

STAT572 - Homework Assignment 1

Juan Carlos Apitz, ID 012523821

September 11, 2014

Lab Exercise - Fisher Scoring on the Standard Normal:

To implement this exercise we need to find the score vector and the information matrix of the normal distribution's log-likelihood function:

The score vector is:
$$U(\vec{\theta}) = \begin{bmatrix} \frac{\partial \log \mathcal{L}}{\partial \mu} \\ \frac{\partial \log \mathcal{L}}{\partial \sigma^2} \end{bmatrix} = \begin{bmatrix} \frac{\sum_{i=1}^n x_i - n\mu}{\sigma^2} \\ -\frac{n}{2\sigma^2} + \frac{\sum_{i=1}^n (x_i - \mu)^2}{2(\sigma^2)^2} \end{bmatrix}$$

The information matrix is:
$$I(\vec{\theta}) = -E[H(\vec{\theta})] = \begin{bmatrix} \frac{n}{\sigma^2} & 0 \\ 0 & \frac{n}{2(\sigma^2)^2} \end{bmatrix}$$

The MLE estimate is given by $\hat{\theta} = \theta_0 + I^{-1}(\vec{\theta})U(\vec{\theta})$

Code (see next page):

The code below implements MLE estimation using the Fisher Scoring method on the normal distribution. For simplicity I chose the standard normal, i.e. $\mu = 0$, $\sigma^2 = 1$.

```

Editor - /home/jcapitz/Documents/Stat572/homework/hw1/FisherScoring.m
FisherScoring.m
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % Lab 1. Implement the Fisher Scoring method on a %
3 % simulated sample of size n drawn from a normal dis- %
4 % tribution with mean mu and variance sigma_squared %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
8 % INITIALIZATION BLOCK: %
9 % Set up of initial conditions: matIt is the max # of %
10 % iterations before stop. tol is the convergence tole- %
11 % rance. mu is the distribution mean; sigma_squared %
12 % is the variance; theta_0 is the previous value of %
13 % parameter vector; score is the score vector; and %
14 % fishInfo is the information matrix. %
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16 maxIt=100000;
17 tol=0.000001;
18 theta_0=[1;2];
19 n=100;
20 x=normrnd(0,1,n,1);
21 mu_0=theta_0(1);
22 sig_sq_0=theta_0(2);
23 score=[(sum(x)-n*mu_0)/sig_sq_0;(-n/(2*sig_sq_0))+(sum((x-mu_0).^2)/(2*(sig_sq_0^2)))]';
24 fishInfo=[n/sig_sq_0,0;0,n/(2*(sig_sq_0^2))];
25 theta_hat=theta_0+inv(fishInfo)*score;
26 count=0;
27
28 while norm(theta_0-theta_hat)>tol
29     theta_0=theta_hat;
30     mu_0=theta_hat(1);
31     sig_sq_0=theta_hat(2);
32     score=[(sum(x)-n*mu_0)/sig_sq_0;(-n/(2*sig_sq_0))+(sum((x-mu_0).^2)/(2*(sig_sq_0^2)))]';
33     fishInfo=[n/sig_sq_0,0;0,n/(2*(sig_sq_0^2))];
34     theta_hat=theta_0+inv(fishInfo)*score;
35     count=count+1;
36     if count >= maxIt
37         break
38     end
39 end
40 count
41 theta_hat

```

Figure 1: Code to implement MLE estimation on the standard normal distribution using Fisher Scoring.

As a trial I used $\mu_0 = 1$ and $\sigma_0^2 = 2$ to initialize the algorithm, set up the maximum number of trials to 100,000 and the convergence tolerance to 0.000001. The tolerance is compared to the value $\|\hat{\theta} - \theta_0\|$, a reasonable measure of vector distance between the vector $\hat{\theta}$ and θ_0 . When the tolerance is applied, the algorithm converges in two iterations. I also let ran for 100,000 iterations to see the results. It was interesting to observe that in the longer run the estimates were actually much worse than the two iteration run. The results are shown below with the first vector component of the MATLAB vector **theta_hat** representing the mean, and the second component representing the variance.

```

Editor - /home/jcapitz/Documents/Stat572/homework/hw1/FisherScoring.m
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

>> FisherScoring

count =

     2

theta_hat =

    -0.0411
     1.1082

>> FisherScoring

count =

    100000

theta_hat =

    -0.1370
     1.2056

fx >>

```

Figure 2: Results on implementing MLE estimation on the standard normal distribution using Fisher Scoring.

Exercise 2.1

```

homework ▸ hw1 ▸
Editor - /home/jcapitz/Documents/Stat572/homework/hw1/expcprob_hw21.m
likelihoodGeo.m x newtonRap.m x expcprob_hw21.m x +
1 function prob = expcprob_hw21( x,beta )
2 % This function calculates P(X<x) for an exponential pdf
3 % Exercise 2.1, Martinez
4 function probout = expcprob(x,beta)
5     probout = (1./beta)*exp(-x./beta);
6 end
7 quad(@(x) expcprob(x,beta),0,x)
8 end
9
10

```

Figure 3: Code to calculate $p(X < x_0)$ if $X \sim \exp(\beta)$.

I wrote a short script to create a table that compares the results of the function from exercise 2.1 with the results of calculating $P(X < x_0)$ for the exponential function with $\beta = 1$ using MATLAB's

built-in function $\text{expcdf}(x, \beta)$. My function matches exactly the MATLAB results. See output below:

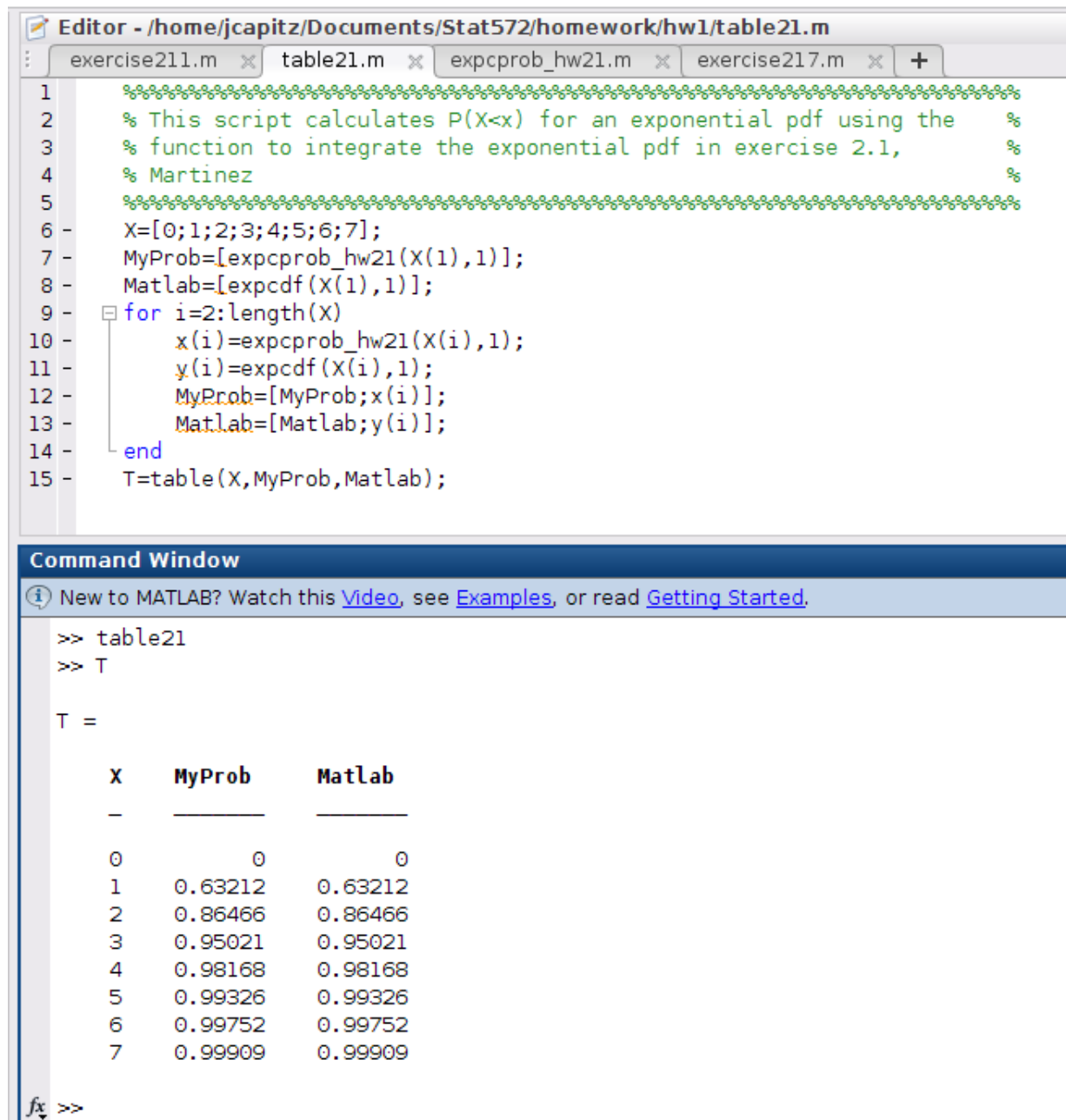


Figure 4: Table showing $p(X < x_0)$ for the exponential cdf using ex. 2.1 function and MATLAB's $\text{expcdf}(x, \beta)$.

Exercise 2.2

In this exercise I simply gave the above functions the input "inf", which in MATLAB it represents infinity. Since the support of the exponential pdf is $(0, \infty)$, the result below confirms that the function developed in ex. 2.1 actually integrates to 1. This confirms this canonical property of probability

functions. In the case of the exponential pdf, we have $\int_0^{\infty} \frac{1}{\beta} e^{-\frac{x}{\beta}} dx = 1$. The result is presented below.

The image shows a MATLAB Editor window with a file named `expcprob_hw21.m`. The code defines a function `expcprob_hw21` that calculates the probability $P(X < x)$ for an exponential distribution with parameter β . The function uses `quadgk` to integrate the probability density function (PDF) from 0 to x . The Command Window shows the execution of `expcprob_hw21(inf,3)`, which returns the value 1.0000.

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 % This function calculates P(X<x) for an exponential pdf                %
3 % Exercise 2.1, Martinez                                                %
4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5 function prob = expcprob_hw21(x,beta )
6     function probout = expcout(x,beta)
7         probout = (1./beta)*exp(-x./beta);
8     end
9     quadgk(@(x)expcout(x,beta),0,x)
10 end

```

Command Window

```

>> expcprob_hw21(inf,3)

ans =

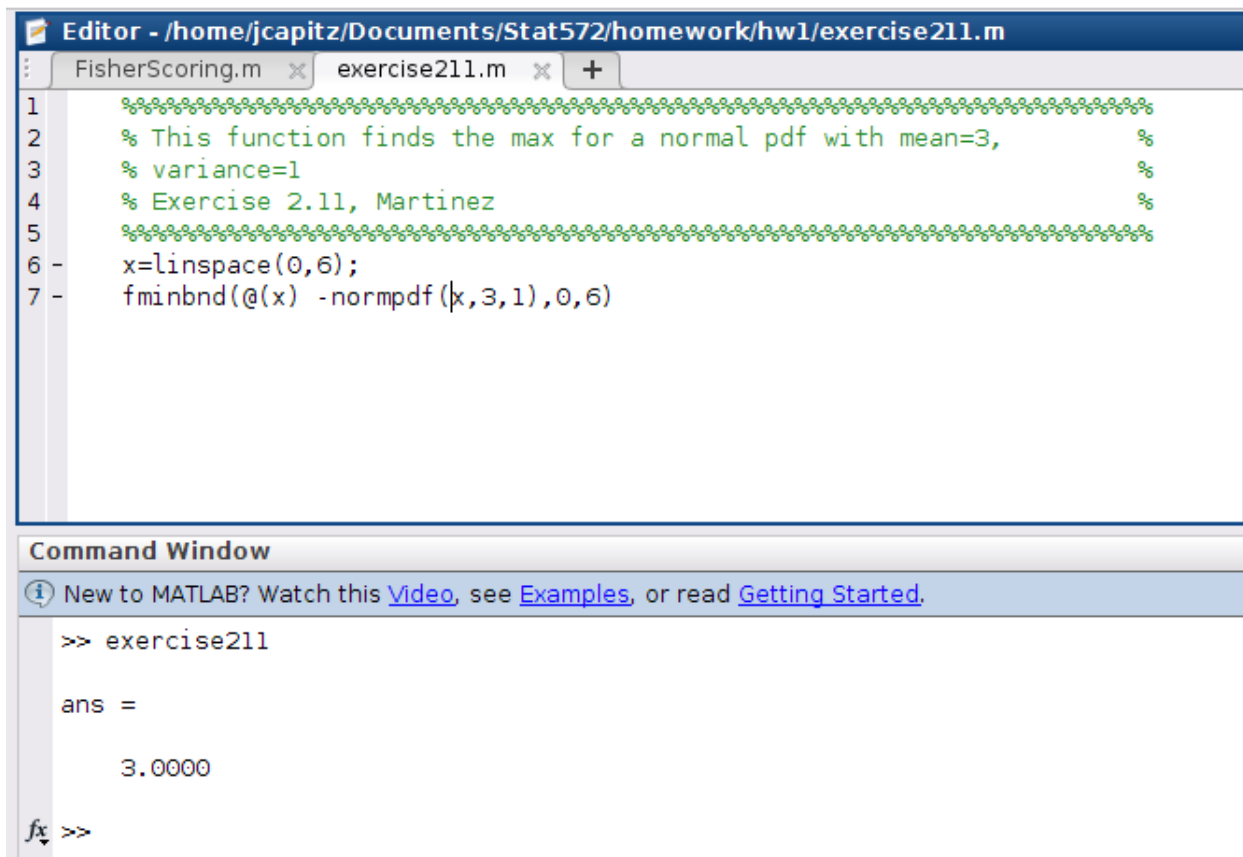
    1.0000

```

Figure 5: Code to calculate $p(0 < X < \infty)$ if $X \sim \exp(\beta = 3)$. The result is 1.

Exercise 2.11

In this exercise we find the maximum for a normal pdf with $\mu = 3$, $\sigma^2 = 1$ using the MATLAB function `fminbnd()`. This function actually find the $\arg\min f(x)$, thus the objective function has to be $-f(x)$.



The image shows a MATLAB Editor window with a file named 'exercise211.m'. The code in the editor is as follows:

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
2 % This function finds the max for a normal pdf with mean=3,           %  
3 % variance=1                                                           %  
4 % Exercise 2.11, Martinez                                              %  
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
6 - x=linspace(0,6);  
7 - fminbnd(@(x) -normpdf(x,3,1),0,6)
```

Below the editor is the Command Window. It shows the command 'exercise211' being executed, and the output 'ans = 3.0000'. The Command Window also includes a message: 'New to MATLAB? Watch this [Video](#), see [Examples](#), or read [Getting Started](#).'

Figure 6: Code to calculate $\operatorname{argmax} f(x)$ where $f(x)$ i.e. a normal distribution with $\mu = 3$, $\sigma^2 = 1$.

The code above implements this maximization with the result $\operatorname{argmax} f(x) = 3$. We know this is correct because this is the value of μ that is most likely for a probability distribution, i.e. this pdf reaches a maximum at the mean.

Exercise 2.17

In this exercise we find $p(X < 3)$ and $p(X > 5)$ for a random variable $X \sim N(\mu = 5, \sigma^2 = 4)$ using MATLAB's `normspec()` and `normcdf()`.

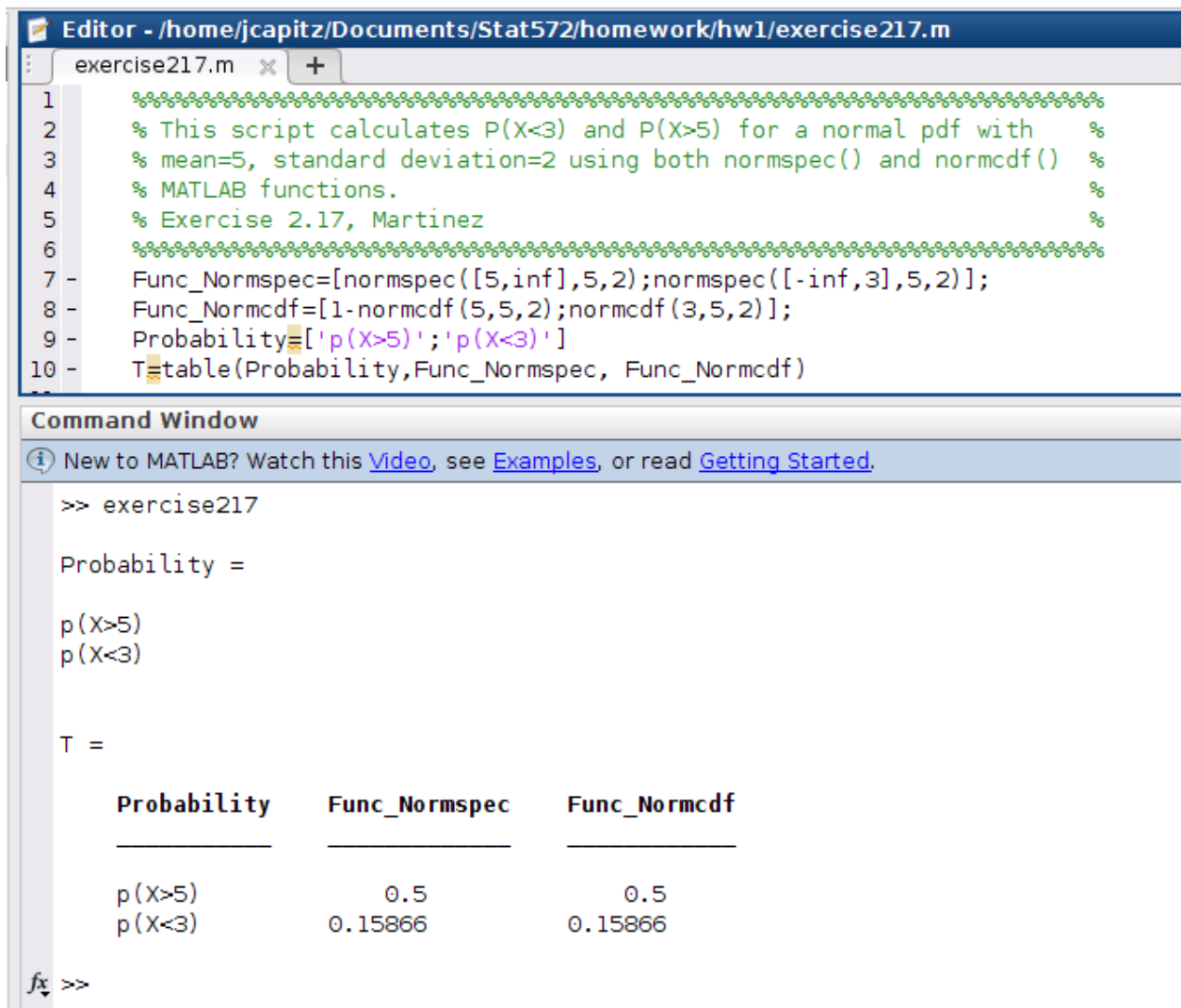


Figure 7: Code to calculate $p(X > 5)$ and $p(X < 3)$ using `normspec()` and `normcdf()`.

The above code computes the required probabilities and places the result on a MATLAB table for comparison. Both function results agree as expected, although to compute $p(X > 5)$ we need to use the reciprocal probability calculation $1 - \text{normcdf}(X, \mu, \sigma)$ since `normcdf()` only computes $p(X < x_0)$. For `normspec()` we can actually input `inf` and get a result. In addition `normspec()` also provides a graphical output for this calculation. See figure 8 and 9 below.

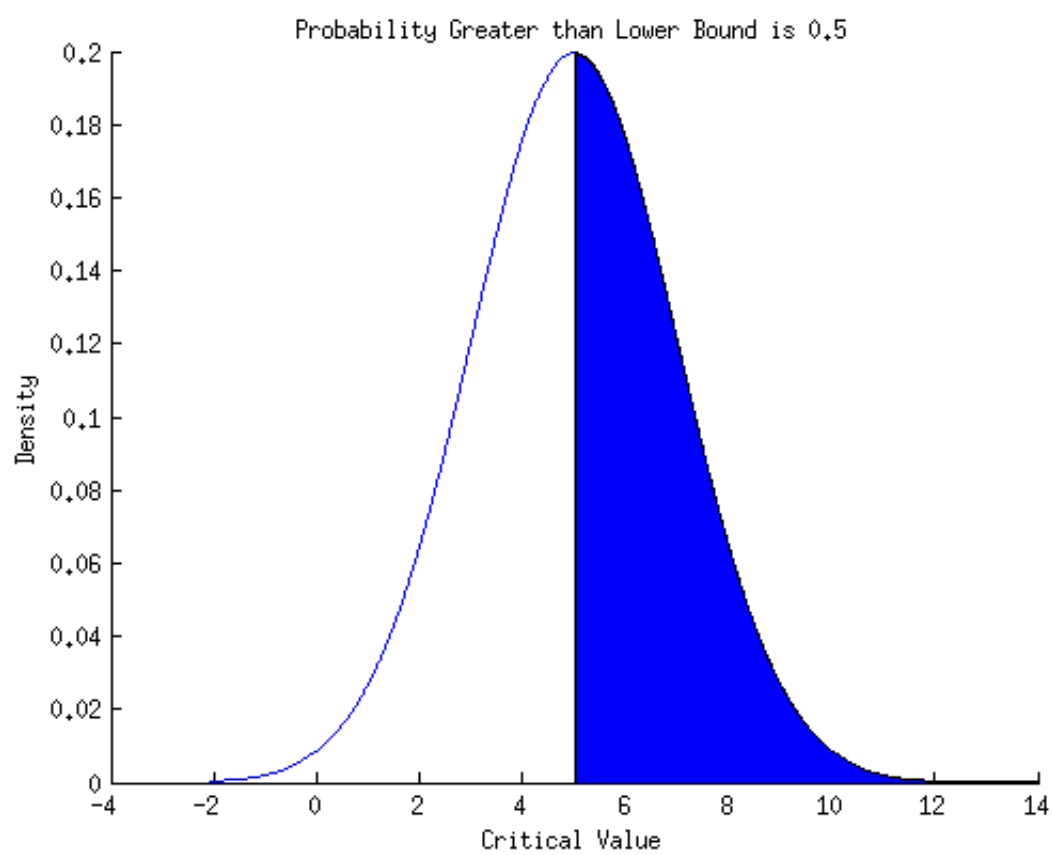


Figure 8: $p(X > 5)$ using `normspec()`.

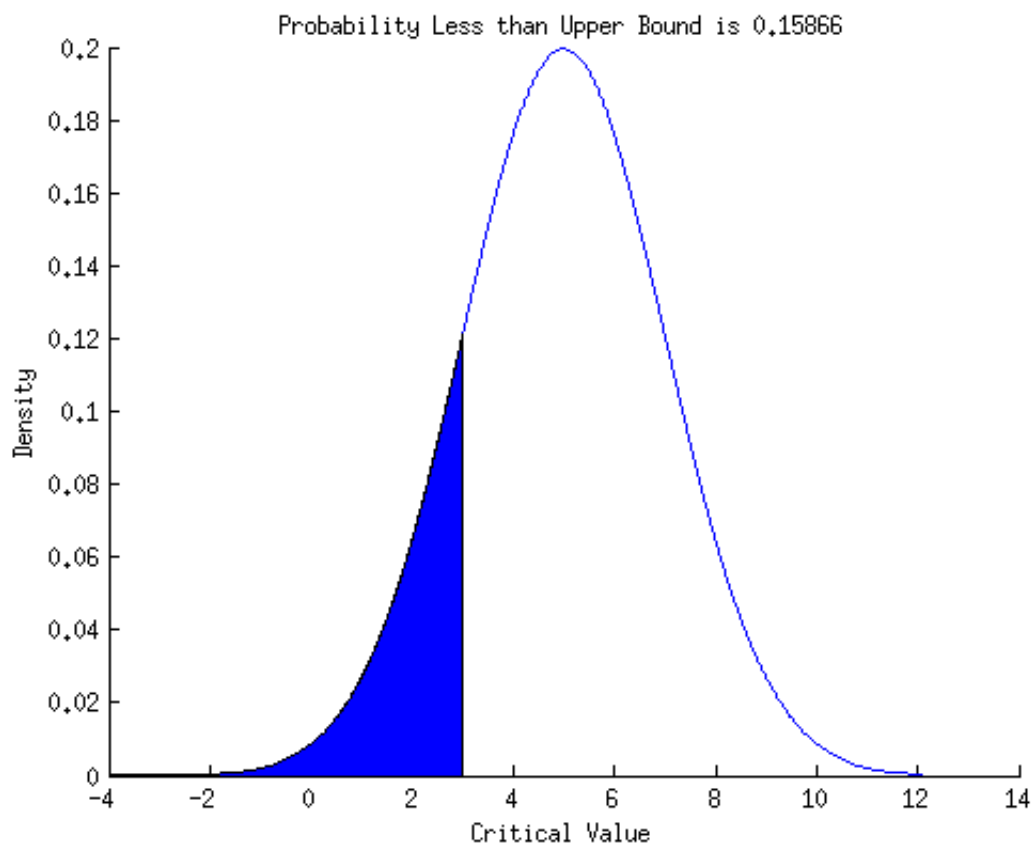


Figure 9: $p(X < 3)$ using `normspec()`.