

Comparando listas e rankings

Fernando Náufel

16/02/2024 16:50

Índice

| | |
|--|-----------|
| Apresentação | 3 |
| 1 Listas e <i>rankings</i> | 4 |
| 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 |
| 2 O <i>ranking</i> concorda com a lista? Posições | 13 |
| 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 |

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

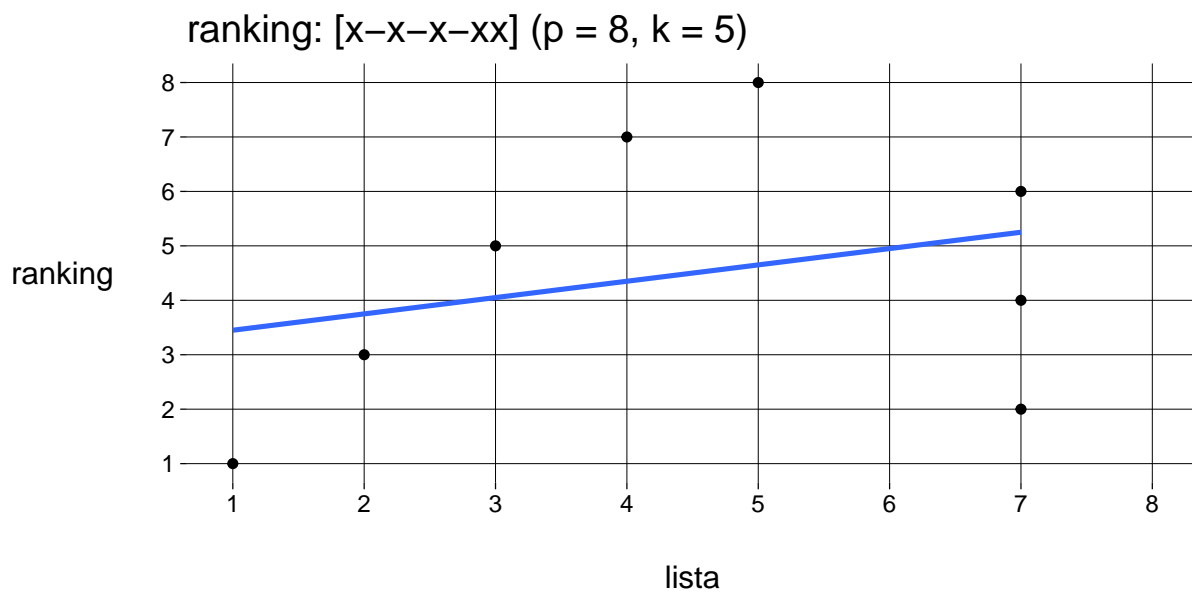
1.3.1 Mostrando um *ranking* graficamente

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.

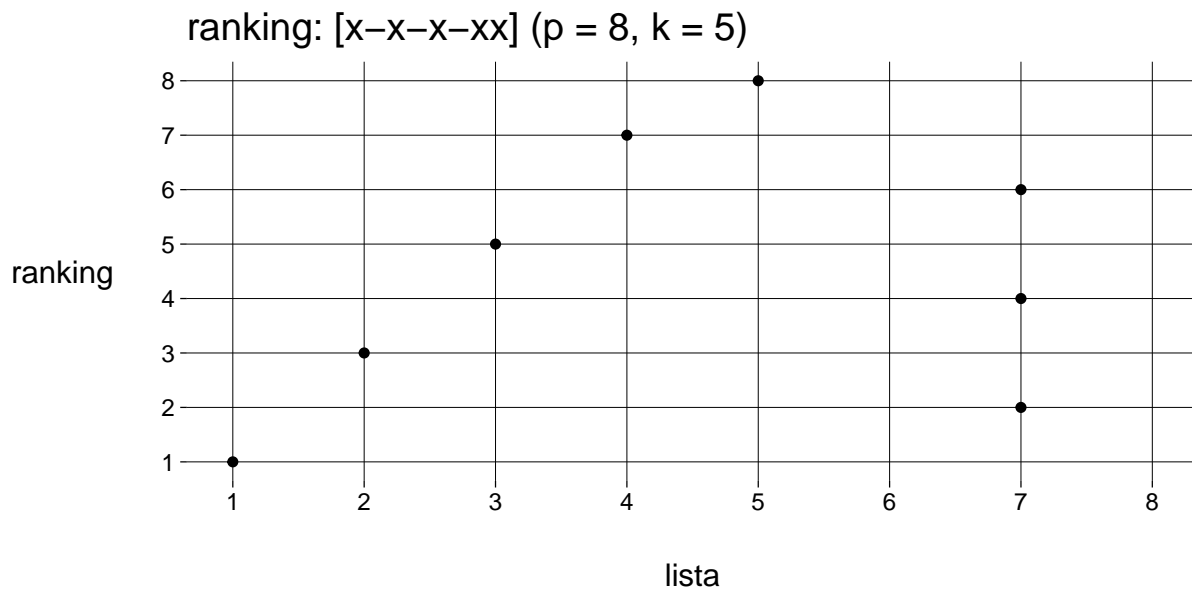
No eixo *y*, a posição do elemento no *ranking*.

```
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`.

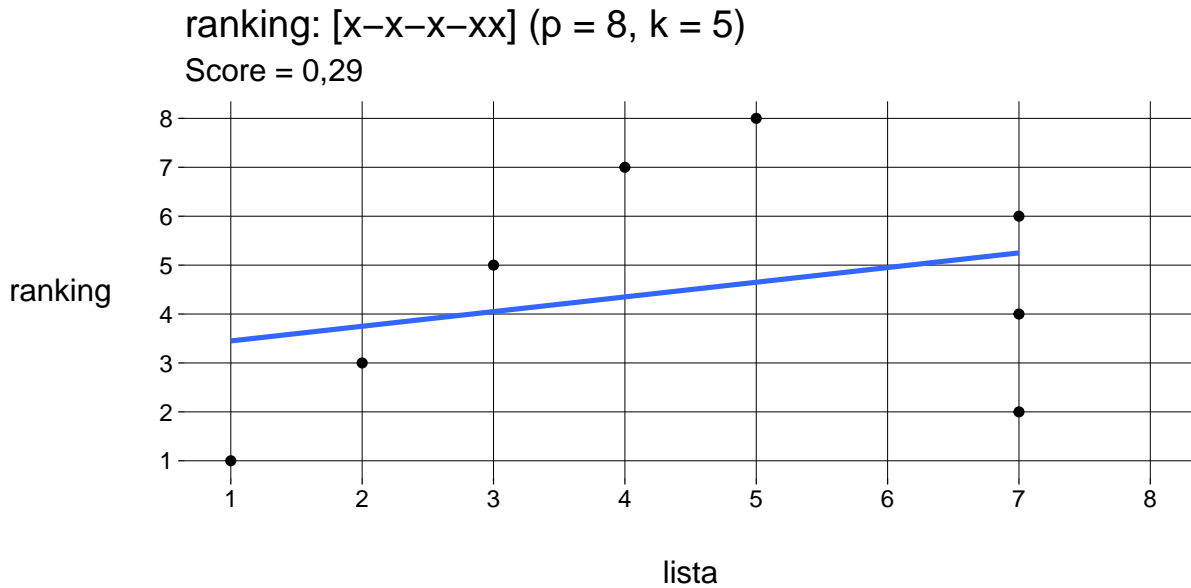
```
plot(r, reta = FALSE)
```



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 um objeto `rk` e retornar o *score* numérico.

```
plot(
  r,
  fun = \(r) {
    df <- as_tibble(r)
    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---xx
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 α 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

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

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:

| ranking_str | s |
|-------------|-----------|
| xxxxxxxx | 1,0000000 |
| xxxxxxx-x | 0,9000000 |
| xxxxxx-xx | 0,8964286 |
| xxxxx-xxx | 0,8928571 |
| xxx-xxxx | 0,8892857 |
| xx-xxxxx | 0,8857143 |
| x-xxxxxx | 0,8821429 |
| -xxxxxxx | 0,8785714 |
| xxxxx-x | 0,7964286 |
| xxxx-x-x | 0,7928571 |
| xxxx-xx | 0,7892857 |
| xxx-xx-x | 0,7892857 |
| xxx-x-xx | 0,7857143 |
| xx-xxx-x | 0,7857143 |
| xxx-xxx | 0,7821429 |
| xx-xx-xx | 0,7821429 |
| x-xxxx-x | 0,7821429 |
| xx-x-xxx | 0,7785714 |
| x-xxx-xx | 0,7785714 |
| -xxxxx-x | 0,7785714 |
| xx-xxxx | 0,7750000 |
| x-xx-xxx | 0,7750000 |
| -xxxx-xx | 0,7750000 |
| x-x-xxxx | 0,7714286 |
| -xxx-xxx | 0,7714286 |
| x-xxxxx | 0,7678571 |
| -xx-xxxx | 0,7678571 |
| -x-xxxxx | 0,7642857 |
| -xxxxxx | 0,7607143 |
| xxxx-x | 0,6892857 |
| xxx-x-x | 0,6857143 |
| xxx-x-x | 0,6821429 |
| xx-xx-x | 0,6821429 |
| xxx-xx | 0,6785714 |
| xx-x-x-x | 0,6785714 |
| x-xxx-x | 0,6785714 |
| xx-x-xx | 0,6750000 |
| xx-xx-x | 0,6750000 |
| x-xx-x-x | 0,6750000 |

| | |
|----------|-----------|
| -XXXX-X | 0,6750000 |
| XX-X-XX | 0,6714286 |
| X-XX-XX | 0,6714286 |
| X-X-XX-X | 0,6714286 |
| -XXX-X-X | 0,6714286 |
| XX—XXX | 0,6678571 |
| X-X-X-XX | 0,6678571 |
| X-XXX-X | 0,6678571 |
| -XXX-XX | 0,6678571 |
| -XX-XX-X | 0,6678571 |
| X-X-XXX | 0,6642857 |
| X-XX-XX | 0,6642857 |
| -XX-X-XX | 0,6642857 |
| -X-XXX-X | 0,6642857 |
| X-X-XXX | 0,6607143 |
| -XX-XXX | 0,6607143 |
| -X-XX-XX | 0,6607143 |
| -XXXX-X | 0,6607143 |
| X—XXXX | 0,6571429 |
| -X-X-XXX | 0,6571429 |
| -XXX-XX | 0,6571429 |
| -X-XXXX | 0,6535714 |
| -XX-XXX | 0,6535714 |
| -X-XXXX | 0,6500000 |
| —XXXXX | 0,6464286 |
| XXX—X | 0,5785714 |
| XX-X—X | 0,5750000 |
| XX-X-X | 0,5714286 |
| X-XX—X | 0,5714286 |
| XX—X-X | 0,5678571 |
| X-X-X-X | 0,5678571 |
| -XXX—X | 0,5678571 |
| XX—XX | 0,5642857 |
| X-X-X-X | 0,5642857 |
| X-XX-X | 0,5642857 |
| -XX-X-X | 0,5642857 |
| X-X—XX | 0,5607143 |
| X-X-X-X | 0,5607143 |
| -XX-X-X | 0,5607143 |
| -X-XX-X | 0,5607143 |
| X-X-XX | 0,5571429 |
| X—XX-X | 0,5571429 |
| -XX—XX | 0,5571429 |

| | |
|----------|-----------|
| -X-X-X-X | 0,5571429 |
| -XXX-X | 0,5571429 |
| X—X-XX | 0,5535714 |
| -X-X-XX | 0,5535714 |
| -X-XX-X | 0,5535714 |
| -XX-X-X | 0,5535714 |
| X—-XXX | 0,5500000 |
| -X-X-XX | 0,5500000 |
| -XX-XX | 0,5500000 |
| -X-XX-X | 0,5500000 |
| -X—XXX | 0,5464286 |
| -X-X-XX | 0,5464286 |
| —XXX-X | 0,5464286 |
| -X-XXX | 0,5428571 |
| —XX-XX | 0,5428571 |
| —X-XXX | 0,5392857 |
| —XXXX | 0,5357143 |
| XX—X | 0,4642857 |
| X-X—X | 0,4607143 |
| X-X—X | 0,4571429 |
| -XX—X | 0,4571429 |
| X—X-X | 0,4535714 |
| -X-X—X | 0,4535714 |
| X—X-X | 0,4500000 |
| -X-X-X | 0,4500000 |
| -XX—X | 0,4500000 |
| X—XX | 0,4464286 |
| -X—X-X | 0,4464286 |
| -X-X-X | 0,4464286 |
| -X—XX | 0,4428571 |
| -X-X-X | 0,4428571 |
| —XX-X | 0,4428571 |
| -X—XX | 0,4392857 |
| —X-X-X | 0,4392857 |
| —X-XX | 0,4357143 |
| —XX-X | 0,4357143 |
| —X-XX | 0,4321429 |
| —XXX | 0,4285714 |
| X—X | 0,3464286 |
| -X—X | 0,3428571 |
| -X—X | 0,3392857 |
| —X—X | 0,3357143 |
| —X-X | 0,3321429 |

| | |
|------|-----------|
| —x-x | 0,3285714 |
| —xx | 0,3250000 |
| —x | 0,2250000 |

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 $k = 2$, os *rankings* possíveis com p variando de 2 a 10.

São 45 *rankings*:

| ranking_str | p | s |
|-------------|---|-----------|
| xx | 2 | 1,0000000 |
| x-x | 3 | 0,7333333 |
| -xx | 3 | 0,7000000 |
| x-x | 4 | 0,5833333 |
| -x-x | 4 | 0,5666667 |
| -xx | 4 | 0,5500000 |
| x—x | 5 | 0,4900000 |
| -x-x | 5 | 0,4800000 |
| -x-x | 5 | 0,4700000 |
| —xx | 5 | 0,4600000 |
| x—x | 6 | 0,4266667 |
| -x—x | 6 | 0,4200000 |
| -x-x | 6 | 0,4133333 |
| —x-x | 6 | 0,4066667 |
| —xx | 6 | 0,4000000 |
| x—x | 7 | 0,3809524 |
| -x—x | 7 | 0,3761905 |
| -x—x | 7 | 0,3714286 |
| —x-x | 7 | 0,3666667 |
| —x-x | 7 | 0,3619048 |
| —xx | 7 | 0,3571429 |
| x—x | 8 | 0,3464286 |
| -x—x | 8 | 0,3428571 |
| -x—x | 8 | 0,3392857 |
| —x-x | 8 | 0,3357143 |

| | | |
|------|----|-----------|
| —x-x | 8 | 0,3321429 |
| —x-x | 8 | 0,3285714 |
| —xx | 8 | 0,3250000 |
| x—x | 9 | 0,3194444 |
| -x—x | 9 | 0,3166667 |
| -x—x | 9 | 0,3138889 |
| —x—x | 9 | 0,3111111 |
| —x—x | 9 | 0,3083333 |
| —x-x | 9 | 0,3055556 |
| —x-x | 9 | 0,3027778 |
| —xx | 9 | 0,3000000 |
| x—x | 10 | 0,2977778 |
| -x—x | 10 | 0,2955556 |
| -x—x | 10 | 0,2933333 |
| —x—x | 10 | 0,2911111 |
| —x—x | 10 | 0,2888889 |
| —x-x | 10 | 0,2866667 |
| —x-x | 10 | 0,2844444 |
| —x-x | 10 | 0,2822222 |
| —xx | 10 | 0,2800000 |

Os gráficos abaixo mostram os *scores* atribuídos para todos os *rankings* com $k = 2$ e p variando de 2 a 10, separados por valores de p :

