Distribuição Binominal

Para que uma situação possa se enquadrar em uma distribuição binomial, deve atender as seguintes condições:

- são realizadas n repetições (tentativas) independentes;
- cada tentativa é uma prova de Bernoulli (somente podem ocorrer dois possíveis resultados); e
- a probabilidade p de sucesso em cada prova é constante.

Se uma situação atende a todas as condições anteriores, então a variável aleatória X = número de sucessos obtidos nas n tentativas terá uma distribuição binomial com n tentativas e p probabilidades de sucesso.

Simbolicamente, temos: $X \sim B\ (n, p)$ com a interpretação:

A variável aleatória X tem distribuição binomial (B) com n ensaios e uma probabilidade p de sucesso (em cada ensaio).

A função binomial de probabilidade é expressa como:

$$\boldsymbol{P(X = x) = C_x^n p^x (1 - p)^{n-x}}$$

$\boldsymbol{P(X = x)}$ – é a probabilidade de x sucessos em n ensaios;

$\boldsymbol{n}$ – é o número de ensaios;

p é probabilidade de "sucesso" em cada ensaio;

$q = 1 - p$ é a probabilidade de "fracasso" em cada ensaio;

$C_x^n = \binom{n}{x} = \frac{n!}{x!(n-x)!}$ - combinação de n valores tomados de x a x.

Como o R e a maioria das linguagens de programação processo isso?

Basta usar as funções disponíveis nas libs ou pacotes.

Esta função gera uma distribuição binominal randomizada.

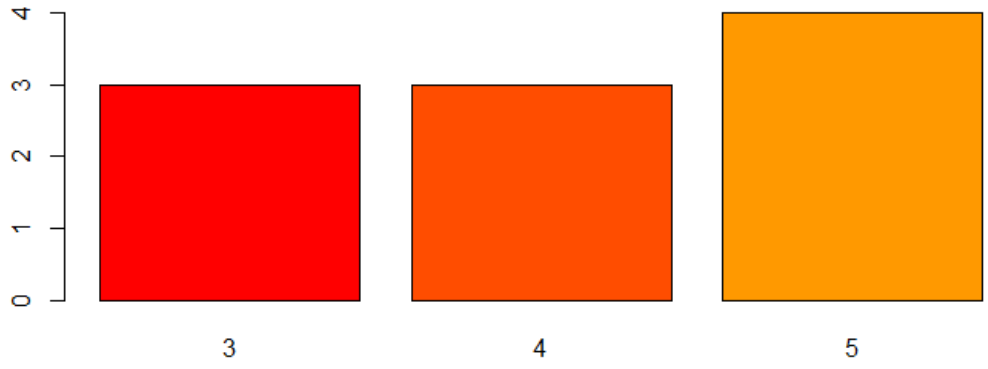
```
rbinom <- function(n,k,p) #n: Tamanho da amostra; k: número de repetiç
ões; p: prob. de sucesso


Test_bin <- rbinom(20, 5, 0.7)

barplot(table(Test_bin), col = rainbow(20))
```

Test_bin

[1] 3 5 5 4 4 5 4 5 3 5 3 4 3 5 4 5 4 2 2 4



Colors:

Rainbow(arco-íris)

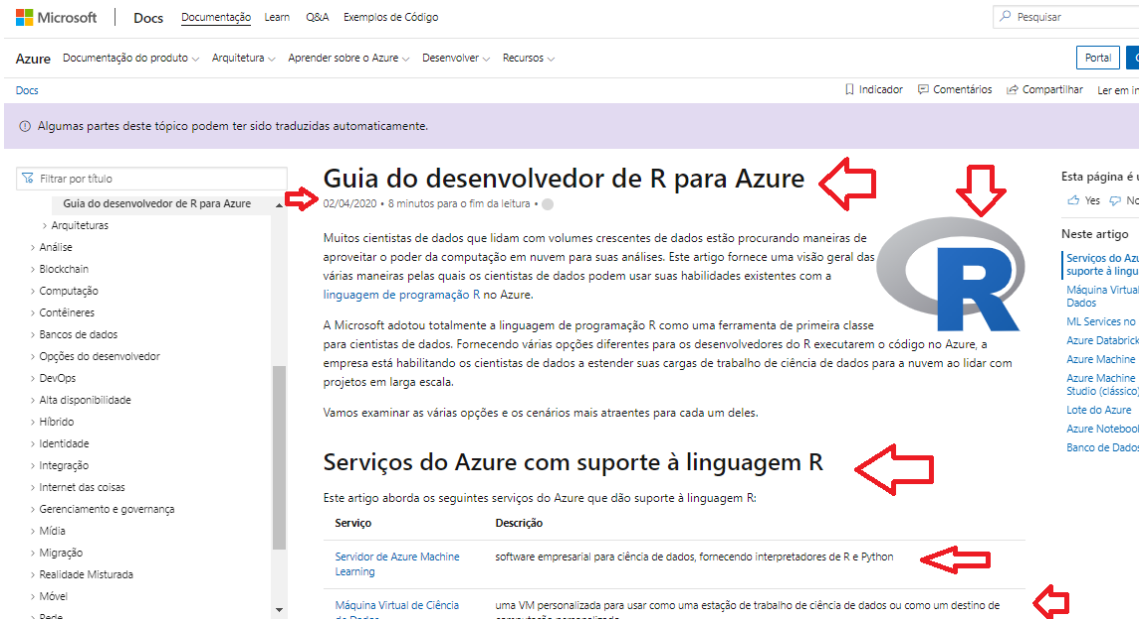
São as cores contíguas do arco íris.

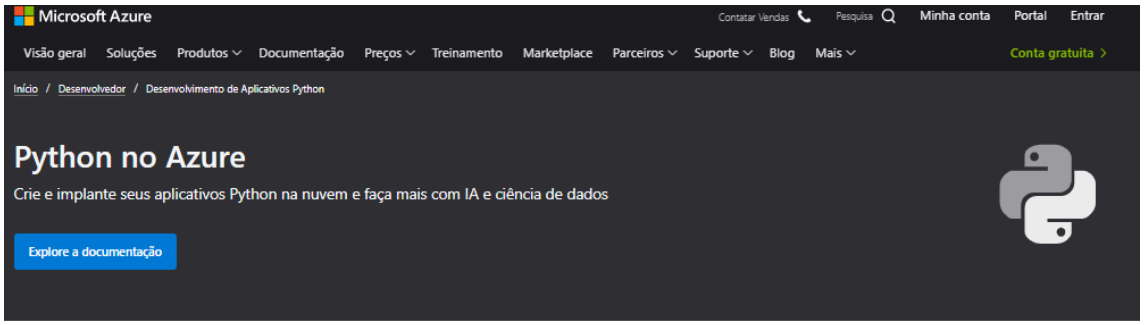
Função rainbow(n)

Você também pode gerar um vetor de n cores contíguas usando as funções **rainbow (n)**

Miscellaneous

Por que usar R e Python?





Por que Python no Azure?

Crie aplicativos Web Python na nuvem

Crie aplicativos Web melhores, mais rapidamente, com nossa plataforma de aplicativos gerenciada otimizada para Python. Conecte seus aplicativos aos dados usando serviços do Azure para bancos de dados relacionais e não relacionais populares.

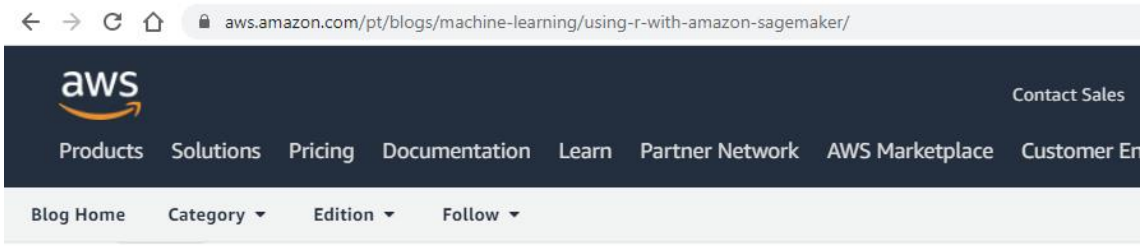
Plataforma flexível para IA e aprendizado de máquina com Python

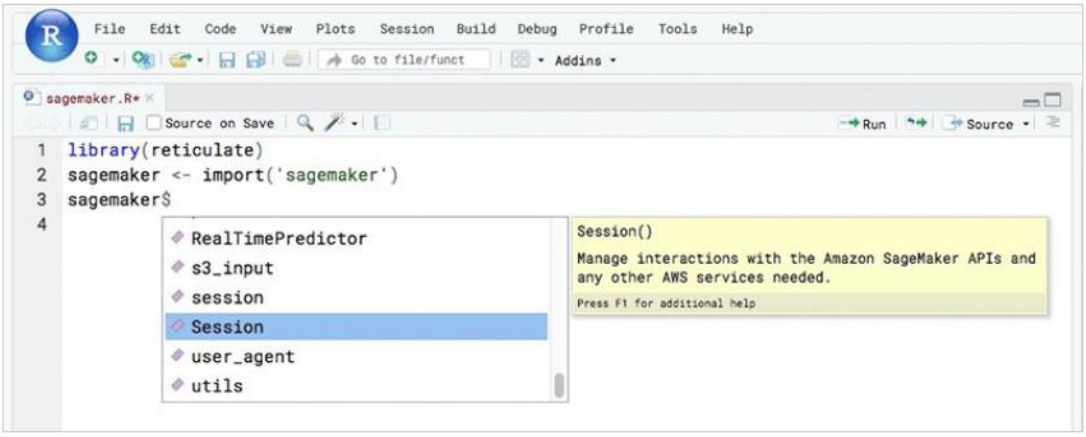
Crie, treine, hospede e implante modelos de forma rápida e fácil em qualquer ambiente Python, com serviços do Azure para ciência de dados e aprendizado de máquina. Ou traga soluções de IA pré-criadas para oferecer experiências de ponta aos seus aplicativos Python.

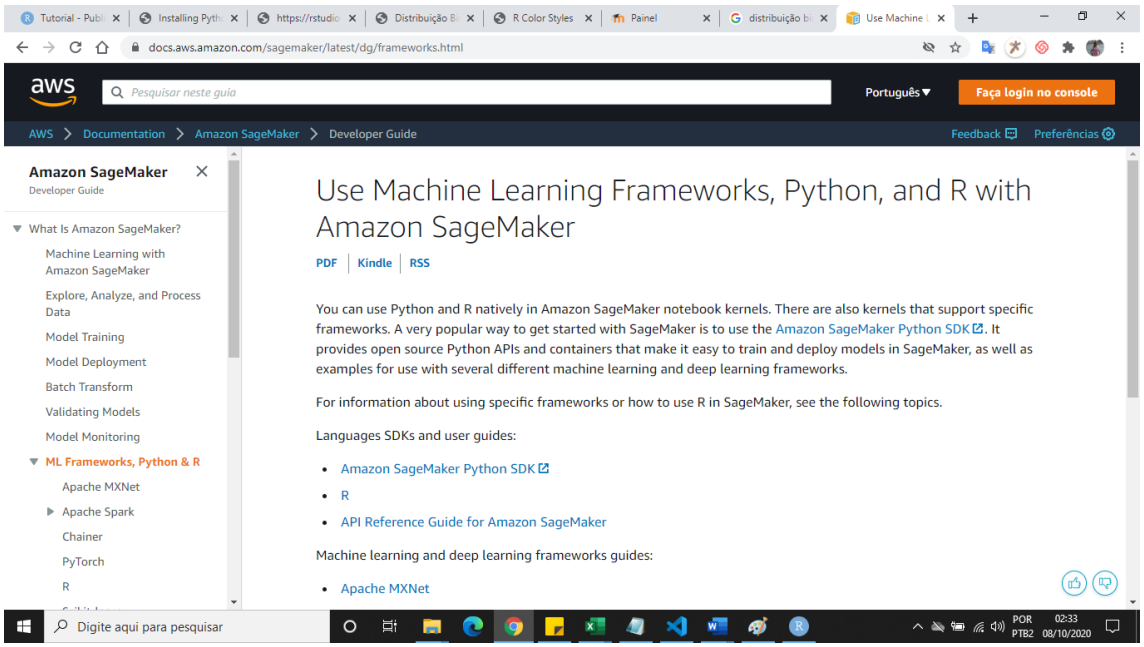
Nossas ferramentas para desenvolvimento em Python ou as suas ferramentas

Crie e depure seus aplicativos Python com o Visual Studio Code e envie-os à nuvem por push com apenas alguns cliques. Depois, experimente o Azure DevOps baseado em nuvem e adote um ciclo de vida completo de DevOps para seus aplicativos Python. Ou traga as ferramentas com as quais você está acostumado.

Miscellaneous







Set.seed ()

a função `set.seed` pode ser usada para controlar o comportamento do gerador de números aleatórios. Esta função define o valor inicial da semente que é mudado a cada geração subsequente de números aleatórios. Portanto para gerar duas amostras idênticas basta usar o comando `set.seed` conforme ilustrado abaixo.

```
> set.seed(214)      # define o valor da semente
> rnorm(5)           # amostra de 5 elementos
[1] -0.46774980  0.04088223  1.00335193  2.02522505  0.30640096
> rnorm(5)           # outra amostra de 5 elementos
[1]  0.4257775  0.7488927  0.4464515 -2.2051418  1.9818137
> set.seed(214)      # retorna o valor da semente ao valor inicial
> rnorm(5)           # gera novamente a primeira amostra de 5 elementos
[1] -0.46774980  0.04088223  1.00335193  2.02522505  0.30640096
```

No comando acima mostramos que depois da primeira amostra ser retirada a semente é mudada e por isto os elementos da segunda amostra são diferentes dos da primeira. Depois retornamos a semente ao seu estado original a a próxima amostra tem portanto os mesmos elementos da primeira.

Referências:

Fonte: http://rstudio-pubs-static.s3.amazonaws.com/452495_b637f8d6f34e43b9bdace49bfe1ae230.html

http://www.leg.ufpr.br/Rpira/Rpira/node9.html