UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Bacharelado em Ciência da Computação
Cálculo Numérico

Estimativa de Público

Baseado em Dados da Semana da Computação 2017

Aluno: Eduardo Carneiro DRE:113149505
Professor: João Paixão Data: 26/11/2017

Sumário

Introdução

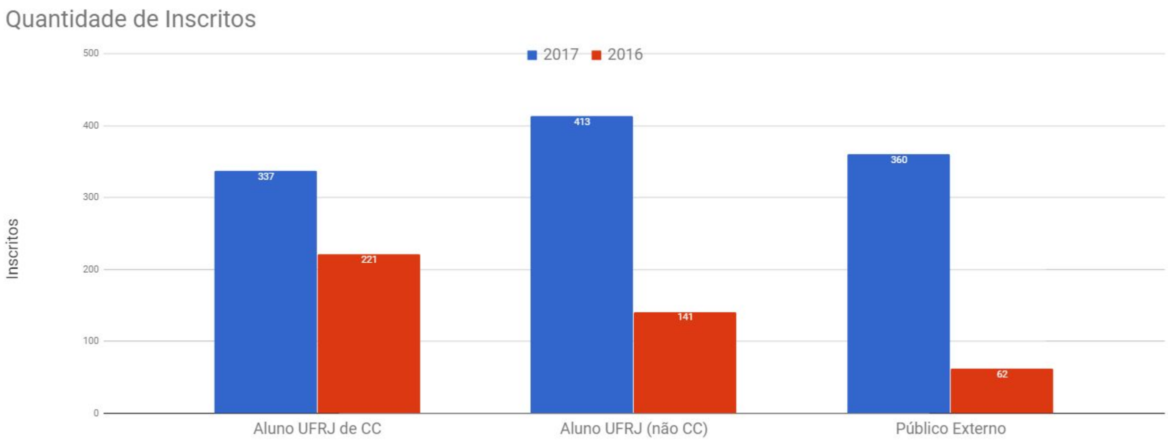
Organizar um evento é sempre um desafio, pois nunca se sabe quais as dificuldades que serão encontradas. Desde a logística até a idealização da programação, o fato de não se saber quantas pessoas comparecerão ao evento é o foco desses desafios, pois é baseado nesse número que se tomam quase todas as decisões.

Geralmente é feita uma estimativa de quantas pessoas comparecerão, mas ela é arbitrária e imprecisa, o que não colabora muito quando o assunto é organizar um evento sem gastos exorbitantes e nem faltas.

Tendo em vista esse desafio, decidimos utilizar o conhecimento adquirido na disciplina de Cálculo Numérico para criar uma estimativa mais precisa, de forma a conseguir prever com mais precisão a quantidade de pessoas que se pode esperar em um evento, baseado no comportamento das inscrições de edições anteriores.

Informações Preliminares

Este ano tivemos a III Semana da Computação UFRJ, um evento anual organizado por alunos do curso de Ciência da Computação em conjunto com alguns professores do departamento, e contamos com um público surpreendente. Este ano, o evento aconteceu do dia 18 de setembro à 22 de setembro.



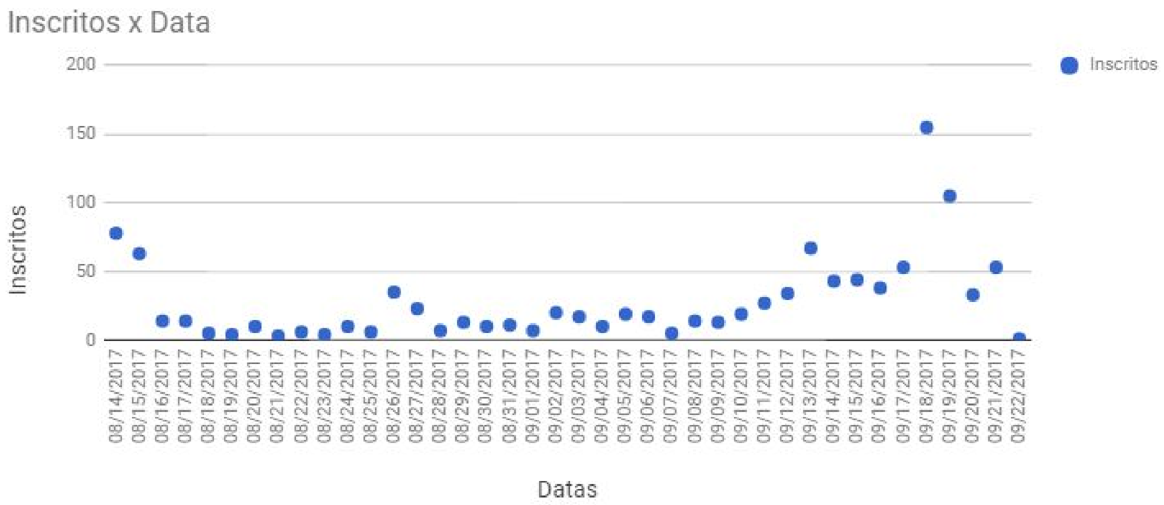
Como pode-se ver no gráfico acima, a quantidade de inscritos na edição de 2017 (em azul) em relação À edição de 2016 (em vermelho) foi quase consideravelmente maior, pois tivemos 422 inscritos em 2016 e 1110 inscritos em 2017.

Sendo assim, estruturamos os dados obtidos na edição deste ano, de forma a conseguirmos estimar o público da próxima edição. Nós abrimos as inscrições 35 dias antes do evento, somado aos cinco dias do evento, obtivemos o número de inscritos por dia até o término do evento, como pode ser visto na tabela a seguir:

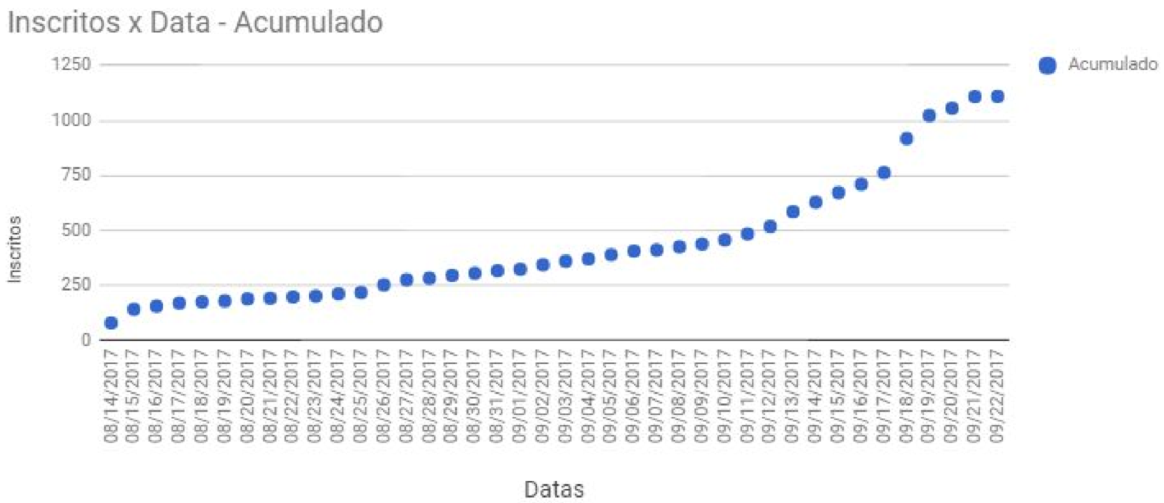
Dados Iniciais

Datas	Inscritos	Datas	Inscritos	Datas	Inscritos	Datas	Inscritos
08/14/2017	78	08/24/2017	10	09/03/2017	17	09/13/2017	67
08/15/2017	63	08/25/2017	6	09/04/2017	10	09/14/2017	43
08/16/2017	14	08/26/2017	35	09/05/2017	19	09/15/2017	44
08/17/2017	14	08/27/2017	23	09/06/2017	17	09/16/2017	38
08/18/2017	5	08/28/2017	7	09/07/2017	5	09/17/2017	53
08/19/2017	4	08/29/2017	13	09/08/2017	14	09/18/2017	155
08/20/2017	10	08/30/2017	10	09/09/2017	13	09/19/2017	105
08/21/2017	3	08/31/2017	11	09/10/2017	19	09/20/2017	33
08/22/2017	6	09/01/2017	7	09/11/2017	27	09/21/2017	53
08/23/2017	4	09/02/2017	20	09/12/2017	34	09/22/2017	1

Para analisarmos o comportamento das inscrições, não bastava expressar em um gráfico a quantidade de inscritos por dia, pois não existe um comportamento padrão para isso.



Mas se analisarmos a quantidade de inscritos acumulada, ou seja, analisar quantos inscritos tem por dia ao invés de contar quantos se inscreveram por dia, podemos ver um comportamento mais matemático.



O comportamento das inscrições nos lembra uma equação de terceiro grau, ou até mesmo uma sigmoidal, mas descobrir quais são os coeficientes dessa equação que modela o nosso caso é um desafio de cálculo numérico. Somente após essa descoberta, conseguiremos usar o mesmo comportamento para estimar as futuras edições.

Para os cálculos, nós eliminamos os cinco últimos valores do gráfico, pois se referem aos dias do evento, para que pudéssemos estimar quantos teríamos até iniciar o evento, mas o código e o método é o mesmo, então para estimar com esses valores, basta inseri-los nos vetores e matrizes.

O Método Numérico

Para dar continuidade à solução do nosso problema, precisamos definir qual o melhor método numérico para o nosso caso. Como temos uma quantidade grande de dados e o nosso objetivo é encontrar uma equação que descreve o comportamento dos dados, os métodos da Bissecção, Diferenças finitas e Interpolação não são muito apropriadas.

Escolhemos utilizar nesta pesquisa o método Mínimos Quadrados. Sabemos que utilizamos este método quando temos uma distribuição de pontos e queremos ajustar a melhor curva a este conjunto de dados, mas precisamos de um modelo de equação para aplicarmos o método, a fim de encontrar os coeficientes da equação e obter a função que descreve o comportamento do nosso caso.

Para que esta seja a curva que melhor se ajusta aos dados, devemos minimizar a soma das diferenças entre os valores de $f(x)$ tabelados b_i e os valores da curva de ajuste $a^2x + bx + c$ em cada ponto.

O método Mínimos Quadrados pode ser resumido na fórmula: $A^T * A * w = A^T * b$, onde *b* é uma matriz coluna que contém a quantidade de inscrições acumuladas e *A* é uma matriz cuja primeira coluna é toda preenchida com 1, a segunda são os pontos do gráfico do eixo x(cada dia corresponde a 0.1), e nas colunas seguintes são as potências da coluna 2, ou seja, segunda coluna ao quadrado, segunda coluna ao cubo etc. Se modelarmos como um polinômio de segundo grau, usaremos três colunas, se de terceiro grau, quatro colunas.

Sendo assim, nossas matrizes ficaram da seguinte forma:

A =	1.	0.1	0.01	0.001	B =	78.
	1.	0.2	0.04	0.008		141.
	1.	0.3	0.09	0.027		155.
	1.	0.4	0.16	0.064		169.
	1.	0.5	0.25	0.125		174.
	1.	0.6	0.36	0.216		178.
	1.	0.7	0.49	0.343		188.
	1.	0.8	0.64	0.512		191.
	1.	0.9	0.81	0.729		197.
	1.	1.	1.	1.		201.
	1.	1.1	1.21	1.331		211.
	1.	1.2	1.44	1.728		217.
	1.	1.3	1.69	2.197		252.
	1.	1.4	1.96	2.744		275.
	1.	1.5	2.25	3.375		282.
	1.	1.6	2.56	4.096		295.
	1.	1.7	2.89	4.913		305.
	1.	1.8	3.24	5.832		316.
	1.	1.9	3.61	6.859		323.
	1.	2.	4.	8.		343.
	1.	2.1	4.41	9.261		360.
	1.	2.2	4.84	10.648		370.
	1.	2.3	5.29	12.167		389.
	1.	2.4	5.76	13.824		406.
	1.	2.5	6.25	15.625		411.
	1.	2.6	6.76	17.576		425.
	1.	2.7	7.29	19.683		438.
	1.	2.8	7.84	21.952		457.
	1.	2.9	8.41	24.389		484.
	1.	3.	9.	27.		518.
	1.	3.1	9.61	29.791		585.
	1.	3.2	10.24	32.768		628.
	1.	3.3	10.89	35.937		672.
	1.	3.4	11.56	39.304		710.
	1.	3.5	12.25	42.875		763.

Precisamos encontrar os valores de w , então precisamos isolá-lo na fórmula $A^T * A * w = A^T * b$, obtendo a equação:

$$w = (A^T * A)^{-1} * (A^T * b)$$

Ao calcular o valor de w, obtemos um vetor com os valores dos coeficientes da equação. Esses coeficientes serão utilizados na equação final da seguinte forma:

$$Y = w_n * x^{n-1} + w_{n-1} * x^{n-2} + ... + w_2 * x + w_1$$

Depois que obtemos a função e os coeficientes que descrevem as inscrições no evento, é necessário desenvolver um algoritmo que nos dê a estimativa dos outros eventos. Como estamos lidando com dados alterados diariamente, montamos um algoritmo que recebe como parâmetro um vetor coluna com a quantidade de inscritos por dia, referentes aos primeiros dias.

Usaremos os dias da edição anterior para preencher os dias que faltam na nova edição, para manter o modelo de equação, acrescentaremos uma taxa de correção na quantidade de inscrições. Essa taxa é a média das diferenças entre as inscrições passadas e novas a cada dia, da seguinte forma:

```
taxa(1) = (novas_inscricoes(1) - inscricoes_antigas(1))
for i=2:size(novas_inscricoes,1)
	taxa(i) = (taxa(i-1) + (novas_inscricoes(i) - inscricoes_antigas(i))) / 2	// Calcula a média entre as variações de cada dia com a taxa anterior
end
```

Sendo assim, nós usamos como base as inscrições antigas para estimar os próximos dias e, consequentemente, estimar a quantidade de inscritos na véspera do evento.

Resultado

Calculamos o valor de x para dois tipos de polinômio, para analisar qual deles descreve melhor as inscrições do evento:

1) Polinômio de segundo grau:

Usando o modelo de equação $Y = a^2x + bx + c$, desenvolvemos um código no Scilab, que pode ser analisado no Apêndice A.

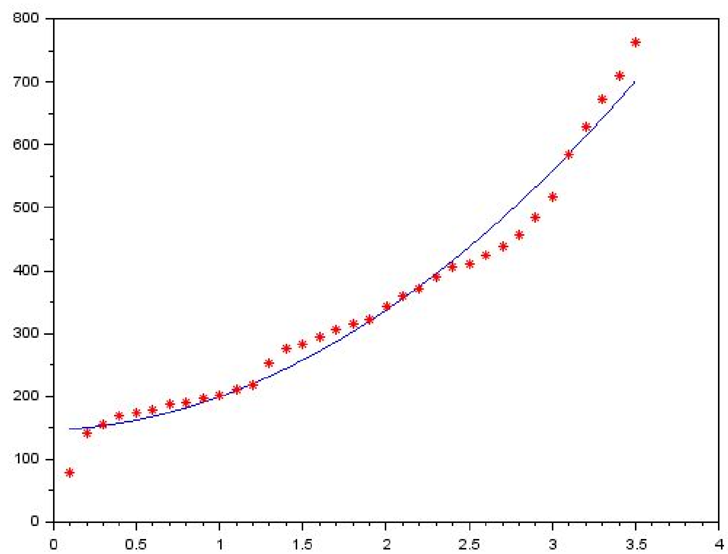
Obtivemos como resultado:

x = 146.23881; 11.08954; 42.186457,

onde cada um dos valores de x corresponde a um coeficiente da equação. Sendo assim, a equação obtida é:

Y = 42.186457*x^2 +11.089540*x +146.238808

Ao plotar o gráfico, obtemos a seguinte imagem, sendo os pontos vermelhos a quantidade de inscrições até o referido o dia e a linha azul, a curva da função Y.



Podemos notar que a equação de segundo grau perde muitos detalhes, não descrevendo bem a distribuição das inscrições.

2) Polinômio de terceiro grau:

Usando o modelo de equação $Y = a^3x + bx^2 + cx + d$, desenvolvemos um código no Scilab, que pode ser analisado no Apêndice B.

Obtivemos como resultado:

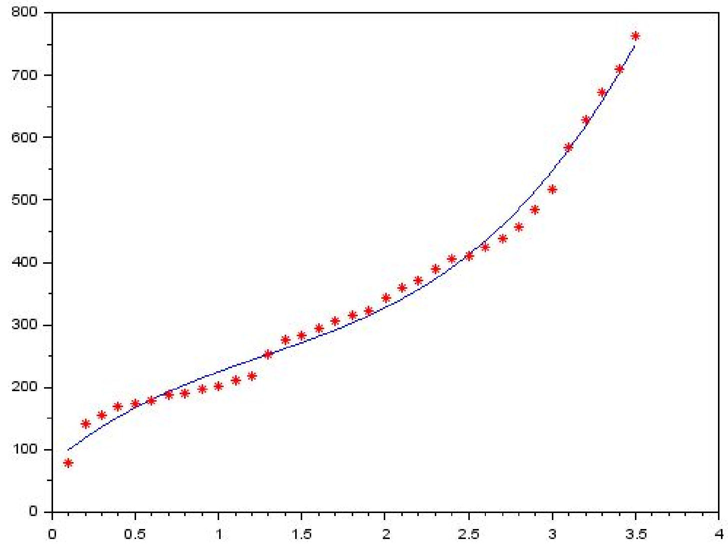
x = 78.64576; 221.7102; -102.03769; 26.708175

,

onde cada um dos valores de x corresponde a um coeficiente da equação. Sendo assim, a equação obtida é:

Y = 26.708175*x^3 -102.037686*x^2 +221.710204*x +78.645760

Ao plotar o gráfico, obtemos a seguinte imagem, sendo os pontos vermelhos a quantidade de inscrições até o referido o dia e a linha azul, a curva da função Y.



Esta equação modela melhor a quantidade de inscritos, consequentemente, nos dará uma estimativa mais precisa.

3) Estimativa de Inscritos

Para obter a estimativa de quantos inscritos teremos na véspera do evento, criamos uma função no Scilab, que recebe como parâmetro de entrada a quantidade de inscritos na nova edição do evento, como pode ser visto no Apêndice C.

Não é necessário ter todos os dias para se calcular a estimativa, basta apenas um dia e a função já começa a mostrar os resultados. Essa função que tem registrada a quantidade de inscrições diárias na última edição, que, somada à taxa explicada na seção anterior, será usada para manter distribuição de acordo com o modelo da edição anterior.

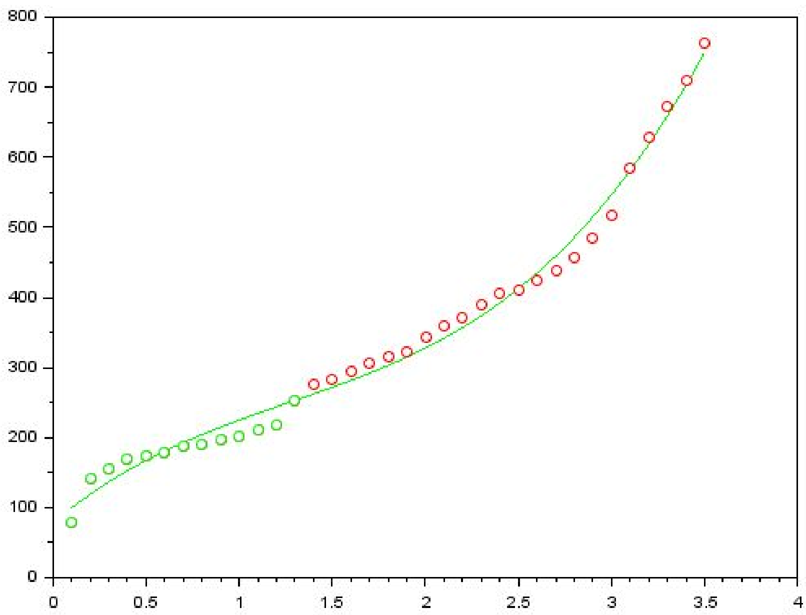
O cálculo da estimativa é o mesmo da equação de terceiro grau, apenas os dados contidos no vetor b, que é a quantidade de inscritos por dia acumulado, foram substituídos pelos valores de entrada da função e acrescidos a taxa nos que não foram alterados baseados na entrada.

Fizemos alguns testes, utilizando diferentes valores de entrada, para simular uma nova edição do evento. Quanto ao gráfico,usamos duas cores diferentes: bolinha verde para os novos dados inseridos, bolinha vermelha para os dados da edição anterior acrescidos da taxa e a linha vermelha para a curva da equação encontrada.

Para demonstrar os testes realizados, usaremos uma função chamada Compara, onde plotaremos a função de terceiro grau com os dados da edição passada e a função da estimativa com novos dados. Os testes são:

Exemplos:

- Se utilizarmos os mesmos dados da edição passada?

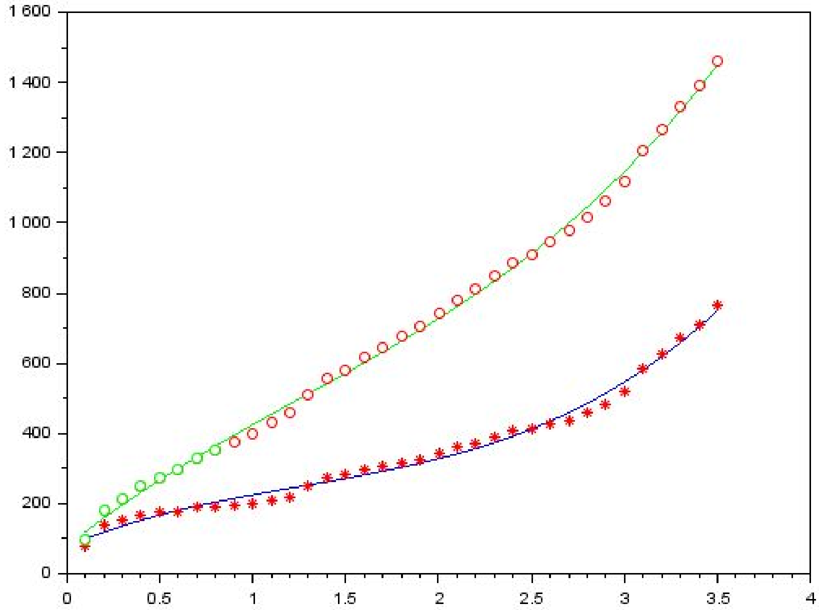


Edição Passada: Y = 26.708175*x^3 -102.037686*x^2 +221.710204*x +78.645760

Nova Edição: Y = 26.708175*x^3 -102.037686*x^2 +221.710204*x +78.645760

Expectativa de público para a véspera: 763

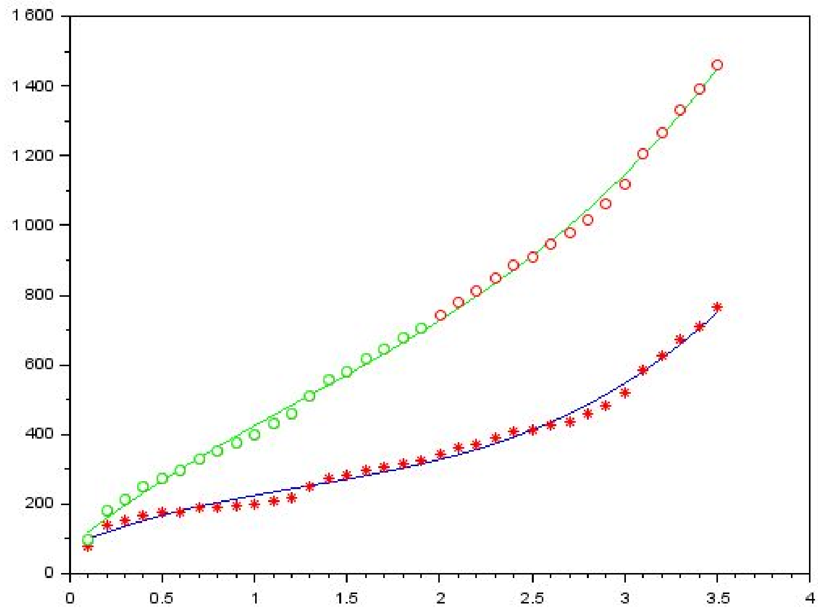
- Se utilizarmos como entrada, o número de inscrições da edição passada acrescido 20 inscrições por dia:



Edição Passada: Y = 26.708175*x^3 -102.037686*x^2 +221.710204*x +78.645760

Nova Edição: Y = 26.708175*x^3 -102.037686*x^2 +421.710204*x +78.645760

Expectativa de público para a véspera: 1450

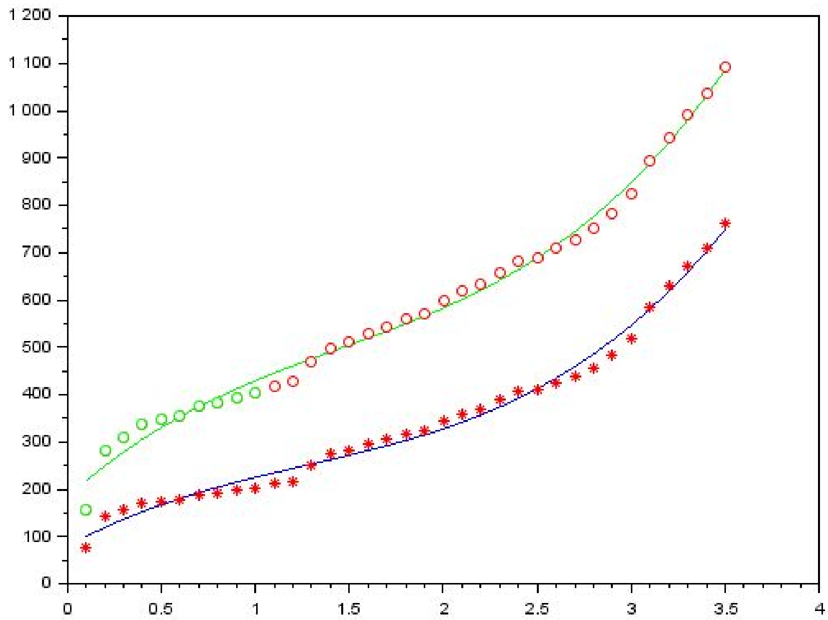


Edição Passada: Y = 26.708175*x^3 -102.037686*x^2 +221.710204*x +78.645760

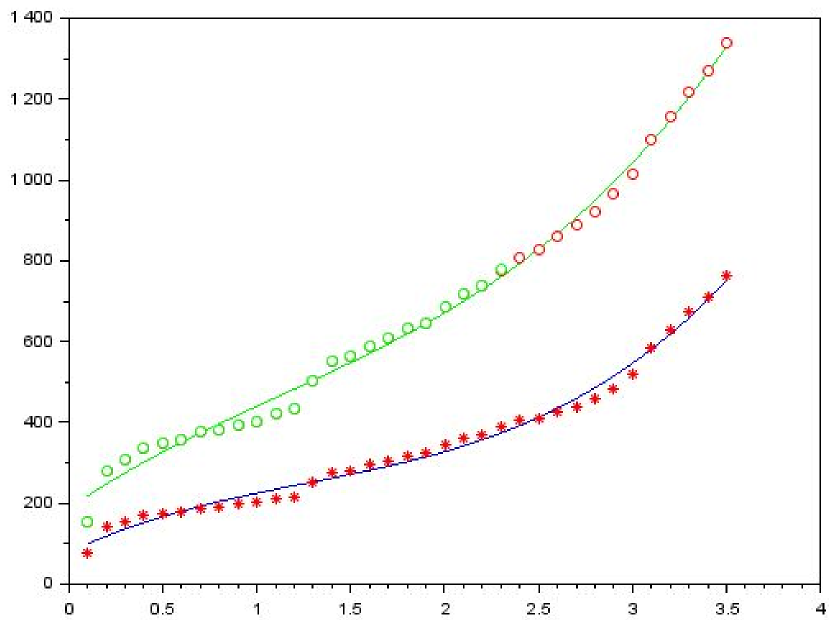
Nova Edição: Y = 26.708175*x^3 -102.037686*x^2 +421.710204*x +78.645760

Expectativa de público para a véspera: 1450

- Se utilizarmos o dobro do número de inscrições da edição passada:



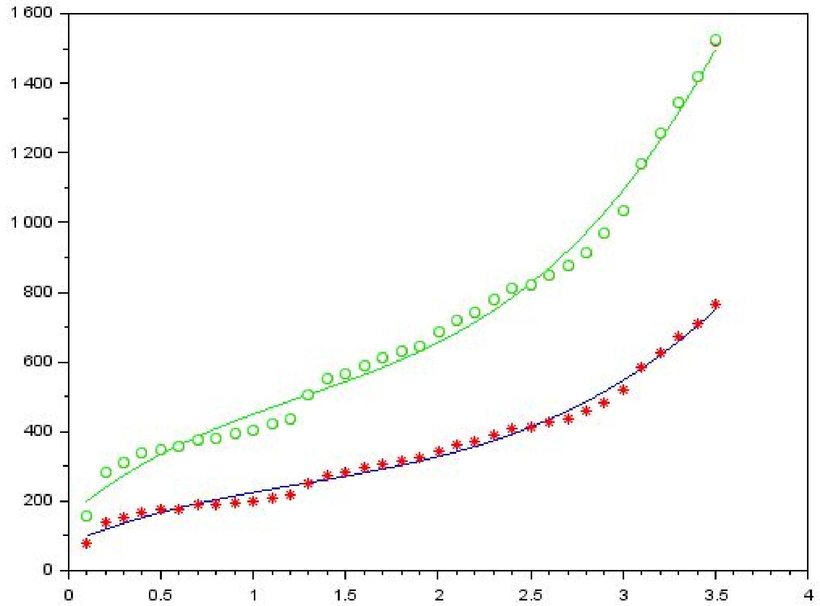
Edição Passada: Y = 26.708175*x^3 -102.037686*x^2 +221.710204*x +78.645760
Nova Edição: Y = 34.120786*x^3 -148.780374*x^2 +361.119311*x +182.718353
Expectativa de público para a véspera: 1087



Edição Passada: Y = 26.708175*x^3 -102.037686*x^2 +221.710204*x +78.645760

Nova Edição: Y = 26.806707*x^3 -90.859381*x^2 +316.013654*x +188.235392

Expectativa de público para a véspera: 1331



Edição Passada: Y = 26.708175*x^3 -102.037686*x^2 +221.710204*x +78.645760

Nova Edição: Y = 52.953824*x^3 -201.954362*x^2 +440.850089*x +157.960411

Expectativa de público para a véspera: 1497

Discussão e Conclusões

Tendo em vista a pesquisa desenvolvida, podemos perceber que para alguns casos, a estimativa foi bem sucedida, chegando ao término com o número de inscritos bem próximo ou igual ao estimado, mas para outros, o resultado apresentou uma diferença considerável.

O fato de usarmos uma equação de terceiro grau pode contribuir para essa divergência, pois não é o modelo ideal para este caso de estudo, além do fato de que criamos valores arbitrários para executarmos os testes. Pode ser que na próxima edição, consigamos chegar estimar a quantidade de inscritos e cheguemos bem próximos da quantidade real.

A maior dificuldade foi estabelecer uma taxa de correção, pois por mais que o termo independente da equação sempre seja a quantidade de inscritos no primeiro dia, a equação ainda apresenta uma diferença considerável no decorrer dela.

A vantagem é que as funções são iterativas, então podemos acrescentar a quantidade de inscritos por dia no decorrer do tempo em que as inscrições estiverem abertas e atualizarmos a estimativa todos os dias. A desvantagem é que as funções possuem uma limitação de 35 dias, pois é a quantidade de dados que temos para a base das inscrições antigas.

Mesmo apresentando algumas dificuldades no momento de modelar a equação de expectativa de público, utilizando as informações do modelo de comportamento das inscrições, achamos o resultado satisfatório. Esperamos poder confirmar o resultado desenvolvido na próxima edição do evento exemplo.

Referências

- https://www.ufrgs.br/numerico/livro/main.html#rias-graficos.html
- http://www.mat.ufmg.br/gaal/aplicacoes/quadrados_minimos.pdf
- https://pt.wikipedia.org/wiki/M%C3%A9todo_dos_m%C3%ADnimos_quadrados
- https://www.youtube.com/watch?v=-aHTbDw0twA&t=385s
- https://www.youtube.com/watch?v=Tm9wH5dUBIY&t=349s

Todos os dados utilizados foram obtidos com a Comissão de Organização da III Semana da Computação, sem comprometer a privacidade dos dados informados pelos participantes.

Apêndice A - Mínimos Quadrados para Equação de 2º Grau

```
function []=minQuad_2()

   // h é uma matriz que contém os vetores h1 (coluna de 1), h2 (coluna de valores do eixo x) e h3 (coluna de x^2)
   A = [1,0.1,0.01;1,0.2,0.04;1,0.3,0.09;1,0.4,0.16;1,0.5,0.25;1,0.6,0.36;1,0.7,0.49;1,0.8,0.64;1,0.9,0.81;1,1,1;1,1.1,1.21;1,1.2,1.44;1,1.3,1.69;1,1.4,1.96;1,1.5,2.25;1,1.6,2.56;1,1.7,2.89;1,1.8,3.24;1,1.9,3.61;1,2,4;1,2.1,4.41;1,2.2,4.84;1,2.3,5.29;1,2.4,5.76;1,2.5,6.25;1,2.6,6.76;1,2.7,7.29;1,2.8,7.84;1,2.9,8.41;1,3,9;1,3.1,9.61;1,3.2,10.24;1,3.3,10.89;1,3.4,11.56;1,3.5,12.25]

   // o vetor b é o vetor com a quantidade de inscrições até o dia do início do evento.
   b = [78;141;155;169;174;178;188;191;197;201;211;217;252;275;282;295;305;316;323;343;360;370;389;406;411;425;438;457;484;518;585;628;672;710;763]

   matriz = A' * A        // Calculamos a multiplicação entre transposta de A pela matriz A
   B = A' * b             // Calculamos a multiplicação entre a transposta de A o vetor de inscritos b
   w = inv(matriz) * B    // w é o vetor das constantes da equação obtida no método

   // Para ver os valores de cada matriz, basta descomentar as linhas abaixo
   // disp(matriz)
   // disp(A)
   // disp(A')
   // disp(b)
   // disp(w)

   // Agora basta imprimir a equação que modela os seus dados, com os valores dos coeficientes armazenados em w.
   printf('\nEdição  passada: Y = %f*x^2', w(3))
   if(w(2)> 0 ) then printf(' +%f*x', w(2)) else printf(' %f*x',w(2)) end
   if(w(1)> 0 ) then printf(' +%f', w(1)) else printf(' %f',w(1)) end

   x = (0.1:0.1:3.5)                  // Esbelece o intervalo de x
   y = w(3).*x.^2 + w(2).*x + w(1)    // Estabelece o formato da função com as constantes

   plot(x,y,'b')          // Desenha a linha da função
```

```
    plot(A(:,2),b,'*r')     // Desenha os pontos dos dados
endfunction
```

Apêndice B - Mínimos Quadrados para Equação de 3º Grau

```
function []=minQuad_3()

    // h é uma matriz que contém os vetores h1 (coluna de 1), h2 (coluna de valores do eixo x), h3 (coluna de x^2)e h4 (coluna de x^3)
    A =
[1,0.1,0.01,0.001;1,0.2,0.04,0.008;1,0.3,0.09,0.027;1,0.4,0.16,0.064;1,0.5,0.25,0.125;1,0.6,0.36,0.216;1,0.7,0.49,0.343;1,0.8,0.64,0.512;1,0.9,0.81,0.729;1,1,1,1;1,1.1,1.21,1.331;1,1.2,1.44,1.728;1,1.3,1.69,2.197;1,1.4,1.96,2.744;1,1.5,2.25,3.375;1,1.6,2.56,4.096;1,1.7,2.89,4.913;1,1.8,3.24,5.832;1,1.9,3.61,6.859;1,2,4,8;1,2.1,4.41,9.261;1,2.2,4.84,10.648;1,2.3,5.29,12.167;1,2.4,5.76,13.824;1,2.5,6.25,15.625;1,2.6,6.76,17.576;1,2.7,7.29,19.683;1,2.8,7.84,21.952;1,2.9,8.41,24.389;1,3,9,27;1,3.1,9.61,29.791;1,3.2,10.24,32.768;1,3.3,10.89,35.937;1,3.4,11.56,39.304;1,3.5,12.25,42.875]

    // o vetor b é o vetor com a quantidade de inscrições até o dia do início do evento.
    b =
[78;141;155;169;174;178;188;191;197;201;211;217;252;275;282;295;305;316;323;343;360;370;389;406;411;425;438;457;484;518;585;628;672;710;763]

    matriz = A' * A       // Calculamos a multiplicação entre transposta de A pela matriz A
    B = A' * b            // Calculamos a multiplicação entre a transposta de A o vetor de inscritos b.

    w = inv(matriz) * B    // w é o vetor das constantes da equação obtida no método

    // Para ver os valores de cada matriz, basta descomentar as linhas abaixo
    // disp(inv(matriz))
    // disp(A)
    // disp(B)
    // disp(A')
    // disp(b)
    // disp(w)

    // Agora basta imprimir a equação que modela os seus dados, com os valores dos coeficientes armazenados em w.
    printf('\nEdição Passada: Y = %f*x^3', w(4))
    if(w(3)> 0 ) then printf(' +%f*x^2', w(3)) else printf(' %f*x^2',w(3)) end
    if(w(2)> 0 ) then printf(' +%f*x', w(2)) else printf(' %f*x',w(2)) end
    if(w(1)> 0 ) then printf(' +%f', w(1)) else printf(' %f',w(1)) end

    x = (0.1:0.1:3.5)      // Esbelece o intervalo de x
```

```
    y = w(4).*x.^3 + w(3).*x.^2 + w(2).*x + w(1)    // Estabelece o formato da função com as constantes

    plot(x,y)            // Desenha a linha da função
    plot(A(:,2),b,'*r')    // Desenha os pontos dos dados

endfunction
```

Apêndice C - Função de Estimativa - 3º Grau

```
function []=minQuad_estimativa(novas_inscricoes)

   // h é uma matriz que contém os vetores h1 (coluna de 1), h2 (coluna de valores do eixo x), h3 (coluna de x^2)e h4 (coluna de x^3)
   A =
[1,0.1,0.01,0.001;1,0.2,0.04,0.008;1,0.3,0.09,0.027;1,0.4,0.16,0.064;1,0.5,0.25,0.125;1,0.6,0.36,0.216;1,0.7,0.49,0.343;1,0.8,0.64,0.512;1,0.9,0.81,0.729;1,1,1,1;1,1.1,1.21,1.331;1,1.2,1.44,1.728;1,1.3,1.69,2.197;1,1.4,1.96,2.744;1,1.5,2.25,3.375;1,1.6,2.56,4.096;1,1.7,2.89,4.913;1,1.8,3.24,5.832;1,1.9,3.61,6.859;1,2,4,8;1,2.1,4.41,9.261;1,2.2,4.84,10.648;1,2.3,5.29,12.167;1,2.4,5.76,13.824;1,2.5,6.25,15.625;1,2.6,6.76,17.576;1,2.7,7.29,19.683;1,2.8,7.84,21.952;1,2.9,8.41,24.389;1,3,9,27;1,3.1,9.61,29.791;1,3.2,10.24,32.768;1,3.3,10.89,35.937;1,3.4,11.56,39.304;1,3.5,12.25,42.875]

   inscricoes_antigas =
[78;63;14;14;5;4;10;3;6;4;10;6;35;23;7;13;10;11;7;20;17;10;19;17;5;14;13;19;27;34;67;43;44;38;53]   // Inscritos antigas por dia

   tam = size(novas_inscricoes,1)    // Quantidade de novas inscrições de entrada

   taxa(1) = (novas_inscricoes(1) - inscricoes_antigas(1))
   for i=2:size(novas_inscricoes,1)
      taxa(i) = (taxa(i-1) + (novas_inscricoes(i) - inscricoes_antigas(i)))/2
   end          // Calcula a media entre as variações de cada dia


   novo_b(1) = novas_inscricoes(1)              // Cria o vetor com novos pontos de novas_inscricoes
   for i=2:size(novas_inscricoes,1)
      novo_b(i) = novo_b(i-1)+ novas_inscricoes(i)
   end

   b(1) = novas_inscricoes(1)                    // Cria o vetor com as inscrições antigas
   for i=2:size(novas_inscricoes,1)
      b(i) = b(i-1)+ novas_inscricoes(i)
   end
   for i=size(novas_inscricoes,1):size(inscricoes_antigas,1) // Sobrescreve os dados novos que existirem
      b(i) = b(i-1)+ inscricoes_antigas(i)+ taxa(tam)      // Para os dias que não tem dos dados novos, somamos a taxa no do ano anterior
   end
```

```
    matriz = A' * A        // Calculamos a multiplicação entre transposta de A pela matriz A

    B = A' * b             // Calculamos a multiplicação entre a transposta de A o vetor de
inscritos b.

    w = inv(matriz) * B    // w é o vetor das constantes da equação obtida no método

    // Para ver os valores de cada matriz, basta descomentar as linhas abaixo
    // disp(matriz)
    // disp(A)
    // disp(B)
    // disp(A')
    // disp(b)
    // disp(w)

    // Agora basta imprimir a equação que modela os seus dados, com os valores dos
coeficientes armazenados em w.
    printf('\nNova Edição: Y = %f*x^3', w(4))
    if(w(3)> 0 ) then printf(' +%f*x^2', w(3)) else printf(' %f*x^2',w(3)) end
    if(w(2)> 0 ) then printf(' +%f*x', w(2)) else printf(' %f*x',w(2)) end
    if(w(1)> 0 ) then printf(' +%f', w(1)) else printf(' %f',w(1)) end

    x = (0.1:0.1:3.5)      // Estabelece o intervalo de x
    k = (0.1:0.1:tam/10)
    y = w(4).*x.^3 + w(3).*x.^2 + w(2).*x + w(1)    // Estabelece o formato da função com
as constantes

    plot(x,y,'g')          // Desenha a linha da função
    plot(x,b,'or')
    plot(k,novo_b,'og')    // Desenha os pontos dos novos dados

endfunction
```

Apêndice D - Função Compara

```
function []=Compara(novas_inscricoes)
   // minQuad_2()                          // Roda as duas funções ao mesmo tempo
   minQuad_3()
   minQuad_estimativa(novas_inscricoes)
endfunction
```