Relatório trabalho prático 2

César A. Galvão 19/0011572 Gabriela Carneiro 18/0120816

18 de July de 2022

Contents

1 Introdução **3**

2 Método **4**

2.1 Variável aleatória uniforme 4

2.2 Variável aleatória Poisson 4

2.3 Variável aleatória exponencial 4

2.4 Variável aleatória Normal 5

3 Resultados **6**

3.1 Uniforme 6

3.2 Poisson 7

3.3 Exponencial 8

3.4 Normal 9

4 Tabelas Normal estimadas **11**

4.1 Tabela Integração Monte Carlo 11

4.2 Tabela estimada pelo método polar 12

4.3 Tabela estimada pelo método da rejeição 13

5 Erros de estimação **14**

Anexo A - código comentado **15**

Resumo

Nesta atividade foram implementadas em R métodos para geração de variáveis aleatórias. Primeiramente foi criado um algoritmo para geração de 200.000 números pseudo-aleatórios uniformemente distribuídos entre (0, 1) pelo método congruencial linear. Este algoritmo foi então utilizado para gerar variáveis aleatórias para diferentes distribuições de probabilidade. A partir dos números pseudo-aleatórios, foram geradas 200.000 variáveis aleatórias de Poisson e 200.000 variáveis aleatórias em distribuição exponencial, pelo método da transformação inversa. Para gerar as 200.000 variáveis aleatórias em distribuição Normal foram utilizados dois métodos: o método da rejeição e método polar. Todas as variáveis geradas pelos métodos foram analisadas por meio de histogramas e teste de aderência. Todos os testes confirmaram que as variáveis geradas seguem as suas distribuições de referência. Por fim, foram geradas tabelas de probabilidade para a distribuição normal pelos métodos de Integração Monte Carlo, polar e da rejeição. Por meio das simulações feitas, pode-se concluir que os métodos de geração das variáveis aleatórias são eficientes e que dos métodos de geração utilizados para a construção das tabelas de probabilidade, o polar é o menos eficiente.[1]

[1] Todos os documentos desse relatório podem ser verificados no repositório https://github.com/cesar-galvao/Estatistica-computacional

1 Introdução

Um número aleatório é um número gerado por meio de um processo cujo resultado é probabilístico e, consequentemente, não pode ser reproduzido de forma determinística. Um número aleatório pode ser gerado por meio de experimentação, como em lançamentos de um dado, contagens de ocorrências de raios gama, usando dígitos de pi, entre outros. No entanto, esses métodos requerem grande esforço e tempo para serem utilizados.

Dessa forma, como alternativa, existem algoritmos que implementam métodos para gerar números pseudo aleatórios. Um gerador de números pseudo aleatórios é um algoritmo que cria sequências de números cujas propriedades se assemelham às propriedades de números aleatórios, mas que não podem ser classificados como aleatórios, pois as sequencias são determinadas por um valor inicial, uma semente.

Dentre os métodos de geração de números pseudo aleatórios existe o gerador linear congruencial. O gerador linear terá como output uma distribuição uniforme. Depois disso, teoremas de estatística e probabilidade serão utilizados para gerar outras variáveis aleatórias correspondentes a outras distribuições, com base no gerador aleatório.

O gerador é definido pela relação de recorrência

$$X_{n+1} = (aX_n + c) \mod m \tag{1}$$

em que a e m são inteiros positivos e $\mod$ é o operador para obtenção do resto da divisão.

A variável X retorna a sequência de números pseudo-aleatórios. Se $c = 0$, o gerador é chamado de gerador congruencial multiplicativo. Se $c \neq 0$, o gerador congruencial é chamado de gerador congruencial misto.

Para gerar números aleatórios, um gerador linear congruencial será desenvolvido. Esse gerador irá inicialmente ser utilizado para geração de variáveis aleatórias uniformemente distribuídas no intervalo $(0, 1)$. A partir dessa função, serão geradas variáveis aleatórias nas distribuições de Poisson, exponencial, e normal padrão.

2 Método

2.1 Variável aleatória uniforme

A geração da variável aleatória uniforme segue o seguinte algoritmo:

1. As constantes a e m são definidas. Aqui, foram utilizados $a = 16.807$ e $m = 2^{31} - 1$;
2. Uma semente aleatória é obtida utilizando o relógio do sistema pela função `Sys.time()`;
3. A semente é redefinida $y_{i+1} = (a \cdot y_i) \mod m$;
4. O número pertencente a $U(0, 1)$ é gerado $x_i = y_{i+1}/m$.

2.2 Variável aleatória Poisson

O algoritmo gera uma variável aleatória que segue a distribuição Poisson com média λ. Primeiramente ele gera um número aleatório uniformemente distribuído U e verifica $U < e^{-\lambda} = p_0$. Se for menor, o algoritmo assume $X = 0$. Caso contrário, ele itera o laço e verifica novamente novamente a condição, mas agora para a proposição $U < p_0 + p_1$. Se o valor for menor, o algoritmo atribui $X = 1$.

O algoritmo implementado verifica sucessivamente se o valor da Poisson é zero, em seguida verifica se o valor é 1, então 2 e assim por diante. Dessa forma, o número de comparações necessários será uma unidade maior que o valor da Poisson (1 + λ). Para valores pequenos de λ o programa é eficiente, mas quando λ assume valores muito grandes esse algoritmo não é o mais otimizado. O ideal é escolher valores mais próximos de lambda para iniciar a verificação do laço. A implementação escolhida foi a primeira, não otimizada.

2.3 Variável aleatória exponencial

Para gerar variáveis na distribuição exponencial, o método implementado foi o da transformação inversa.

Supondo que X é uma variável aleatória com distribuição exponencial de $\lambda = 1$, sua função de distribuição pode ser expressa por

$$F(x) = 1 - e^{-x} \tag{2}$$

Se, $x = F^{-1}(u)$, então,

$$u = F(x) = 1 - e^{-x} \longrightarrow 1 - u = e^{-x} \tag{3}$$

Aplicando o logaritmo, tem-se $x = -log(1 - u)$. Dessa forma, podemos gerar X por meio de um número aleatório distribuído uniformemente e definindo $X = F^{-1}(U) = -\log(1 - U)$.

Uma pequena economia de tempo pode ser obtida observando que $1 - U$ também é uniforme em $(0, 1)$ e assim $-\log(1 - U)$ tem a mesma distribuição que $-\log U$. Ou seja, o logaritmo negativo de um número aleatório é distribuído exponencialmente com $\lambda = 1$.

Além disso, observa-se que se X é exponencial com média 1, então, para qualquer constante c, cX é exponencial com média c. Portanto, uma variável aleatória exponencial X com parâmetro λ (média $1/\lambda$) pode ser gerada por meio de um número aleatório U e definindo:

$$X = \frac{-1}{\lambda} \cdot \log U \tag{4}$$

2.4 Variável aleatória Normal

2.4.1 Método polar

O método polar segue a abordagem abaixo para gerar um par de normais padrão independentes:

1. Gerar números pseudo aleatórios uniformente distribuídos, U_1 e U_2;
2. Gerar $V_1 = 2U_1 - 1,\ V_2 = 2U_2 - 1,\ S = V_1^2 + V_2^2$;
3. Se $S > 1$, retornar ao passo 1;
4. Retornar as variáveis independentes com distribuição normal padrão.

$$X = \sqrt{\frac{-2\log S}{S}}V_1, \quad Y = \sqrt{\frac{-2\log S}{S}}V_2 \tag{5}$$

2.4.2 Método da rejeição

O método da rejeição gera valores de uma distribuição X com função de densidade $f(x)$ a partir de uma distribuição de partida Y com densidade de probabilidade $g(x)$. A ideia é que se pode gerar um valor de X por meio de Y aceitando ou não o valor de Y com probabilidade $\frac{f(y)}{c\,g(y)}$. As comparações são repetidas Y até que um valor seja aceito. Especificamente, seja c uma constante tal que $f(y)/g(y) \leq c$, para todo y. Dessa forma, a técnica a seguir gera uma variável aleatória com densidade f:

1. Gerar Y tendo densidade g;
2. Gerar um número pseudo-aleatório U;
3. Se $U \leq \frac{f(y)}{c\,g(y)}$, assumir $X = Y$. Caso contrário, retornar para passo 1.

Com base nisso, pode-se afirmar que a variável aleatória gerada pelo método de rejeição tem densidade f e que o número de iterações do algoritmo que são necessárias é um número que segue uma distribuição geométrica com média c.

O histograma mostra que as variáveis aleatórias geradas pelo método aparentemente têm uma tendencia normal, apesar de apresentar uma queda em torno do zero. Assim como as distribuições anteriores, o fato de o gerador do número pseudo-aleatório uniforme não ter se comportado como o esperado nas extremidades pode ter afetado a geração das variáveis aleatórias normalmente distribuídas.

3 Resultados

A seguir são expostas duas comparações para avaliação da qualidade de cada um dos algoritmos utilizados: a primeira é gráfica e expõe histogramas das distribuições geradas e a linha correspondente à distribuição teórica. A segunda é a realização de testes de aderência qui-quadrado, os quais foram realizados comparando os quartis das distribuições empíricas com as distribuições hipotéticas.

Para os testes de aderência, as hipóteses utilizadas são as mesmas em todos os casos:

$$\begin{cases} H_0 : \text{não existe diferença significativa entre os valores observados e os esperados;} \\ H_a : \text{existe diferença significativa entre os valores observados e os esperados;} \end{cases} \tag{6}$$

Além disso, todas as amostras têm tamanho $n = 200.000$.

3.1 Uniforme

A amostra gerada para a distribuição uniforme se aproxima graficamente de sua função densidade teórica. Exceções podem ser observadas nos valores extremos da amostra e os motivos para isso, pelo menos graficamente, podem ser dois:

1. De fato o algoritmo utilizado gera precariamente valores nas extremidades da distribuição; ou

2. O histograma gerado utiliza valores acima e abaixo dos centros das barras para projetar a altura das barras. Como valores abaixo de 0 e acima de 1 não existem, suas barras são mais curtas.

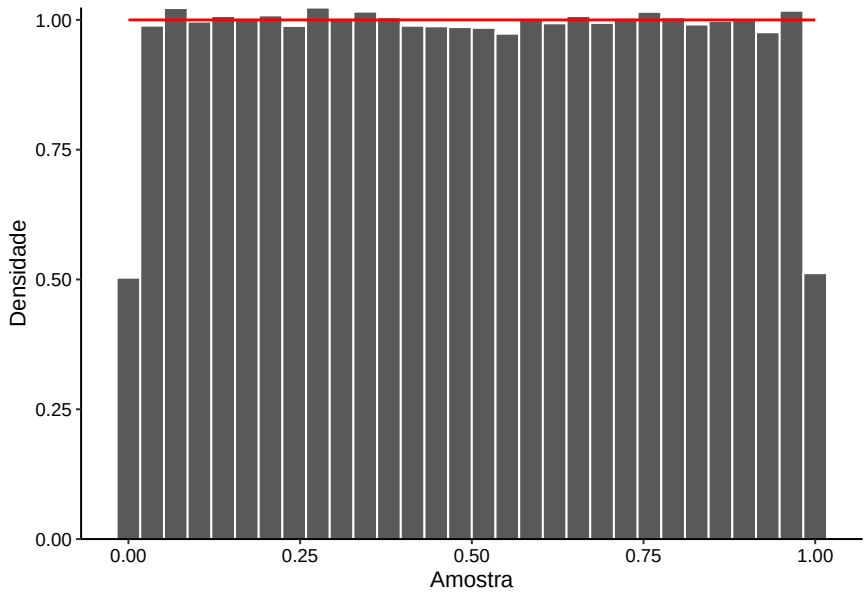


Figure 1: Histograma da amostra de v.a. uniforme com linha da f.d.p. uniforme

Ao realizar o teste de aderência, obtém-se p-valor igual a 1, corroborando a maior chance de o tópico 2 ser o motivo para as barras mais curtas nas extremidades. O teste indica portanto que, utilizando a distribuição dos quartis da amostra, não seria possível diferenciá-la de uma variável aleatória com distribuição uniforme teórica.

3.2 Poisson

O gráfico gerado para a v.a. Poisson suscita as mesmas dúvidas quanto à altura das barras. Nota-se que as barras estão abaixo dos pontos teóricos enquanto estão também deslocados em relação aos centros das barras.

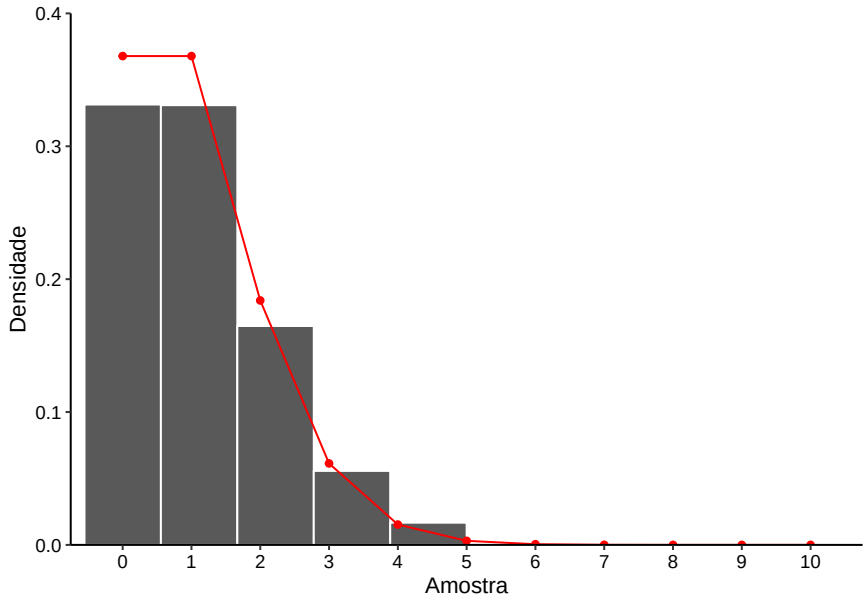


Figure 2: Histograma da amostra de v.a. Poisson com linha da f.p. Poisson

O teste de aderência para a distribuição indica p-valor muito próximo de 1, o que sugere novamente uma dificuldade de composição do gráfico e a impossibilidade de diferenciar a distribuição da amostra da distribuição teórica.

3.3 Exponencial

A amostra gerada para a distribuição exponencial também parece estar próxima da distribuição teórica, a menos da barra inicial correspondente a valores próximos a zero.

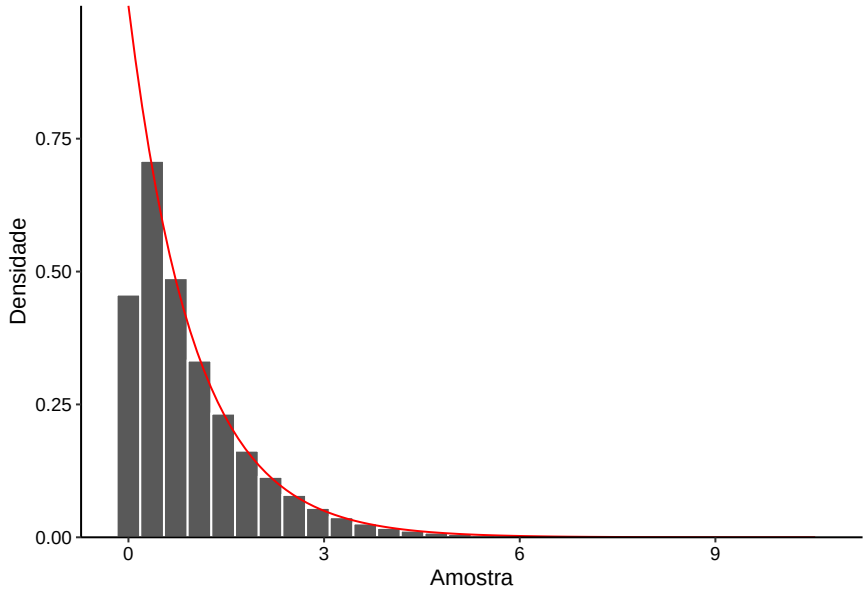


Figure 3: Histograma da amostra de v.a. exponencial com linha da f.d.p. exponencial

Novamente o teste de aderência é realizado com p-valor 1, sugerindo a impossibilidade de diferenciação da distribuição da amostra em relação à distribuição teórica.

3.4 Normal

O histograma mostra que as variáveis aleatórias geradas pelo método aparentemente têm uma tendencia normal, apesar de apresentar uma queda em torno do zero. Assim como as distribuições anteriores, o fato de o gerador do número pseudo-aleatório uniforme não ter se comportado como o esperado nas extremidades pode ter afetado a geração das variáveis aleatórias normalmente distribuídas.

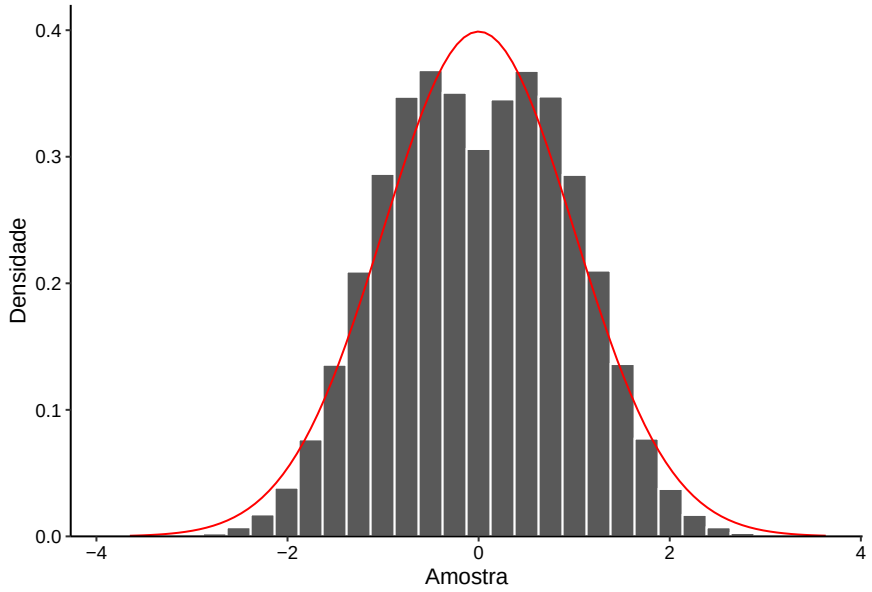


Figure 4: Histograma da amostra de v.a. normal, método rejeição, com linha da f.d.p. normal

Conforme histograma a seguir, tanto X quanto Y gerados pelo método polar seguem distribuição normal com médias centradas em zero.

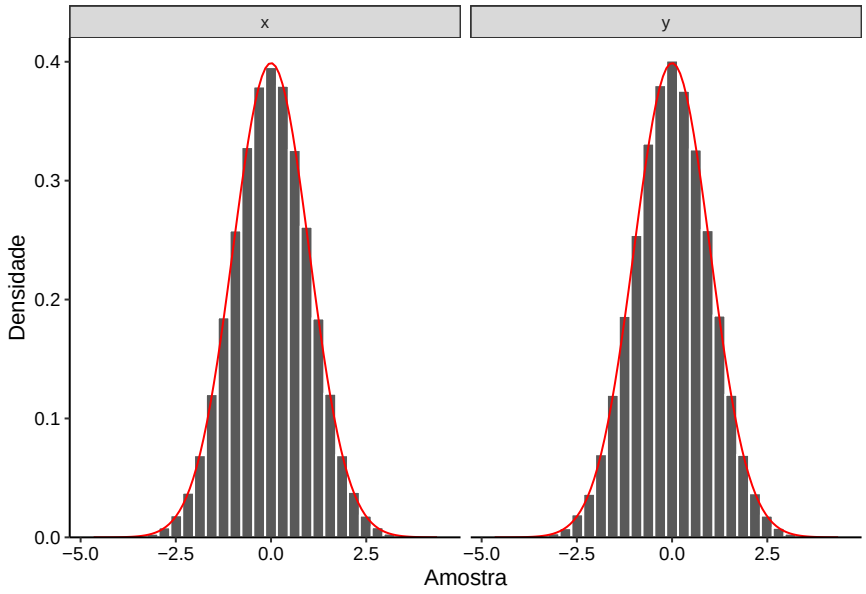


Figure 5: Histograma da amostra de v.a. normal, método polar, com linha da f.d.p. normal

4 Tabelas Normal estimadas

As estimações a seguir consideram a probabilidade acumulada de $(-\infty, b]$. É esperado que os valores variem de 0.5 a 1, no entanto as aproximações realizadas pelos métodos de Integração de Monte Carlo e os demais utilizados para geração de amostras normais estão suscetíveis a erros aleatórios. Observa-se nas tabelas a seguir esses erros de estimação quando se atinge, antes do valor máximo dos quantis disponíveis, $p = 1$ ou $p < 1$.

4.1 Tabela Integração Monte Carlo

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6737	0.6773	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7020	0.7054	0.7089	0.7123	0.7157	0.7191	0.7224
0.6	0.7258	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7518	0.7549
0.7	0.7581	0.7612	0.7643	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7853
0.8	0.7882	0.7911	0.7939	0.7968	0.7996	0.8024	0.8052	0.8079	0.8106	0.8133
0.9	0.8160	0.8186	0.8213	0.8239	0.8265	0.8290	0.8315	0.8340	0.8365	0.8390
1	0.8414	0.8438	0.8462	0.8486	0.8509	0.8532	0.8555	0.8578	0.8600	0.8622
1.1	0.8644	0.8666	0.8687	0.8708	0.8729	0.8750	0.8771	0.8791	0.8811	0.8831
1.2	0.8850	0.8869	0.8889	0.8907	0.8926	0.8944	0.8963	0.8980	0.8998	0.9016
1.3	0.9033	0.9050	0.9067	0.9083	0.9100	0.9116	0.9132	0.9147	0.9163	0.9178
1.4	0.9193	0.9208	0.9223	0.9237	0.9251	0.9266	0.9279	0.9293	0.9306	0.9320
1.5	0.9333	0.9345	0.9358	0.9371	0.9383	0.9395	0.9407	0.9419	0.9430	0.9441
1.6	0.9453	0.9463	0.9474	0.9485	0.9495	0.9506	0.9516	0.9526	0.9535	0.9545
1.7	0.9555	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9624	0.9633
1.8	0.9640	0.9648	0.9656	0.9663	0.9671	0.9678	0.9685	0.9692	0.9699	0.9706
1.9	0.9712	0.9719	0.9725	0.9731	0.9737	0.9743	0.9749	0.9755	0.9760	0.9766
2	0.9771	0.9776	0.9781	0.9787	0.9792	0.9796	0.9801	0.9806	0.9810	0.9815
2.1	0.9819	0.9823	0.9828	0.9832	0.9836	0.9840	0.9843	0.9847	0.9851	0.9854
2.2	0.9858	0.9861	0.9865	0.9868	0.9871	0.9874	0.9877	0.9880	0.9883	0.9886
2.3	0.9889	0.9891	0.9894	0.9896	0.9899	0.9901	0.9904	0.9906	0.9908	0.9911
2.4	0.9913	0.9915	0.9917	0.9919	0.9921	0.9923	0.9924	0.9926	0.9928	0.9930
2.5	0.9931	0.9933	0.9934	0.9936	0.9937	0.9939	0.9940	0.9942	0.9943	0.9944
2.6	0.9945	0.9947	0.9948	0.9949	0.9950	0.9951	0.9952	0.9953	0.9954	0.9955
2.7	0.9956	0.9957	0.9958	0.9959	0.9959	0.9960	0.9961	0.9962	0.9962	0.9963
2.8	0.9964	0.9964	0.9965	0.9965	0.9966	0.9966	0.9967	0.9967	0.9968	0.9968
2.9	0.9969	0.9969	0.9970	0.9970	0.9970	0.9971	0.9971	0.9971	0.9972	0.9972
3	0.9972	0.9973	0.9973	0.9973	0.9973	0.9974	0.9974	0.9974	0.9974	0.9974
3.1	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975	0.9976
3.2	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976
3.3	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976
3.4	0.9976	0.9976	0.9976	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975	0.9975
3.5	0.9975	0.9975	0.9975	0.9975	0.9975	0.9974	0.9974	0.9974	0.9974	0.9974
3.6	0.9974	0.9974	0.9974	0.9974	0.9973	0.9973	0.9973	0.9973	0.9973	0.9973
3.7	0.9973	0.9973	0.9972	0.9972	0.9972	0.9972	0.9972	0.9972	0.9972	0.9971
3.8	0.9971	0.9971	0.9971	0.9971	0.9971	0.9971	0.9970	0.9970	0.9970	0.9970
3.9	0.9970	0.9970	0.9970	0.9969	0.9969	0.9969	0.9969	0.9969	0.9969	0.9968

4.2 Tabela estimada pelo método polar

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0	0.5000	0.5039	0.5075	0.5115	0.5156	0.5195	0.5235	0.5277	0.5319	0.5359
0.1	0.5399	0.5438	0.5478	0.5517	0.5555	0.5592	0.5631	0.5669	0.5708	0.5749
0.2	0.5789	0.5830	0.5867	0.5906	0.5944	0.5983	0.6023	0.6061	0.6102	0.6140
0.3	0.6178	0.6217	0.6255	0.6292	0.6330	0.6367	0.6405	0.6443	0.6477	0.6513
0.4	0.6549	0.6585	0.6622	0.6661	0.6699	0.6736	0.6773	0.6808	0.6842	0.6879
0.5	0.6914	0.6949	0.6984	0.7016	0.7051	0.7084	0.7119	0.7154	0.7188	0.7221
0.6	0.7254	0.7288	0.7320	0.7353	0.7387	0.7418	0.7449	0.7480	0.7511	0.7543
0.7	0.7574	0.7605	0.7634	0.7664	0.7694	0.7723	0.7754	0.7784	0.7815	0.7844
0.8	0.7875	0.7903	0.7931	0.7959	0.7988	0.8015	0.8044	0.8072	0.8100	0.8129
0.9	0.8155	0.8183	0.8209	0.8234	0.8262	0.8288	0.8314	0.8338	0.8361	0.8386
1	0.8409	0.8434	0.8458	0.8481	0.8504	0.8528	0.8553	0.8577	0.8601	0.8623
1.1	0.8645	0.8668	0.8689	0.8710	0.8732	0.8753	0.8773	0.8794	0.8814	0.8834
1.2	0.8854	0.8874	0.8894	0.8913	0.8933	0.8950	0.8969	0.8987	0.9004	0.9020
1.3	0.9038	0.9055	0.9071	0.9088	0.9105	0.9119	0.9134	0.9148	0.9163	0.9179
1.4	0.9193	0.9207	0.9222	0.9238	0.9252	0.9265	0.9278	0.9294	0.9307	0.9320
1.5	0.9332	0.9345	0.9358	0.9372	0.9384	0.9396	0.9408	0.9419	0.9430	0.9440
1.6	0.9452	0.9464	0.9476	0.9489	0.9500	0.9511	0.9521	0.9531	0.9541	0.9549
1.7	0.9558	0.9567	0.9576	0.9586	0.9596	0.9605	0.9613	0.9621	0.9630	0.9637
1.8	0.9645	0.9653	0.9659	0.9666	0.9673	0.9680	0.9687	0.9693	0.9700	0.9707
1.9	0.9713	0.9720	0.9726	0.9732	0.9738	0.9743	0.9749	0.9755	0.9761	0.9766
2	0.9771	0.9776	0.9781	0.9787	0.9792	0.9797	0.9803	0.9808	0.9814	0.9817
2.1	0.9821	0.9825	0.9830	0.9834	0.9838	0.9842	0.9845	0.9849	0.9853	0.9857
2.2	0.9860	0.9862	0.9866	0.9869	0.9872	0.9875	0.9878	0.9881	0.9885	0.9889
2.3	0.9892	0.9894	0.9897	0.9901	0.9903	0.9906	0.9909	0.9911	0.9914	0.9917
2.4	0.9919	0.9921	0.9924	0.9925	0.9927	0.9929	0.9930	0.9932	0.9935	0.9936
2.5	0.9938	0.9939	0.9940	0.9942	0.9944	0.9945	0.9947	0.9949	0.9951	0.9952
2.6	0.9953	0.9954	0.9955	0.9956	0.9958	0.9959	0.9960	0.9961	0.9962	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9973
2.8	0.9974	0.9975	0.9976	0.9976	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9982	0.9982	0.9983	0.9984	0.9984	0.9984	0.9985	0.9986	0.9986	0.9987
3	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990	0.9991
3.1	0.9991	0.9991	0.9991	0.9991	0.9992	0.9992	0.9993	0.9993	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998	0.9998	0.9998	0.9998
3.5	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9999	0.9999	0.9999
3.6	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
3.7	0.9999	0.9999	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

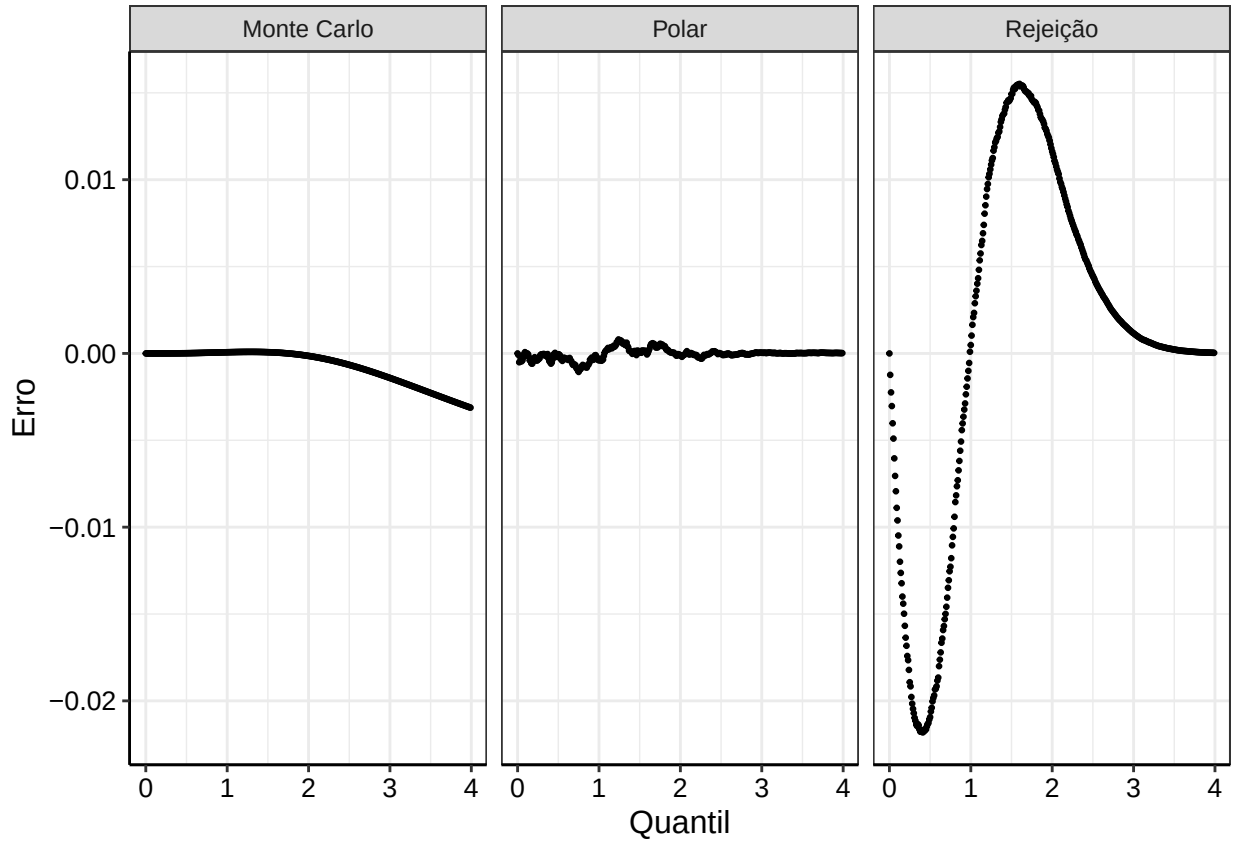
4.3 Tabela estimada pelo método da rejeição

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0	0.5000	0.5028	0.5057	0.5089	0.5119	0.5150	0.5179	0.5209	0.5239	0.5270
0.1	0.5302	0.5333	0.5367	0.5397	0.5430	0.5464	0.5496	0.5531	0.5564	0.5597
0.2	0.5629	0.5663	0.5696	0.5733	0.5766	0.5798	0.5834	0.5866	0.5901	0.5936
0.3	0.5972	0.6007	0.6044	0.6079	0.6116	0.6155	0.6192	0.6227	0.6262	0.6300
0.4	0.6337	0.6373	0.6410	0.6447	0.6484	0.6520	0.6558	0.6595	0.6631	0.6669
0.5	0.6705	0.6744	0.6781	0.6819	0.6856	0.6892	0.6929	0.6964	0.6999	0.7036
0.6	0.7071	0.7111	0.7147	0.7184	0.7223	0.7257	0.7294	0.7329	0.7364	0.7399
0.7	0.7435	0.7471	0.7507	0.7542	0.7578	0.7611	0.7646	0.7683	0.7717	0.7752
0.8	0.7787	0.7825	0.7857	0.7891	0.7922	0.7956	0.7989	0.8022	0.8055	0.8088
0.9	0.8119	0.8149	0.8180	0.8209	0.8240	0.8270	0.8300	0.8330	0.8359	0.8390
1	0.8418	0.8448	0.8478	0.8506	0.8532	0.8560	0.8587	0.8613	0.8639	0.8665
1.1	0.8691	0.8719	0.8744	0.8770	0.8793	0.8818	0.8844	0.8870	0.8895	0.8920
1.2	0.8944	0.8966	0.8989	0.9010	0.9031	0.9052	0.9073	0.9092	0.9114	0.9133
1.3	0.9153	0.9171	0.9190	0.9207	0.9226	0.9243	0.9261	0.9280	0.9297	0.9314
1.4	0.9330	0.9345	0.9362	0.9378	0.9395	0.9409	0.9424	0.9437	0.9452	0.9466
1.5	0.9481	0.9494	0.9509	0.9523	0.9534	0.9547	0.9560	0.9572	0.9584	0.9596
1.6	0.9607	0.9617	0.9628	0.9639	0.9649	0.9658	0.9667	0.9676	0.9686	0.9695
1.7	0.9704	0.9714	0.9721	0.9730	0.9739	0.9746	0.9754	0.9761	0.9769	0.9777
1.8	0.9785	0.9791	0.9798	0.9804	0.9811	0.9816	0.9822	0.9828	0.9834	0.9840
1.9	0.9845	0.9849	0.9855	0.9860	0.9864	0.9869	0.9874	0.9878	0.9882	0.9885
2	0.9889	0.9893	0.9896	0.9899	0.9903	0.9906	0.9909	0.9912	0.9916	0.9918
2.1	0.9920	0.9923	0.9926	0.9929	0.9931	0.9933	0.9935	0.9937	0.9939	0.9942
2.2	0.9943	0.9945	0.9947	0.9949	0.9951	0.9953	0.9954	0.9956	0.9958	0.9960
2.3	0.9961	0.9963	0.9964	0.9966	0.9967	0.9969	0.9970	0.9971	0.9972	0.9973
2.4	0.9973	0.9974	0.9976	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981	0.9982
2.5	0.9983	0.9983	0.9984	0.9984	0.9985	0.9986	0.9986	0.9987	0.9987	0.9988
2.6	0.9988	0.9989	0.9990	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992
2.7	0.9993	0.9993	0.9993	0.9993	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
2.8	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997	0.9997	0.9997
2.9	0.9997	0.9997	0.9997	0.9997	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998
3	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998	0.9999
3.1	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
3.2	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	1.0000	1.0000
3.3	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.5	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.6	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.8	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3.9	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

5 Erros de estimação

A seguir são expostos gráficos com os erros de estimação da distribuição normal. Eles foram calculados da seguinte forma: $\varepsilon = \hat{z} - z$, onde z é o valor obtido para um determinado quantil utilizando `pnorm(quantil, 0, 1)`.

No método de Monte Carlo é possível observar que erros parecem ocorrer em maior magnitude, especificamente subestimados, em um quantil próximo de 2. Para o método Polar, quantis maiores parecem ter menos erro de estimação enquanto quantis abaixo de 3 parecem estar suscetíveis a um ruído aleatório. Por último, o método da rejeição é o que exibe erros de maior magnitude com comportamento aparentemente sinoidal – erros parecem diminuir no caso assintótico.



Anexo A - código comentado

```r
# VA UNIFORME
uniforme <- function(n){

  x <- c()

  a <- 16807
  m <- 2^31 -1

  # se tem o arquivo com seed, pega o seed k do arquivo
  # Se nao tiver arquivo, gera o arquivo e escreve o seed
  # seed começa com o relogio
  # Os que extrapolarem inserir no arquivo

  if (file.exists("../trabalho pratico 2/seeds.Rdata")){
    y <- readRDS("../trabalho pratico 2/seeds.Rdata")
  } else {
    y <- as.numeric(Sys.time())
  }

  for (i in 1:n){
    y <- (a*y)%%m
    x[i] <- y/m
  }

  #guardar o ultimo y num arquivo
  saveRDS(y, "../trabalho pratico 2/seeds.Rdata")

  return(x)
}

# VA POISSON
poisson <- function(lambda){

  if(!is.integer(lambda)){
    message("Usando a parte inteira do lambda fornecido.")
    lambda <- as.integer(lambda)
  }

  p <- exp(-lambda)
  U <- uniforme(1)
  f <- p

  i <- 0

  # testa se o número está na média
    while (U >= f){
      p <- (p*lambda)/(i+1)
      f <- f+p
      i <- i+1
    }
```

```
    return(i)
}

# VA EXPONENCIAL
geraexp <- function(n, lambda){
  u <- uniforme(n)

  saida <- -log(u)/lambda

  return(saida)
}

# VA NORMAL
geranormal <- function(metodo){
  if (metodo == "rejeicao"){
    y <- geraexp(n = 1, lambda = 1)
    u <- uniforme(1)
    while(u > exp(-(y-1)^2)/2){
      u <- uniforme(1)
      y <- geraexp(n = 1, lambda = 1)
    }
    modZ <- abs(y)

    if(uniforme(1) <= 0.5){
      Z <- modZ
    } else {
      Z <- -modZ
    }
    return(Z)
  }

  if(metodo == "polar"){
    v1 <- (2*uniforme(1)-1)
    v2 <- (2*uniforme(1)-1)
    u <- v1^2 + v2^2
    while (u > 1){
      v1 <- (2*uniforme(1)-1)
      v2 <- (2*uniforme(1)-1)
      u <- v1^2 + v2^2
    }
    x <- sqrt(-2*log(u)/u)*v1
    y <- sqrt(-2*log(u)/u)*v2
    return(list(x = x, y = y))
  }
}

# GERACAO DE AMOSTRAS
if(!file.exists("../trabalho pratico 2/amostra_uniforme.Rdata")){
  # uniforme
  am_uni <- data.frame(amostra = uniforme(200000))

  # poisson lambda = 1
  am_poisson <- c()
```

```
  for (i in 1:200000){
    am_poisson[i] <- poisson(1L)
  }
  am_poisson <- data.frame(amostra = am_poisson)

  # exponencial lambda 1
  am_exp <- data.frame(amostra = geraexp(200000, 1))

  # normal rejeição
  am_normrej <- c()
  for (i in 1:200000){
    am_normrej[i] <- geranormal("rejeicao")
  }
  am_normrej <- data.frame(amostra = am_normrej)

  # normal polar
  x <- c()
  y <- c()
  for (i in 1:200000){
    x[i] <- geranormal("polar")$x
    y[i] <- geranormal("polar")$y
  }

  am_normpolar <- data.frame(x = x, y = y)
  rm(x,y,i)

  saveRDS(am_uni, "../trabalho pratico 2/amostra_uniforme.Rdata")
  saveRDS(am_poisson, "../trabalho pratico 2/amostra_poisson.Rdata")
  saveRDS(am_exp, "../trabalho pratico 2/amostra_exponencial.Rdata")
  saveRDS(am_normrej, "../trabalho pratico 2/amostra_normal_rejeicao.Rdata")
  saveRDS(am_normpolar, "../trabalho pratico 2/amostra_normal_polar.Rdata")
} else {
  am_uni <- readRDS("../trabalho pratico 2/amostra_uniforme.Rdata")
  am_poisson <- readRDS("../trabalho pratico 2/amostra_poisson.Rdata")
  am_exp <- readRDS("../trabalho pratico 2/amostra_exponencial.Rdata")
  am_normrej <- readRDS("../trabalho pratico 2/amostra_normal_rejeicao.Rdata")
  am_normpolar <- readRDS("../trabalho pratico 2/amostra_normal_polar.Rdata")
}

# GRAFICO E TESTE UNIFORME
ggplot(am_uni, aes(amostra))+
  geom_histogram(aes(y = ..density..),color = 'white')+
  stat_function(fun = dunif, color = 'red', args = list(min = 0, max = 1), size = 0.7)+
  labs(x = "Amostra", y = "Densidade")+
  scale_y_continuous(expand = c(0,0))+
  theme_bw() +
  theme(axis.title.y=element_text(colour="black", size=12),
        axis.title.x = element_text(colour="black", size=12),
        axis.text = element_text(colour = "black", size=9.5),
        panel.border = element_blank(),
        axis.line = element_line(colour = "black"),
        panel.grid = element_blank())
```

```r
q1 <- sum(am_uni$amostra < .25)
q2 <- sum(am_uni$amostra  >= .25 & am_uni$amostra < .5)
q3 <- sum(am_uni$amostra  >= .5 & am_uni$amostra < .75)
q4 <- sum(am_uni$amostra  >= .75 & am_uni$amostra <= 1)

quartis <- c(q1,q2, q3, q4)/2000

pvalor <- chisq.test(quartis, p = c(.25, .25, .25, .25))$p.value

# GRAFICO E TESTE POISSON
x.values <- seq(0, 10, 1)
y2 <- dpois(x.values, 1)
df2 <- data.frame(x.values, y2)

ggplot(df2, aes(x=x.values, y=y2))+
  geom_histogram(data = am_poisson, aes(x = amostra, y = ..density..),color = 'white', bins = 10)+
  geom_point(stat="identity", width = 0.5, color = "red")+
  geom_line(color = "red")+
  labs(x = "Amostra", y = "Densidade")+
  scale_y_continuous(expand = c(0,0), limits = c(0, .4))+
  scale_x_continuous(breaks = c(0:10), expand = c(0, 0.2))+
  theme_bw() +
  theme(axis.title.y=element_text(colour="black", size=12),
        axis.title.x = element_text(colour="black", size=12),
        axis.text = element_text(colour = "black", size=9.5),
        panel.border = element_blank(),
        axis.line = element_line(colour = "black"),
        panel.grid = element_blank())

q1 <- sum(am_poisson$amostra <= qpois(.25, lambda = 1))
q2 <- sum(am_poisson$amostra  > qpois(.25, lambda = 1) & am_poisson$amostra <= qpois(.5, lambda = 1))
q3 <- sum(am_poisson$amostra  > qpois(.5, lambda = 1) & am_poisson$amostra <= qpois(.75, lambda = 1))
q4 <- sum(am_poisson$amostra  > qpois(.75, lambda = 1))

quartis <- c(q1,q2, q3, q4)/2000

p <- c(dpois(0, lambda = 1),dpois(1, lambda = 1), dpois(2, lambda = 1))
p[4] <- 1-sum(p)

pvalor <- chisq.test(quartis, p = p)$p.value

# GRAFICO E TESTE EXPONENCIAL

ggplot(am_exp, aes(amostra))+
  geom_histogram(aes(y = ..density..),color = 'white')+
  stat_function(fun = dexp, color = 'red', args = list(1))+
  labs(x = "Amostra", y = "Densidade")+
  scale_y_continuous(expand = c(0,0))+
  theme_bw() +
  theme(axis.title.y=element_text(colour="black", size=12),
        axis.title.x = element_text(colour="black", size=12),
        axis.text = element_text(colour = "black", size=9.5),
        panel.border = element_blank(),
```

```
        axis.line = element_line(colour = "black"),
        panel.grid = element_blank())

q1 <- sum(am_exp$amostra <= qexp(.25, rate = 1))
q2 <- sum(am_exp$amostra  > qexp(.25, rate = 1) & am_exp$amostra <= qexp(.5, rate = 1))
q3 <- sum(am_exp$amostra  > qexp(.5, rate = 1) & am_exp$amostra <= qexp(.75, rate = 1))
q4 <- sum(am_exp$amostra  > qexp(.75, rate = 1))

quartis <- c(q1,q2, q3, q4)/2000

pvalor <- chisq.test(quartis, p = c(.25, .25, .25, .25))$p.value

# GRAFICO E TESTE NORMAL

ggplot(am_normrej, aes(amostra))+
  geom_histogram(aes(y = ..density..),color = 'white')+
  stat_function(fun = dnorm, color = 'red', args = list(mean = 0, sd = 1))+
  labs(x = "Amostra", y = "Densidade")+
  scale_y_continuous(expand = c(0,0), limits = c(0, .42))+
  theme_bw() +
  theme(axis.title.y=element_text(colour="black", size=12),
        axis.title.x = element_text(colour="black", size=12),
        axis.text = element_text(colour = "black", size=9.5),
        panel.border = element_blank(),
        axis.line = element_line(colour = "black"),
        panel.grid = element_blank())
q1 <- sum(am_normrej$amostra <= qnorm(.25))
q2 <- sum(am_normrej$amostra  > qnorm(.25) & am_normrej$amostra <= qnorm(.5))
q3 <- sum(am_normrej$amostra  > qnorm(.5) & am_normrej$amostra <= qnorm(.75))
q4 <- sum(am_normrej$amostra  > qnorm(.75))

quartis <- c(q1, q2, q3, q4)/2000

pvalor <- chisq.test(quartis, p = c(.25, .25, .25, .25))$p.value

polar_long <- am_normpolar %>%
  pivot_longer(cols = everything(), names_to = 'eixo', values_to = 'values')

ggplot(polar_long, aes(values))+
  geom_histogram(aes(y = ..density..),color = 'white')+
  stat_function(fun = dnorm, color = 'red', args = list(mean = 0, sd = 1))+
  labs(x = "Amostra", y = "Densidade")+
  scale_y_continuous(expand = c(0,0), limits = c(0, .42))+
  theme_bw() +
  theme(axis.title.y=element_text(colour="black", size=12),
        axis.title.x = element_text(colour="black", size=12),
        axis.text = element_text(colour = "black", size=9.5),
        panel.border = element_blank(),
        axis.line = element_line(colour = "black"),
        panel.grid = element_blank())+
  facet_wrap(vars(eixo),ncol = 2)
q1 <- sum(am_normpolar$x <= qnorm(.25))
q2 <- sum(am_normpolar$x  > qnorm(.25) & am_normpolar$x <= qnorm(.5))
```

```r
q3 <- sum(am_normpolar$x  > qnorm(.5) & am_normpolar$x <= qnorm(.75))
q4 <- sum(am_normpolar$x  > qnorm(.75))

quartis <- c(q1, q2, q3, q4)/2000

pvalor <- chisq.test(quartis, p = c(.25, .25, .25, .25))$p.value

# ESTIMACAO MONTECARLO
set.seed(1234)
y <- runif(10000)

acumulada_normal <- function(b, n=10000){
  h <- (1/sqrt(2*pi))*(exp((-(y*(b))^2)/2)*(b))
  return(mean(h))
}

intervalos <- seq(0, 3.99, by = 0.01)

montecarlo <- sapply(intervalos, acumulada_normal)+.5

linhas <- seq(0.0,3.9,by=0.1)
colunas <- seq(0.0,0.09,by=0.01)


z <- matrix(montecarlo, ncol=10, byrow=TRUE, dimnames = list(linhas,colunas))

# ESTIMACAO REJEICAO
size <- length(am_normrej$amostra[am_normrej$amostra>0])

estim_rej <- sort(am_normrej$amostra[am_normrej$amostra>0])

intervalos <- seq(0, 3.99, by = 0.01)

rej <- c()

for(i in 1:length(intervalos)){
  rej[i] <- length(estim_rej[estim_rej>0 & estim_rej<= intervalos[i]])/(2*size)
}

rej <- rej+.5

linhas <- seq(0.0,3.9,by=0.1)
colunas <- seq(0.0,0.09,by=0.01)

z <- matrix(rej, ncol=10, byrow=TRUE, dimnames = list(linhas,colunas))

# ESTIMACAO POLAR
size <- length(am_normpolar$x[am_normpolar$x>0])

estim_polar <- sort(am_normpolar$x[am_normpolar$x>0])

intervalos <- seq(0, 3.99, by = 0.01)
polar <- c()
```

```r
for(i in 1:length(intervalos)){
  polar[i] <- length(estim_polar[estim_polar>0 & estim_polar<= intervalos[i]])/(2*size)
}

linhas <- seq(0.0,3.9,by=0.1)
colunas <- seq(0.0,0.09,by=0.01)

polar <- polar+.5

z <- matrix(polar, ncol=10, byrow=TRUE, dimnames = list(linhas,colunas))

# ESTIMACAO DOS ERROS
erros <- data.frame(
  rej, polar, montecarlo,
  normal = NA
)

intervalos <- seq(0, 3.99, by = 0.01)

for(i in 1:length(intervalos)){
  erros$normal[i] <- pnorm(intervalos[i])
}

erros$erro_rej <- (erros$rej - erros$normal)
erros$erro_polar <- (erros$polar - erros$normal)
erros$erro_montecarlo <- (erros$montecarlo - erros$normal)

erros <- erros %>%
  select(starts_with("erro")) %>%
  pivot_longer(cols = everything(),names_to = "tipo", values_to = "erro") %>%
  mutate(quantil = rep(intervalos, each = 3),
         tipo = case_when(
           tipo == "erro_rej" ~ "Rejeição",
           tipo == "erro_polar" ~ "Polar",
           tipo == "erro_montecarlo" ~ "Monte Carlo"
         ))

ggplot(erros, aes(x = quantil, y = erro))+
  geom_point(size = .5)+
  theme_bw() +
  labs(x = "Quantil", y = "Erro")+
  theme(axis.title.y=element_text(colour="black", size=12),
        axis.title.x = element_text(colour="black", size=12),
        axis.text = element_text(colour = "black", size=9.5),
        #panel.border = element_blank(),
        axis.line = element_line(colour = "black")#,
        #panel.grid = element_blank()
        )+
  facet_wrap(vars(tipo), nrow = 1)
```