variando de 15 a 20, separados por valores de  $p$ :

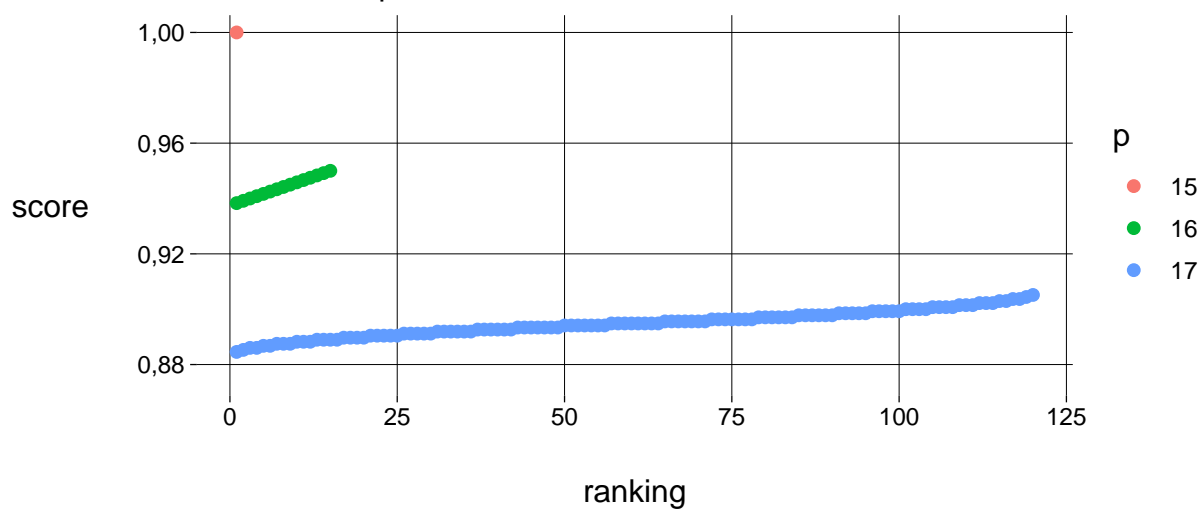
## Scores de todos os rankings

com  $k = 15$ ,  $p = 15$  a 20



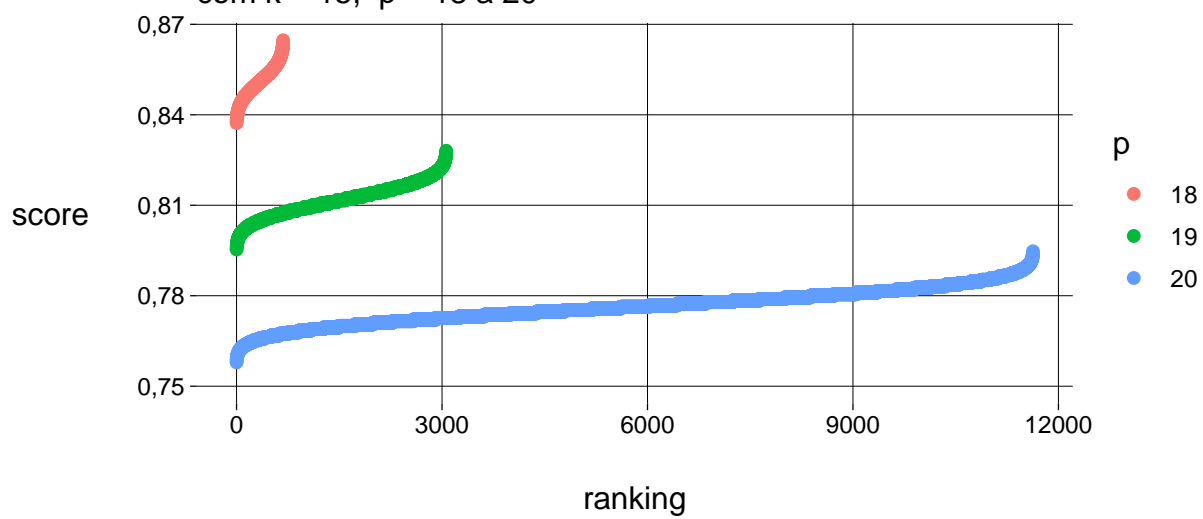
## Scores de todos os rankings

com  $k = 15$ ,  $p = 15$  a 17



# Scores de todos os rankings

com  $k = 15$ ,  $p = 18$  a  $20$



### 3 O *ranking* concorda com a lista? Correlação

```
r <- rk('xxx-x')
s(r)
```

```
[1] 0,9
```

Para  $p = 8$ , alguns exemplos:

```
s(
  list(
    rk('xxxxxxxx'),
    rk('xxxxxx-x'),
    rk('-xxxxxxx')
  )
)
```

```
[1] 1,0000000 0,9761905 0,3333333
```

Eis todos os *rankings* de comprimento 8, com suas pontuações:

```
# A tibble: 128 x 3
  ranking ranking_str      s
  <list>   <chr>         <dbl>
1 <rk>     xxxxxxxx         1
2 <rk>     xxxxxxx-x      0.976
3 <rk>     xxxxxx-xx       0.929
4 <rk>     xxxxxx--x       0.922
5 <rk>     xxxxx-x-x       0.862
6 <rk>     xxxxx-xxx       0.857
7 <rk>     xxxxx---x       0.830
8 <rk>     xxxxx--xx       0.802
9 <rk>     xxx-xx-x        0.778
10 <rk>     xxx-xxxx        0.762
```



11	<rk>	xxx-x--x	0.756
12	<rk>	xxx-x-xx	0.719
13	<rk>	xxx----x	0.685
14	<rk>	xxx--x-x	0.683
15	<rk>	xx-xxx-x	0.671
16	<rk>	xx-xx--x	0.659
17	<rk>	xx-xxxxx	0.643
18	<rk>	xxx--xxx	0.635
19	<rk>	xx-xx-xx	0.611
20	<rk>	xxx---xx	0.610
21	<rk>	xx-x---x	0.596
22	<rk>	xx-x-x-x	0.586
23	<rk>	x-xxxx-x	0.539
24	<rk>	x-xxx--x	0.537
25	<rk>	xx-x-xxx	0.527
26	<rk>	xx-x--xx	0.512
27	<rk>	xx--x--x	0.507
28	<rk>	x-xxxxxx	0.5
29	<rk>	xx--xx-x	0.488
30	<rk>	x-xx---x	0.482
31	<rk>	x-xxx-xx	0.479
32	<rk>	xx-----x	0.464
33	<rk>	x-xx-x-x	0.464
34	<rk>	xx--xxxx	0.419
35	<rk>	xx---x-x	0.419
36	<rk>	xx--x-xx	0.415
37	<rk>	x-xx-xxx	0.395
38	<rk>	x-x-x--x	0.393
39	<rk>	x-xx--xx	0.390
40	<rk>	-xxxx--x	0.390
41	<rk>	-xxxxx-x	0.383
42	<rk>	x-x-xx-x	0.366
43	<rk>	x-x----x	0.355
44	<rk>	-xxx---x	0.342
45	<rk>	-xxxxxxx	0.333
46	<rk>	xx----xx	0.330
47	<rk>	-xxxx-xx	0.323
48	<rk>	xx---xxx	0.317
49	<rk>	-xxx-x-x	0.317
50	<rk>	x-x--x-x	0.304
51	<rk>	x-x-x-xx	0.293
52	<rk>	x-x-xxxx	0.287
53	<rk>	x--xx--x	0.279

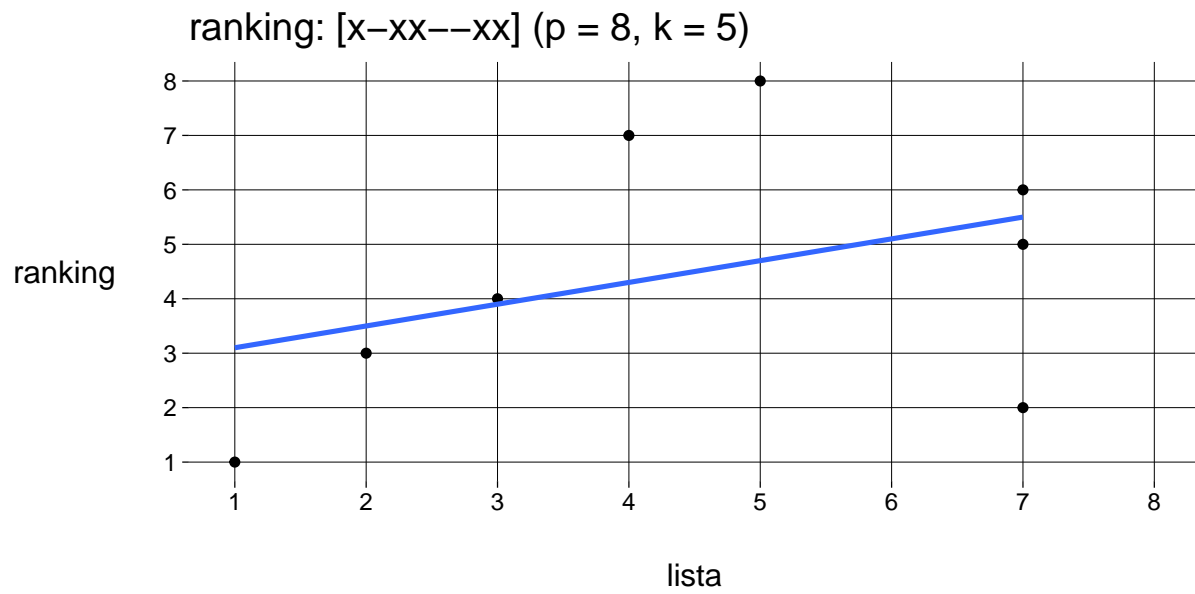
54	<rk>	-xx-x--x	0.254
55	<rk>	x--x---x	0.245
56	<rk>	x--xxx-x	0.244
57	<rk>	-xxx--xx	0.244
58	<rk>	-xxx-xxx	0.240
59	<rk>	-xx-xx-x	0.220
60	<rk>	-xx----x	0.218
61	<rk>	x-x---xx	0.216
62	<rk>	x-x--xxx	0.195
63	<rk>	x--x-x-x	0.190
64	<rk>	x--xx-xx	0.171
65	<rk>	-xx--x-x	0.165
66	<rk>	x--xxxxx	0.156
67	<rk>	-xx-x-xx	0.146
68	<rk>	-x-xx--x	0.140
69	<rk>	x---x--x	0.136
70	<rk>	-xx-xxxx	0.132
71	<rk>	x-----x	0.109
72	<rk>	-x-x---x	0.109
73	<rk>	x--x--xx	0.101
74	<rk>	-x-xxx-x	0.0976
75	<rk>	x---xx-x	0.0761
76	<rk>	-xx---xx	0.0761
77	<rk>	x--x-xxx	0.0732
78	<rk>	-x-x-x-x	0.0507
79	<rk>	-xx--xxx	0.0488
80	<rk>	x----x-x	0.0273
81	<rk>	-x-xx-xx	0.0244
82	<rk>	-x--x--x	0
83	<rk>	--xxx--x	0
84	<rk>	-x-xxxxx	0
85	<rk>	x---x-xx	-0.0127
86	<rk>	--xx---x	-0.0273
87	<rk>	-x-----x	-0.0312
88	<rk>	-x-x--xx	-0.0381
89	<rk>	x---xxxx	-0.0488
90	<rk>	--xxxx-x	-0.0488
91	<rk>	-x--xx-x	-0.0634
92	<rk>	-x-x-xxx	-0.0732
93	<rk>	x-----xx	-0.0818
94	<rk>	--xx-x-x	-0.0888
95	<rk>	-x---x-x	-0.109
96	<rk>	--xxx-xx	-0.122

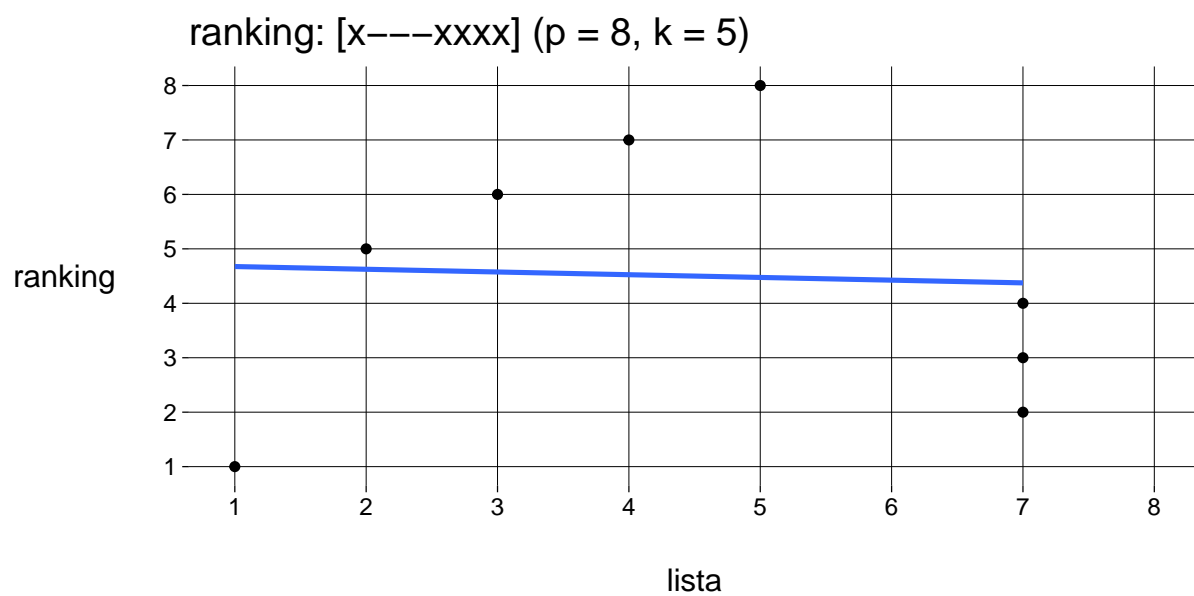
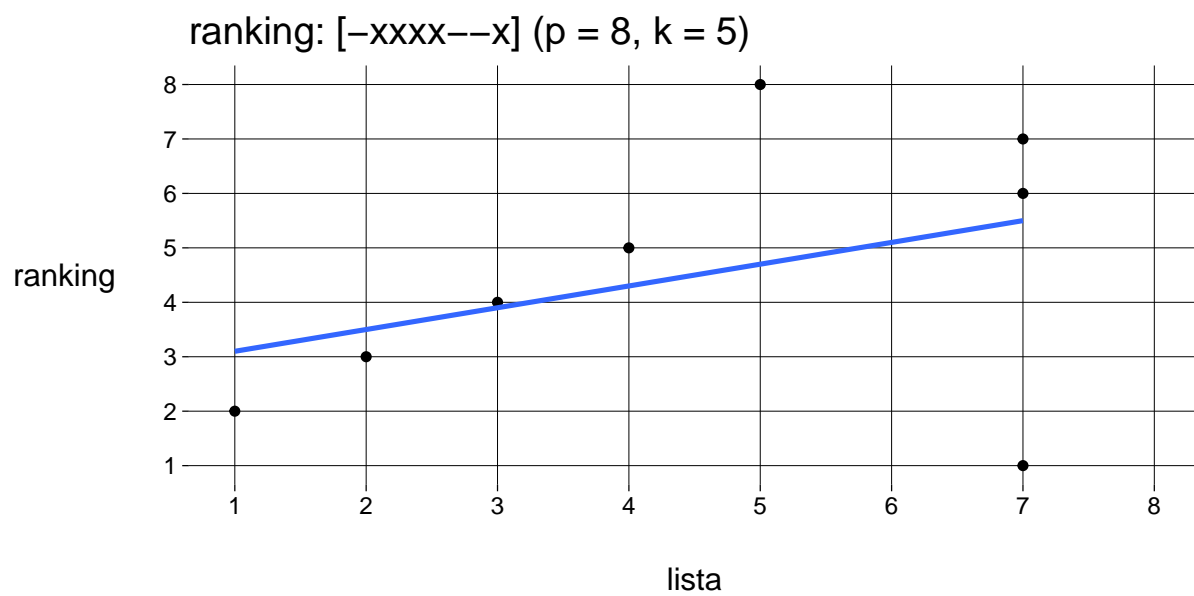
97	<rk>	x----xxx	-0.127
98	<rk>	--x-x--x	-0.136
99	<rk>	-x--x-xx	-0.152
100	<rk>	--xxxxxx	-0.156
101	<rk>	--x----x	-0.171
102	<rk>	--xx--xx	-0.178
103	<rk>	-x--xxxx	-0.195
104	<rk>	--x-xx-x	-0.203
105	<rk>	-x----xx	-0.218
106	<rk>	--xx-xxx	-0.220
107	<rk>	--x--x-x	-0.245
108	<rk>	-x---xxx	-0.266
109	<rk>	---xx--x	-0.273
110	<rk>	--x-x-xx	-0.292
111	<rk>	---x---x	-0.312
112	<rk>	--x-xxxx	-0.342
113	<rk>	---xxx-x	-0.342
114	<rk>	--x---xx	-0.355
115	<rk>	---x-x-x	-0.382
116	<rk>	--x--xxx	-0.406
117	<rk>	---xx-xx	-0.431
118	<rk>	----x--x	-0.452
119	<rk>	---xxxxx	-0.488
120	<rk>	---x--xx	-0.491
121	<rk>	----xx-x	-0.518
122	<rk>	---x-xxx	-0.545
123	<rk>	-----x	-0.577
124	<rk>	-----x-x	-0.592
125	<rk>	-----x-xx	-0.627
126	<rk>	-----xxxx	-0.685
127	<rk>	-----xx	-0.733
128	<rk>	-----xxx	-0.764

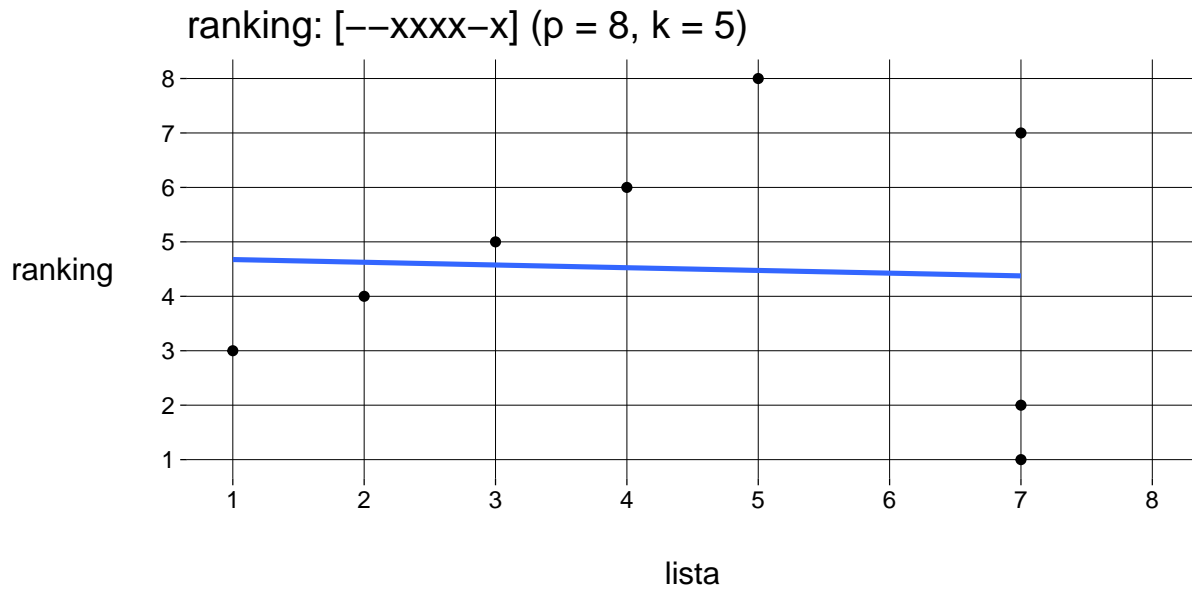
Perceba que pode haver empates:

```
# A tibble: 15 x 2
  ranking_str      s
  <chr>          <dbl>
1 x-xx--xx      0.390
2 -xxxx--x      0.390
3 xx---xxx      0.317
4 -xxx-x-x      0.317
```

5	x--xxx-x	0.244
6	-xxx--xx	0.244
7	x-----x	0.109
8	-x-x---x	0.109
9	x---xx-x	0.0761
10	-xx---xx	0.0761
11	-x--x--x	0
12	--xxx--x	0
13	-x-xxxxx	0
14	x---xxxx	-0.0488
15	--xxxx-x	-0.0488







### 3.0.1 Comparando *rankings* com valores diferentes de $p$

Como a lista é dada e fixa, só faz sentido, na prática, comparar *rankings* com o mesmo valor de  $k$ .

Vamos examinar, para uma lista com  $k = 15$ , os *rankings* possíveis com  $p$  variando de 15 a 20.

São 15.504 *rankings*. Eis os 100 melhores:

```
# A tibble: 100 x 4
  ranking ranking_str      s      p
  <list>   <chr>         <dbl> <int>
1 <rk>     xxxxxxxxxxxxxxxx      1      15
2 <rk>     xxxxxxxxxxxxxxxx-x  0.997      16
3 <rk>     xxxxxxxxxxxxxxxx--x  0.992      17
4 <rk>     xxxxxxxxxxxxxxxx-xx  0.991      16
5 <rk>     xxxxxxxxxxxxxxxx-x-x  0.986      17
6 <rk>     xxxxxxxxxxxxxxxx---x  0.986      18
7 <rk>     xxxxxxxxxxxxxxxx-xxx  0.982      16
8 <rk>     xxxxxxxxxxxxxxxx--xx  0.980      17
9 <rk>     xxxxxxxxxxxxxxxx-x--x  0.979      18
10 <rk>     xxxxxxxxxxxxxxxx----x  0.978      19
11 <rk>     xxxxxxxxxxxxxxxx-xx-x  0.977      17
12 <rk>     xxxxxxxxxxxxxxxx--x-x  0.973      18
```

13	<rk>	xxxxxxxxxxxxxxxxx-x---x	0.972	19
14	<rk>	xxxxxxxxxxxxxxxxx-x-xx	0.971	17
15	<rk>	xxxxxxxxxxxxxxxxx-xx--x	0.971	18
16	<rk>	xxxxxxxxxxxxxxxxx-xxxx	0.971	16
17	<rk>	xxxxxxxxxxxxxxxxx-----x	0.970	20
18	<rk>	xxxxxxxxxxxxxxxxx---xx	0.967	18
19	<rk>	xxxxxxxxxxxxxxxxx-xxx-x	0.966	17
20	<rk>	xxxxxxxxxxxxxxxxx--x--x	0.966	19
21	<rk>	xxxxxxxxxxxxxxxxx-x-x-x	0.965	18
22	<rk>	xxxxxxxxxxxxxxxxx-xx---x	0.964	19
23	<rk>	xxxxxxxxxxxxxxxxx-x-----x	0.964	20
24	<rk>	xxxxxxxxxxxxxxxxx--xxx	0.963	17
25	<rk>	xxxxxxxxxxxxxxxxx-xxx--x	0.961	18
26	<rk>	xxxxxxxxxxxxxxxxx-xx-xx	0.960	17
27	<rk>	xxxxxxxxxxxxxxxxx---x-x	0.959	19
28	<rk>	xxxxxxxxxxxxxxxxx-x--xx	0.959	18
29	<rk>	xxxxxxxxxxxxxxxxx-x-x--x	0.958	19
30	<rk>	xxxxxxxxxxxxxxxxx--x---x	0.958	20
31	<rk>	xxxxxxxxxxxxxxxxx--xx-x	0.957	18
32	<rk>	xxxxxxxxxxxxxxxxx-xx-----x	0.956	20
33	<rk>	xxxxxxxxxxxxxxxxx-xxxxx	0.956	16
34	<rk>	xxxxxxxxxxxxxxxxx-xx-x-x	0.955	18
35	<rk>	xxxxxxxxxxxxxxxxx-xxx---x	0.954	19
36	<rk>	xxxxxxxxxxxxxxxxx-----xx	0.953	19
37	<rk>	xxxxxxxxxxxxxxxxx-xxxx-x	0.953	17
38	<rk>	xxxxxxxxxxxxxxxxx-x-xxx	0.952	17
39	<rk>	xxxxxxxxxxxxxxxxx-x--x-x	0.952	19
40	<rk>	xxxxxxxxxxxxxxxxx---x--x	0.952	20
41	<rk>	xxxxxxxxxxxxxxxxx--x-xx	0.950	18
42	<rk>	xxxxxxxxxxxxxxxxx-x-x---x	0.950	20
43	<rk>	xxxxxxxxxxxxxxxxx--xx--x	0.950	19
44	<rk>	xxxxxxxxxxxxxxxxx-xx--xx	0.948	18
45	<rk>	xxxxxxxxxxxxxxxxx-xxxx--x	0.948	18
46	<rk>	xxxxxxxxxxxxxxxxx-xx-x--x	0.948	19
47	<rk>	xxxxxxxxxxxxxxxxx-xxx-----x	0.947	20
48	<rk>	xxxxxxxxxxxxxxxxx-xxx-xx	0.947	17
49	<rk>	xxxxxxxxxxxxxxxxx-x-xx-x	0.946	18
50	<rk>	xxxxxxxxxxxxxxxxx-----x-x	0.945	20
51	<rk>	xxxxxxxxxxxxxxxxx-x---xx	0.945	19
52	<rk>	xxxxxxxxxxxxxxxxx-x--x--x	0.944	20
53	<rk>	xxxxxxxxxxxxxxxxx--x-x-x	0.944	19
54	<rk>	xxxxxxxxxxxxxxxxx-xxxx---x	0.943	19
55	<rk>	xxxxxxxxxxxxxxxxx--xx---x	0.942	20

56	<rk>	xxxxxxxxxxxxxxx---xxx	0.942	18
57	<rk>	xxxxxxxxxxxxxxx-xxx-x-x	0.942	18
58	<rk>	xxxxxxxxxxxxxxx-xx--x-x	0.942	19
59	<rk>	xxxxxxxxxxxxxxx-xx-x---x	0.941	20
60	<rk>	xxxxxxxxxxxxxxx--xxxx	0.941	17
61	<rk>	xxxxxxxxxxxxxxx-x-xx--x	0.940	19
62	<rk>	xxxxxxxxxxxxxxx-x-x-xx	0.940	18
63	<rk>	xxxxxxxxxxxxxxx-----xx	0.939	20
64	<rk>	xxxxxxxxxxx-xxxxxx	0.938	16
65	<rk>	xxxxxxxxxxx-xx-xxx	0.938	17
66	<rk>	xxxxxxxxxxxxxxx-x---x-x	0.938	20
67	<rk>	xxxxxxxxxxxxxxx--x--xx	0.937	19
68	<rk>	xxxxxxxxxxx-xxxxx-x	0.937	17
69	<rk>	xxxxxxxxxxxxxxx-xxx-x--x	0.937	19
70	<rk>	xxxxxxxxxxxxxxx--x-x--x	0.936	20
71	<rk>	xxxxxxxxxxxxxxx-xxxx----x	0.936	20
72	<rk>	xxxxxxxxxxxxxxx--xxx-x	0.936	18
73	<rk>	xxxxxxxxxxxxxxx-xxx--xx	0.936	18
74	<rk>	xxxxxxxxxxxxxxx---xx-x	0.936	19
75	<rk>	xxxxxxxxxxxxxxx-xx---xx	0.936	19
76	<rk>	xxxxxxxxxxxxxxx-xx--x--x	0.935	20
77	<rk>	xxxxxxxxxxxxxxx-x-x-x-x	0.934	19
78	<rk>	xxxxxxxxxxxxxxx-xx-xx-x	0.934	18
79	<rk>	xxxxxxxxxxx-xxxxx--x	0.934	18
80	<rk>	xxxxxxxxxxxxxxx-x-xx---x	0.933	20
81	<rk>	xxxxxxxxxxxxxxx-x-----xx	0.932	20
82	<rk>	xxxxxxxxxxxxxxx-x--xxx	0.932	18
83	<rk>	xxxxxxxxxxx-xxxx-xx	0.931	17
84	<rk>	xxxxxxxxxxxxxxx--xxx--x	0.930	19
85	<rk>	xxxxxxxxxxxxxxx-xxx--x-x	0.930	19
86	<rk>	xxxxxxxxxxxxxxx--x--x-x	0.930	20
87	<rk>	xxxxxxxxxxxxxxx-xxx-x---x	0.930	20
88	<rk>	xxxxxxxxxxxxxxx--xx-xx	0.930	18
89	<rk>	xxxxxxxxxxxxxxx---x-xx	0.930	19
90	<rk>	xxxxxxxxxxx-xxxxx---x	0.930	19
91	<rk>	xxxxxxxxxxxxxxx---xx--x	0.929	20
92	<rk>	xxxxxxxxxxxxxxx-xx---x-x	0.929	20
93	<rk>	xxxxxxxxxxxxxxx-xx-xx--x	0.929	19
94	<rk>	xxxxxxxxxxxxxxx-x-x--xx	0.928	19
95	<rk>	xxxxxxxxxxxxxxx-xx-x-xx	0.928	18
96	<rk>	xxxxxxxxxxx-xxxx-x-x	0.928	18
97	<rk>	xxxxxxxxxxxxxxx-x-x-x--x	0.927	20
98	<rk>	xxxxxxxxxxx-x-xxxx	0.927	17

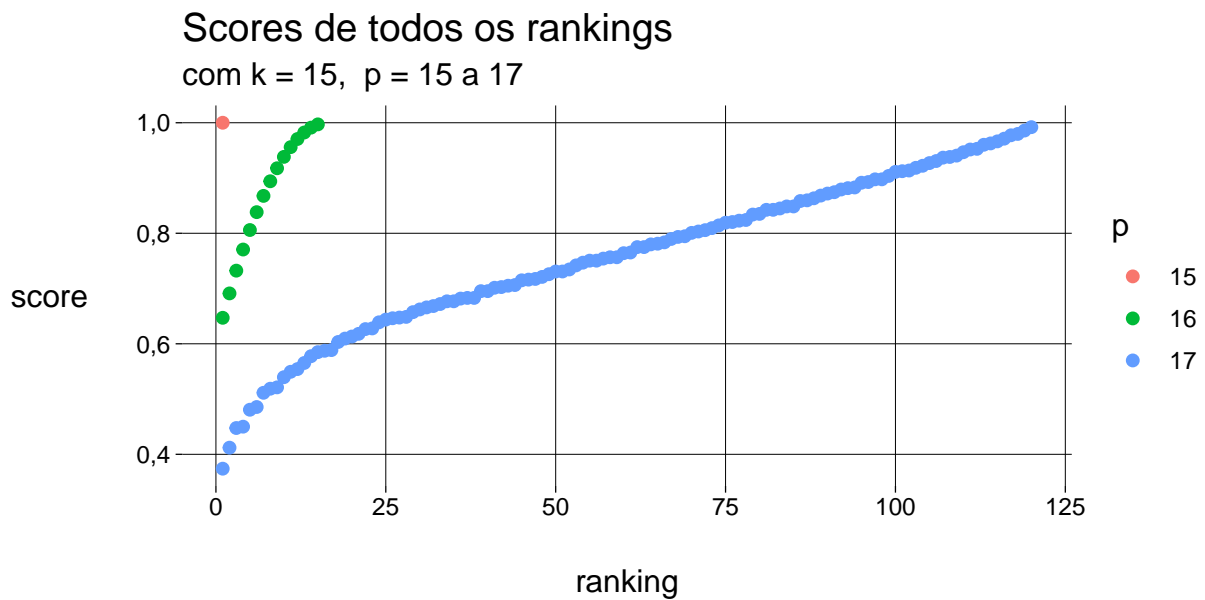
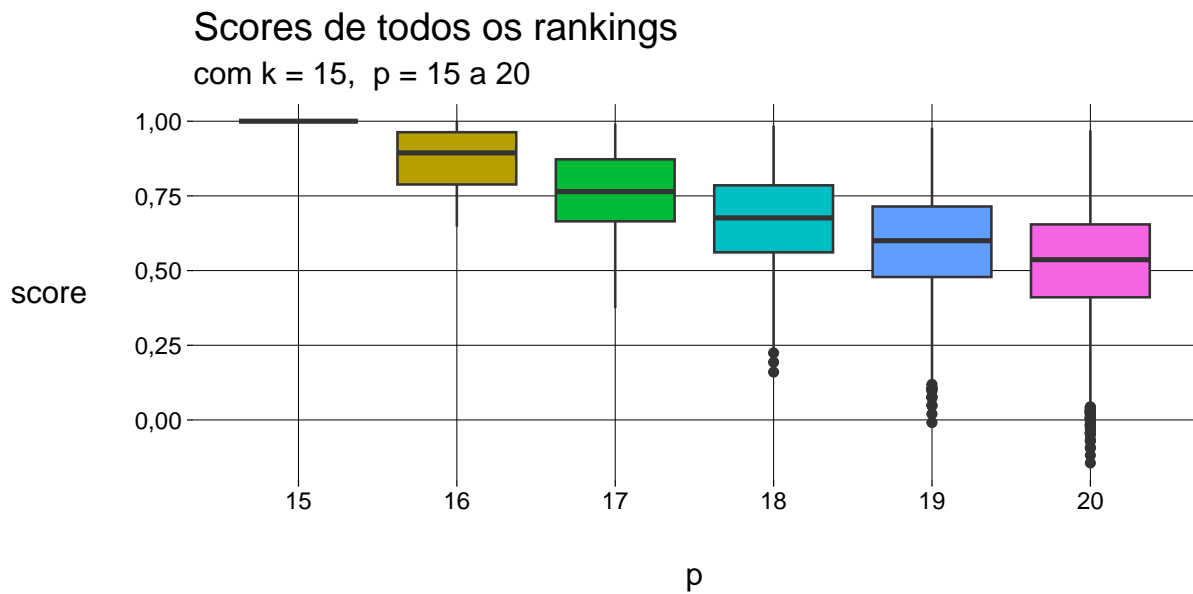


```

99 <rk>      xxxxxxxxxxxx-x--xx-x  0.926   19
100 <rk>      xxxxxxxxxxxx--x---xx  0.924   20

```

Os gráficos abaixo mostram os *scores* atribuídos para todos os *rankings* com  $k = 15$  e  $p$  variando de 15 a 20, separados por valores de  $p$ :





Negativos:

