Hypothesis testing

Contents

1 Student's t-test

The main purpose of this document is to make difficult aspects of hypothesis testing easier to comprehend, in particular the significance levels and the values from the statistical table containing critical values. This table is also called the t table. The critical values are required to reject the null hypothesis of a statistical test. To make understanding of significance levels and critical values topics easier to understand, MATLAB is used to write functions that visually display these subjects. In the function files we will use the limiting distributions of the corresponding statistical tests.

When manually using the statistical table, two values are required to find the correct critical value of the hypothesis test in this table. The two required inputs are the significance level and the number of observations minus one, which is also known as the degrees of freedom. Therefore, the minimal information required for the functions to work are the significance level and the degrees of freedom. After filling in these values and running the code, a graph displaying the limiting distribution and the value from the statistical table will be shown. The user also has the option to give the manually calculated test statistic as input to the function file. In this case, this test statistic will also be displayed, from which the user is able to see if the corresponding null hypothesis can be rejected.

All the code can be found here on GitHub.

1.1 Using the functions

To help the learner understand the code, the `TTesting.m` file has been written. This file explains what input is required for the function and offers the ability to run the function files.

The t-test can be right tailed, left tailed or two tailed. The side of the test depends on the formulation of the null and alternative hypotheses. When the hypotheses are formulated as

$$\begin{aligned} H_0 &: \hat{\beta}_1 = \beta_1^0 \\ H_1 &: \hat{\beta}_1 \neq \beta_1^0 \end{aligned} \tag{1}$$

a two tailed test is required. In case that the alternative hypothesis H_1 is $\hat{\beta}_1 < \beta_1^0$, a left tailed test needs to be performed. The last case is a right tailed test, which is when the alternative hypothesis H_1 is formulated as $\hat{\beta}_1 > \beta_1^0$.

```
% -----------------------------------------------------------------------
% When using TTest(SIDE, NU, ALPHA, TSTAT),
%
% SIDE indicates the side of the hypothesis test. Input for SIDE can be
% any of the three following:
%   - 'RightSided' for a right-tailed t-test
%   - 'LeftSided' for a left-tailed t-test
%   - 'TwoSided' for a two-tailed t-test
%
% NU is the degrees of freedom and can be any positive valued integer.
%
% ALPHA denotes the significance level. The input for ALPHA can be any
% real valued number on the interval (0,1).
%
% TSTAT is the manually derived t-statistic, which can take on any finite
% value. This input argument is optional and in case input is given, the
% function plots the manually calculated test statistic valued TSTAT and
```

```
% show whether the null hypothesis can or cannot be rejected.
%
% The size of the plot is predetermined and can be customized by changing
% the values of TTR.scale, TTL.scale, TTTS.scale in their corresponding
% function files.
%
% TTest(SIDE, NU, ALPHA, TSTAT)
% -----------------------------------------------------------------------
% Example
TTest('TwoSided', 35, 0.05, -2.3)
```

As can be observed, an example command has been written on the last line of the function file. With that line of code, we visualize a two tailed t-test with degrees of freedom $\nu = 35$, a significance level $\alpha = 0.05$ and a test statistic valued -2.3. The result is the following graph:

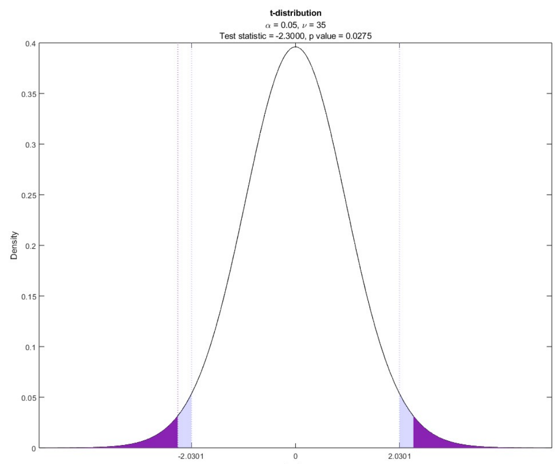

Figure 1: t-test example

1.2 t-test function

When a statistics learner wants to perform the hypothesis test corresponding to equation (1), they will use the Student's t-test. The function `TTest.m` helps the learner's understanding of the two difficult subjects by visualizing the underlying statistical meaning and motivation of the tests. The function `TTest.m` checks what side of the test the user wants to perform and then 'sends' the task to the corresponding function.

```
function TT = TTest(side, nu, alpha, tstat)
%TTEST Visualize a Student's t-test
%   TTest(SIDE, NU, ALPHA, TSTAT) plots the theoretical
%   Student's t-distribution with NU degrees of freedom. Depending on the
%   input argument for SIDE, it calculates one or two critical values
%   corresponding to a one or two sided t-test with NU degrees of freedom
%   at an ALPHA level of significance and plots the related rejection
%   region(s). A vertical line representing the manually calculated
%   test statistic valued TSTAT will be plotted as well, this input
%   argument is optional.
```

```
%
%   SIDE can be:
%       'RightSided',
%       'LeftSided',
%       'TwoSided'.

% ---------------------------------------------------------------------------
% Check if the user does not want to plot the test statistic, which
% happens when the number of input arguments is equal to three.
% ---------------------------------------------------------------------------
if (nargin == 3)
    switch(side)
        case "RightSided"
            TTestRight(nu, alpha);
        case "LeftSided"
            TTestLeft(nu, alpha);
        case "TwoSided"
            TTestTwoSided(nu, alpha);
        otherwise
            uiwait(warndlg(['Please specify the correct side, right, ' ...
                'left or two sided']))
            return
    end
else
    switch(side)
        case "RightSided"
            TTestRight(nu, alpha, tstat);
        case "LeftSided"
            TTestLeft(nu, alpha,tstat);
        case "TwoSided"
            TTestTwoSided(nu, alpha, tstat);
        otherwise
            uiwait(warndlg(['Please specify the correct side, right, ' ...
                'left or two sided']))
            return
    end
end
```

1.3 Right tailed test

When the alternative hypothesis H_1 is $\hat{\beta}_1 > \beta_1^0$, `TTestRight.m` can be used. A significance level of $\alpha = 0.05$ is commonly used in hypothesis testing. This means in for a right sided test, we want to find the $95th$ percentile.

When calculating the critical value of the t test, the inverse cumulative distribution function (ICDF) needs to be used. The ICDF is also known as the quantile function. The inverse CDF tells us what value of x is needed to make $F_X(x)$ return our required probability p.

We use this ICDF to calculate the critical value since we want to find what value of the test statistic is at least or at most required (depending on the alternative hypothesis) to reject the null hypothesis. Since we make use of the Student's t-distribution, the inverse cumulative distribution function also requires the degrees of freedom as input.

```
function TTR = TTestRight(nu, alpha, tstat)
```

```matlab
%TTESTRIGHT Visualize a right sided Student's t-test
%   TTR = TTestRight(NU, ALPHA, TSTAT) plots the theoretical
%   student's t-distribution with NU degrees of freedom. It calculates the
%   critical value corresponding to a right sided t-test with NU degrees
%   of freedom at an ALPHA level of significance and plots the related
%   rejection region. A vertical line representing the manually calculated
%   test statistic valued TSTAT will be plotted, this input argument is
%   optional.

% -----------------------------------------------------------------------
% Check whether the input is valid.
% -----------------------------------------------------------------------
if (nu <= 0)
    uiwait(warndlg('The degrees of freedom should be larger than zero.'));
    return
elseif (mod(nu, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of ' ...
        'freedom.']));
    return
elseif (alpha <= 0 || alpha >= 1)
    uiwait(warndlg(['Please fill in a value of alpha between zero and '...
        'one.']));
    return
end

% -----------------------------------------------------------------------
% Check if the user wants to plot the test statistic.
% -----------------------------------------------------------------------
if (nargin == 3)
    TTR.Display = 1;
else
    TTR.Display = 0;
end

% -----------------------------------------------------------------------
% Calculating the critical value.
% -----------------------------------------------------------------------
TTR.CV = icdf('T', 1-alpha, nu);

% -----------------------------------------------------------------------
% Determining the length of the horizontal axis, which depends on the
% critical value as having only one or two degrees of freedom can result
% in a large critical value. In most cases however, the critical value is
% relatively small and [-5, 5] is a good interval to display the curve of
% the student's t-distribution and the crititcal value. xmin uses -TTR.CV
% as the critical value is positive by construction.
% -----------------------------------------------------------------------
TTR.xmin = min([-TTR.CV-1 -5]);
TTR.xmax = max([TTR.CV+1 5]);
TTR.x = TTR.xmin:0.01:TTR.xmax;

% -----------------------------------------------------------------------
% Creating the density.
% -----------------------------------------------------------------------
```

```matlab
TTR.y = pdf('T', TTR.x, nu);

% ---------------------------------------------------------------------------
% Calculating the rejection region, as the area needs to be shown in the
% plot
% ---------------------------------------------------------------------------
TTR.xright = TTR.CV:0.001:TTR.xmax;
TTR.yright = pdf('T', TTR.xright, nu);

% ---------------------------------------------------------------------------
% Setting up the plot. To create a subtitle consisting of two lines, the
% sprintf() function is used in the subtitle() function. The subtitle will
% be split up in two lines when the user wants to plot the test statistic
% and calculate the p value. TTR.alphadec is used to determine the number
% of decimals for displaying alpha, which depends on the user and hence is
% dynamic. TTR.nodec is used for the degrees of freedom, which don't have
% decimals. The code then asks for the size of the monitor of the user to
% calculate the size (in pixels) of the graph. The standard 4:3 ratio is
% used. TTR.scale scales the graph with respect to the monitor size of the
% user. xticks is used as it is necessary to show the exact critical value
% on the horizontal axis.
% ---------------------------------------------------------------------------
TTR.alphadec = sprintf('%%.%df', ...
    length(char(extractAfter(string(alpha),'.'))));
TTR.nodec = sprintf('%%.%df', 0);
TTR.variables = sprintf(['\\alpha = ', TTR.alphadec, ', \\nu = ', ...
    TTR.nodec], alpha, nu);

TTR.mp = get(0, 'MonitorPositions');
TTR.mwidth = TTR.mp(1, 3);
TTR.mheight = TTR.mp(1, 4);
TTR.scale = 0.45;

TTR.gwidth = TTR.scale*TTR.mwidth;
TTR.gheight = 0.75*TTR.gwidth;
TTR.x0 = 0.5*TTR.mwidth*(1 - TTR.scale);
TTR.y0 = (TTR.mheight - TTR.gheight - 84)*0.5;

figure
plot(TTR.x,TTR.y,'-black');
xticks([0 TTR.CV]);
title("t-distribution");
subtitle({TTR.variables}, 'Interpreter', 'tex');
xlabel("t-value");
ylabel("Density");
TTR.fig = gcf;
TTR.fig.Position = [TTR.x0, TTR.y0, TTR.gwidth, TTR.gheight];
hold on

% ---------------------------------------------------------------------------
% Marking the critical values in the plot.
% ---------------------------------------------------------------------------
xline([TTR.CV], 'LineStyle', ':', 'Color', '#9a9afc', 'LineWidth', 1.4);

```

```
% -----------------------------------------------------------------------
% Filling the areas of the rejection region.
% -----------------------------------------------------------------------
TTR.ar = area(TTR.xright, TTR.yright);
TTR.ar.FaceColor = 'blue';
TTR.ar.FaceAlpha = 0.15;
TTR.ar.EdgeColor = 'none';

% -----------------------------------------------------------------------
% The user has the option to also plot the self calculated test statistic
% and compute the corresponding p value. This part of the code will only
% run when there is input for the third argument (the value of the test
% statistic). In the case that there is no input for the third argument,
% the p value will not be calculated and the function has finished
% running.
%
% The first nested if else statement will check, on the condition that the
% user gave input for the third argument, if the test statistic is smaller
% than the critical value. If this is the case, the null hypothesis cannot
% be rejected and a vertical light purple dotted line corresponding to the
% value of the test statistic and a light purple shaded area will be added
% to the plot. Else, the null can be rejected and a purple vertical dotted
% line corresponding to the value of the test statistic and a dark purple
% shaded area will be plotted.
%
% Afterwards the subtitle is updated and the code has finished
% running.
% -----------------------------------------------------------------------
if (TTR.Display == 1)
    if (tstat < TTR.CV)
        xline(tstat, 'LineStyle', ':', 'Color', '#ae9ab5', 'LineWidth',...
            1.4);
        TTR.tint = tstat:0.001:TTR.CV;
        TTR.ty = pdf('T', TTR.tint, nu);
        TTR.tar = area(TTR.tint, TTR.ty);
        TTR.tar.FaceColor = '#8a22b3';
        TTR.tar.FaceAlpha = 0.04;
        TTR.tar.EdgeColor = 'none';
    else
        xline(tstat, 'LineStyle', ':', 'Color','#8a22b3', 'LineWidth', ...
            1.4);
        TTR.tint = tstat:0.001:TTR.xmax;
        TTR.ty = pdf('T', TTR.tint, nu);
        TTR.tar = area(TTR.tint, TTR.ty);
        TTR.tar.FaceColor = '#8a22b3';
        TTR.tar.FaceAlpha = 1;
        TTR.tar.EdgeColor = 'none';
    end
    TTR.pval = 1-cdf('T', tstat, nu);
    TTR.empdec = sprintf('%%.%df', 4);
    TTR.pdec = sprintf('%%.%df', 2);
    TTR.tp = sprintf(['Test statistic = ', TTR.empdec,', p value = ', ...
        TTR.pdec], tstat,TTR.pval);
    subtitle({TTR.variables, TTR.tp}, 'Interpreter', 'tex');
```

```
end
```

1.4 Left tailed test

Instead of a right tailed test, it is also possible to perform a left tailed test. The alternative hypothesis H_1 then becomes $\hat{\beta}_1 < \beta_1^0$. Keeping the significance level of $\alpha = 0.05$, we now want to find the $5th$ percentile. Due to symmetry of the t-distribution at zero, this $5th$ percentile has the same value as the negative value of the $95th$ percentile, which is used in the function `TTestLeft.m`.

```
function TTL = TTestLeft(nu, alpha, tstat)
%TTESTLEFT Visualize a left sided Student's t-test
%   TTL = TTestLeft(NU, ALPHA, TSTAT) plots the theoretical
%   student's t-distribution with NU degrees of freedom. It calculates the
%   critical value corresponding to a left sided t-test with NU degrees of
%   freedom at an ALPHA level of significance and plots the related
%   rejection region. A vertical line representing the manually calculated
%   test statistic valued TSTAT will be plotted, this input argument is
%   optional.

% -----------------------------------------------------------------------
% Check whether the input is valid.
% -----------------------------------------------------------------------
if (nu <= 0)
    uiwait(warndlg('The degrees of freedom should be larger than zero.'));
    return
elseif (mod(nu, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of ' ...
        'freedom.']));
    return
elseif (alpha <= 0 || alpha >= 1)
    uiwait(warndlg(['Please fill in a value of alpha between zero and '...
        'one.']));
    return
end

% -----------------------------------------------------------------------
% Check if the user wants to plot the test statistic.
% -----------------------------------------------------------------------
if (nargin == 3)
    TTL.Display = 1;
else
    TTL.Display = 0;
end

% -----------------------------------------------------------------------
% Calculating the critical value.
% -----------------------------------------------------------------------
TTL.CV = -icdf('T', 1-alpha, nu);

% -----------------------------------------------------------------------
% Determining the length of the horizontal axis, which depends on the
% critical value as having only one or two degrees of freedom can result
```

```
% in a large critical value. In most cases however, the critical value is
% relatively small and [-5, 5] is a good interval to display the curve of
% the student's t-distribution and the crititcal values. xmax uses -TTL.CV
% as the critical value is negative by construction.
% -------------------------------------------------------------------------
TTL.xmin = min([TTL.CV-1 -5]);
TTL.xmax = max([-TTL.CV+1 5]);
TTL.x = TTL.xmin:0.01:TTL.xmax;

% -------------------------------------------------------------------------
% Creating the density.
% -------------------------------------------------------------------------
TTL.y = pdf('T', TTL.x, nu);

% -------------------------------------------------------------------------
% Calculating the rejection region, since this area needs to be shown in
% the plot.
% -------------------------------------------------------------------------
TTL.xleft = TTL.xmin:0.001:TTL.CV;
TTL.yleft = pdf('T', TTL.xleft, nu);

% -------------------------------------------------------------------------
% Setting up the plot. To create a subtitle consisting of two lines, the
% sprintf() function is used in the subtitle() function. The subtitle will
% be split up in two lines when the user wants to plot the test statistic
% and calculate the p value. TTL.alphadec is used to determine the number
% of decimals for displaying alpha, which depends on the user and hence is
% dynamic. TTL.nodec is used for the degrees of freedom, which don't have
% decimals. The code then asks for the size of the monitor of the user to
% calculate the size (in pixels) of the graph. The standard 4:3 ratio is
% used. TTL.scale scales the graph with respect to the monitor size of the
% user. xticks is used as it is necessary to show the exact critical value
% on the horizontal axis.
% -------------------------------------------------------------------------
TTL.alphadec = sprintf('%%.%df', ...
    length(char(extractAfter(string(alpha),'.'))));
TTL.nodec = sprintf('%%.%df', 0);
TTL.variables = sprintf(['\\alpha = ', TTL.alphadec, ', \\nu = ', ...
    TTL.nodec], alpha, nu);

TTL.mp = get(0, 'MonitorPositions');
TTL.mwidth = TTL.mp(1, 3);
TTL.mheight = TTL.mp(1, 4);
TTL.scale = 0.45;

TTL.gwidth = TTL.scale*TTL.mwidth;
TTL.gheight = 0.75*TTL.gwidth;
TTL.x0 = 0.5*TTL.mwidth*(1 - TTL.scale);
TTL.y0 = (TTL.mheight - TTL.gheight - 84)*0.5;

figure
plot(TTL.x,TTL.y,'-black');
xticks([TTL.CV 0]);
title("t-distribution");
```

```matlab
subtitle({TTL.variables}, 'Interpreter', 'tex');
xlabel("t-value");
ylabel("Density");
TTL.fig = gcf;
TTL.fig.Position = [TTL.x0, TTL.y0, TTL.gwidth, TTL.gheight];
hold on

% ------------------------------------------------------------------------
% Marking the critical value in the plot.
% ------------------------------------------------------------------------
xline([TTL.CV], 'LineStyle', ':', 'Color', '#9a9afc', 'LineWidth', 1.4);

% ------------------------------------------------------------------------
% Filling the area of the rejection region.
% ------------------------------------------------------------------------
TTL.ar = area(TTL.xleft, TTL.yleft);
TTL.ar.FaceColor = 'blue';
TTL.ar.FaceAlpha = 0.15;
TTL.ar.EdgeColor = 'none';

% ------------------------------------------------------------------------
% The user has the option to also plot the self calculated test statistic
% and compute the corresponding p value. This part of the code will only
% run when there is input for the third argument (the value of the test
% statistic). In the case that there is no input for the third argument,
% the p value will not be calculated and the function has finished
% running.
%
% The first nested if else statement will check, on the condition that the
% user gave input for the third argument, if the test statistic is larger
% than the critical value. If this is the case, the null hypothesis cannot
% be rejected and a vertical light purple dotted line corresponding to the
% value of the test statistic and a light purple shaded area representing
% the p value will be added to the plot. Else, the null can be rejected
% and a purple vertical dotted line corresponding to the value of the test
% statistic and a dark purple shaded area displaying the p value will be
% plotted.
%
% Afterwards the subtitle will be updated and the code has finished
% running.
% ------------------------------------------------------------------------
if (TTL.Display == 1)
    if (tstat > TTL.CV)
        xline(tstat, 'LineStyle', ':', 'Color', '#ae9ab5', 'LineWidth',...
            1.4);
        TTL.tint = TTL.CV:0.001:tstat;
        TTL.ty = pdf('T', TTL.tint, nu);
        TTL.tar = area(TTL.tint, TTL.ty);
        TTL.tar.FaceColor = '#8a22b3';
        TTL.tar.FaceAlpha = 0.04;
        TTL.tar.EdgeColor = 'none';
    else
        xline(tstat, 'LineStyle', ':', 'Color','#8a22b3', 'LineWidth', ...
            1.4);
```

```
        TTL.tint = TTL.xmin:0.001:tstat;
        TTL.ty = pdf('T', TTL.tint, nu);
        TTL.tar = area(TTL.tint, TTL.ty);
        TTL.tar.FaceColor = '#8a22b3';
        TTL.tar.FaceAlpha = 1;
        TTL.tar.EdgeColor = 'none';
    end
    TTL.pval = cdf('T', tstat, nu);
    TTL.empdec = sprintf('%%.%df', 4);
    TTL.pdec = sprintf('%%.%df', 4);
    TTL.tp = sprintf(['Test statistic = ', TTL.empdec,', p value = ', ...
        TTL.pdec], tstat, TTL.pval);
    subtitle({TTL.variables, TTL.tp}, 'Interpreter', 'tex');
end
```

1.5 Two tailed test

If the alternative hypothesis H_1 is specified as in equation (1), a two tailed test needs to be performed. The function `TTestTwoSided.m` will do so. For a two tailed test, two critical values need to be calculated. If in this case we also use the significance level $\alpha = 0.05$, we need to calculate the $2.5th$ and $97.5th$ percentile: the significance level α is 'equally divided' for both sides.

The same approach goes for the p value: the p value of a two tailed t-test is equal to double the p value in one tail. To visualize the p value, the code needs to find on which intervals the tails of the distribution need to be shaded. To do this, the code will first evaluate whether the test statistic is negative or positive valued (i.e., lies in the left or right tail). Then, depending on the sign of the test statistic, the codes takes the negative or positive 'counterpart' of the test statistic and is now able to shade both tails. The method works because the Student's t-distribution is symmetric around zero.

```
function TTTS = TTestTwoSided(nu, alpha, tstat)
%TTESTTWOSIDED Visualize a two sided Student's t-test
%   TTTS = TTestTwoSided(NU, ALPHA, TSTAT) plots the theoretical
%   student's t-distribution with NU degrees of freedom. It calculates the
%   two critical values corresponding to a two sided t-test with NU
%   degrees of freedom at an ALPHA level of significance and plots the
%   related rejection regions. A vertical line representing the manually
%   calculated test statistic valued TSTAT will be plotted, this input
%   argument is optional.

% -----------------------------------------------------------------------
% Check whether the input is valid.
% -----------------------------------------------------------------------
if (nu <= 0)
    uiwait(warndlg('The degrees of freedom should be larger than zero.'));
    return
elseif (mod(nu, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of ' ...
        'freedom.']));
    return
elseif (alpha <= 0 || alpha >= 1)
    uiwait(warndlg(['Please fill in a value of alpha between zero and ' ...
```

```
        'one.']));
    return
end

% ------------------------------------------------------------------------
% Check if the user wants to plot the test statistic.
% ------------------------------------------------------------------------
if (nargin == 3)
    TTTS.Display = 1;
else
    TTTS.Display = 0;
end

% ------------------------------------------------------------------------
% Calculating the critical values.
% ------------------------------------------------------------------------
TTTS.CV = icdf('T', 1-alpha/2, nu);
TTTS.CVleft = -TTTS.CV;
TTTS.CVright = TTTS.CV;

% ------------------------------------------------------------------------
% Determining the length of the horizontal axis, which depends on the
% critical values as having only one or two degrees of freedom can result
% in large critical values. In most cases however, the critical values are
% relatively small and [-5, 5] is a good interval to display the curve of
% the student's t-distribution and the crititcal values.
% ------------------------------------------------------------------------
TTTS.xmin = min([TTTS.CVleft-1 -5]);
TTTS.xmax = max([TTTS.CVright+1 5]);
TTTS.x = TTTS.xmin:0.01:TTTS.xmax;

% ------------------------------------------------------------------------
% Creating the density.
% ------------------------------------------------------------------------
TTTS.y = pdf('T', TTTS.x, nu);

% ------------------------------------------------------------------------
% Calculating the rejection regions, as these areas need to be shown in
% the plot.
% ------------------------------------------------------------------------
TTTS.xleft = TTTS.xmin:0.001:TTTS.CVleft;
TTTS.xright = TTTS.CVright:0.001:TTTS.xmax;
TTTS.yleft = pdf('T', TTTS.xleft, nu);
TTTS.yright = pdf('T', TTTS.xright, nu);

% ------------------------------------------------------------------------
% Setting up the plot. To create a subtitle consisting of two lines, the
% sprintf() function is used in the subtitle() function. The subtitle will
% be split up in two lines when the user wants to plot the test statistic
% and calculate the p value. TTTS.alphadec is used to determine the number
% of decimals for displaying alpha, which depends on the user and hence is
% dynamic. TTTS.nodec is used for the degrees of freedom, which don't have
% decimals. The code then asks for the size of the monitor of the user to
% calculate the size (in pixels) of the graph. The standard 4:3 ratio is
```

```
% used. TTTS.scale scales the graph with respect to the monitor size of
% the user. xticks is used as it is necessary to show the exact critical
% value on the horizontal axis.
% -----------------------------------------------------------------------
TTTS.alphadec = sprintf('%%.%df', ...
    length(char(extractAfter(string(alpha),'.'))));
TTTS.nodec = sprintf('%%.%df', 0);
TTTS.variables = sprintf(['\\alpha = ', TTTS.alphadec, ', \\nu = ', ...
    TTTS.nodec], alpha, nu);

TTTS.mp = get(0, 'MonitorPositions');
TTTS.mwidth = TTTS.mp(1, 3);
TTTS.mheight = TTTS.mp(1, 4);
TTTS.scale = 0.45;

TTTS.gwidth = TTTS.scale*TTTS.mwidth;
TTTS.gheight = 0.75*TTTS.gwidth;
TTTS.x0 = 0.5*TTTS.mwidth*(1 - TTTS.scale);
TTTS.y0 = (TTTS.mheight - TTTS.gheight - 84)*0.5;

figure
plot(TTTS.x,TTTS.y,'-black');
xticks([TTTS.CVleft 0 TTTS.CVright]);
title("t-distribution");
subtitle({TTTS.variables}, 'Interpreter', 'tex');
xlabel("t-value");
ylabel("Density");
TTTS.fig = gcf;
TTTS.fig.Position = [TTTS.x0, TTTS.y0, TTTS.gwidth, TTTS.gheight];
hold on

% -----------------------------------------------------------------------
% Marking the critical values in the plot.
% -----------------------------------------------------------------------
xline([TTTS.CVleft TTTS.CVright], 'LineStyle', ':', 'Color', '#9a9afc',...
    'LineWidth', 1.4);

% -----------------------------------------------------------------------
% Filling the areas of the rejection region.
% -----------------------------------------------------------------------
TTTS.arleft = area(TTTS.xleft, TTTS.yleft);
TTTS.arleft.FaceColor = 'blue';
TTTS.arleft.FaceAlpha = 0.15;
TTTS.arleft.EdgeColor = 'none';

TTTS.arright = area(TTTS.xright, TTTS.yright);
TTTS.arright.FaceColor = 'blue';
TTTS.arright.FaceAlpha = 0.15;
TTTS.arright.EdgeColor = 'none';

% -----------------------------------------------------------------------
% The user has the option to also plot the self calculated test statistic
% and compute the corresponding p value. This part of the code will only
% run when there is input for the third argument (the value of the test
```

```
% statistic). In the case that there is no input for the third argument,
% the p value will not be calculated and the function has finished
% running.
%
% The first nested if else statement will check, on the condition that the
% user gave input for the third argument, if the absolute value of the
% test statistic is smaller than the critical value. If this is the case,
% the null hypothesis cannot be rejected and a vertical light purple
% dotted line corresponding to the value of the test statistic and a
% light purple shaded area will be added to the plot. Else, the null can
% be rejected and a purple vertical dotted line corresponding to the value
% of the test statistic and a dark purple shaded area will be plotted.
%
% Afterwards the subtitle will be updated and the code has finished
% running.
% -------------------------------------------------------------------------
if (TTTS.Display == 1)
    if (abs(tstat) < TTTS.CV)
        xline(tstat, 'LineStyle', ':', 'Color', '#ae9ab5', 'LineWidth',...
            1.4);
        TTTS.tintleft = TTTS.CVleft:0.001:-abs(tstat);
        TTTS.tintright = abs(tstat):0.001:TTTS.CVright;

        TTTS.tyl = pdf('T', TTTS.tintleft, nu);
        TTTS.tlar = area(TTTS.tintleft, TTTS.tyl);
        TTTS.tlar.FaceColor = '#8a22b3';
        TTTS.tlar.FaceAlpha = 0.04;
        TTTS.tlar.EdgeColor = 'none';

        TTTS.tyr = pdf('T', TTTS.tintright, nu);
        TTTS.trar = area(TTTS.tintright, TTTS.tyr);
        TTTS.trar.FaceColor = '#8a22b3';
        TTTS.trar.FaceAlpha = 0.04;
        TTTS.trar.EdgeColor = 'none';
    else
        xline(tstat, 'LineStyle', ':', 'Color','#8a22b3', 'LineWidth', ...
            1.4);
        TTTS.tintleft = TTTS.xmin:0.001:-abs(tstat);
        TTTS.tintright = abs(tstat):0.001:TTTS.xmax;

        TTTS.tyl = pdf('T', TTTS.tintleft, nu);
        TTTS.tlar = area(TTTS.tintleft, TTTS.tyl);
        TTTS.tlar.FaceColor = '#8a22b3';
        TTTS.tlar.FaceAlpha = 1;
        TTTS.tlar.EdgeColor = 'none';

        TTTS.tyr = pdf('T', TTTS.tintright, nu);
        TTTS.trar = area(TTTS.tintright, TTTS.tyr);
        TTTS.trar.FaceColor = '#8a22b3';
        TTTS.trar.FaceAlpha = 1;
        TTTS.trar.EdgeColor = 'none';
    end
    TTTS.pval = 2*cdf('T', -abs(tstat), nu);
    TTTS.empdec = sprintf('%%.%df', 4);
```

```
    TTTS.pdec = sprintf('%%.%df', 4);
    TTTS.tp = sprintf(['Test statistic = ', TTTS.empdec,', p value = ',...
        TTTS.pdec], tstat, TTTS.pval);
    subtitle({TTTS.variables, TTTS.tp}, 'Interpreter', 'tex');
end
```

2 F-test

When the goal of the hypothesis test focuses on the variances of two groups, an F-test can be performed. Under the null hypothesis, the F statistics follows an F-distribution. If $X \sim \chi^2_{\nu_1}$ and $Y \sim \chi^2_{\nu_2}$ (Chi-squared distributed with ν_1 and ν_2 degrees of freedom respectively), then

$$Z = \frac{X/\nu_1}{Y/\nu_2} \sim F_{\nu_1,\nu_2}$$

Z is F-distributed with ν_1 and ν_2 degrees of freedom. ν_1 is also called the numerator degrees of freedom and ν_2 the denominator degrees of freedom. The usual form that the corresponding hypothesis test is used is:

$$H_0 : \sigma_1^2 \leq \sigma_2^2$$
$$H_1 : \sigma_1^2 > \sigma_2^2$$

With this test, we determine whether there is enough 'evidence' that the variance of the first group is larger than the variance of the second group.

When manually using the statistical table, three values are required to find the correct critical value of the hypothesis test in this table. These three values are the significance level, the numerator degrees of freedom and the denominator degrees of freedom. For every significance level, a specific table exists. The column headings denote the numerator degrees of freedom and the row headings the denominator degrees of freedom. Therefore, the minimal information required for the functions to work are the significance level and the two degrees of freedom. After filling in these values and running the code, a graph displaying the limiting distribution and the value from the statistical table will be shown. The user also has the option to give the manually calculated test statistic as input to the function file. In this case, this test statistic will also be displayed, from which the user is able to see if the corresponding null hypothesis can be rejected.

2.1 Using the functions

To help the learner understand the code, the `FTesting.m` file has been written. This file explains what input is required for the function and offers the ability to run the function files.

The F-test can be right tailed, left tailed or two tailed. The side of the test depends on the formulation of the alternative hypothesis. When the alternative hypothesis is formulated as

$$H_1 : \sigma_1^2 \neq \sigma_2^2,$$

a two tailed test is required. In case that the alternative hypothesis H_1 is $\sigma_1^2 > \sigma_2^2$, a right tailed test needs to be performed. The last case is a left tailed test, which is when the alternative hypothesis H_1 is formulated as $\sigma_1^2 < \sigma_2^2$.

```
% -----------------------------------------------------------------------
% When using  FTest(SIDE, NU1, NU2, ALPHA, TSTAT),
%
% SIDE indicates the side of the hypothesis test. Input for SIDE can be
% any of the three following:
%    - 'RightSided' for a right-tailed F-test
%    - 'LeftSided' for a left-tailed F-test
%    - 'TwoSided' for a two-tailed F-test
%
% NU1 and NU2 are the degrees of freedom and can be any positive valued
% integer.
%
% ALPHA denotes the significance level. The input for ALPHA can be any
% real valued number on the interval (0,1).
%
% TSTAT is the manually derived F-statistic, which can take on any
% nonnegative value. This input argument is optional and in case input is
% given, the function plots the manually calculated test statistic valued
% TSTAT and show whether the null hypothesis can or cannot be rejected.
%
% The size of the plot is predetermined and can be customized by changing
% the values of FTR.scale, FTL.scale, FTTS.scale in their corresponding
% function files.
%
% FTest(SIDE, NU1, NU2, ALPHA, TSTAT)
% -----------------------------------------------------------------------
% Example
FTest('RightSided', 30, 20, 0.05, 1.8)
```

When we run the example command that has been written on the last line of the function file, we visualize a right tailed F-test with numerator degrees of freedom $\nu_1 = 30$, denominator degrees of freedom $\nu_2 = 20$, a significance level $\alpha = 0.05$ and a test statistic valued 1.8. The result is the following graph:

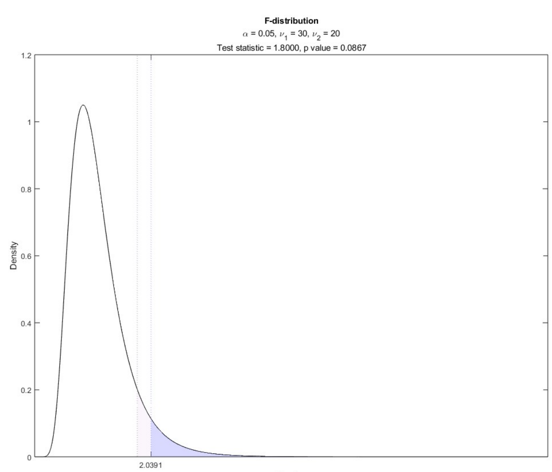

Figure 2: F-test example

2.2 F-test function

The function file `FTest.m` helps the learner's understanding of the two difficult subjects by visualizing the underlying statistical motivation and meaning of the tests. The function file `FTest.m` checks what side of the test the user wants to perform and then 'sends' the task to the corresponding function file.

```
function FT = FTest(side, nu1, nu2, alpha, tstat)
%FTEST Visualize an F-test
%   FTest(SIDE, NU1, NU2, ALPHA, TSTAT) plots the theoretical
%   F-distribution with NU1 and NU2 degrees of freedom. Depending on the
%   input argument for SIDE, it calculates one or two critical values
%   corresponding to a one or two sided F-test with NU1 and NU2 degrees of
%   freedom at an ALPHA level of significance and plots the related
%   rejection region(s). A vertical line representing the manually
%   calculated test statistic valued TSTAT will be plotted as well, this
%   input argument is optional.
%
%   SIDE can be:
%      'RightSided',
%      'LeftSided',
%      'TwoSided'.

% -----------------------------------------------------------------------
% Check if the user does not want to plot the test statistic, which
% happens when the number of input arguments is equal to four.
% -----------------------------------------------------------------------
if (nargin == 4)
    switch(side)
        case "RightSided"
            FTestRight(nu1, nu2, alpha);
        case "LeftSided"
            FTestLeft(nu1, nu2, alpha);
        case "TwoSided"
            FTestTwoSided(nu1, nu2, alpha);
        otherwise
            uiwait(warndlg(['Please specify the correct side, right, ' ...
                'left or two sided']))
            return
    end
else
    switch(side)
        case "RightSided"
            FTestRight(nu1, nu2, alpha, tstat);
        case "LeftSided"
            FTestLeft(nu1, nu2, alpha,tstat);
        case "TwoSided"
            FTestTwoSided(nu1, nu2, alpha, tstat);
        otherwise
            uiwait(warndlg(['Please specify the correct side, right, ' ...
                'left or two sided']))
            return
    end
end
```

2.3 Right tailed test

The function file `FTestRight.m` can be used when the alternative hypothesis H_1 is $\sigma_1^2 > \sigma_2^2$.

```
function FTR = FTestRight(nu1, nu2, alpha, tstat)
%FTESTRIGHT Visualize a right sided F-test
%   FTR = FTestLeft(NU1, NU2, ALPHA, TSTAT) plots the theoretical
%   F-distribution with NU1 and NU2 degrees of freedom. It calculates the
%   critical value corresponding to a right sided F-test with NU1 and NU2
%   degrees of freedom at an ALPHA level of significance and plots the
%   related rejection region. A vertical line representing the manually
%   calculated test statistic valued TSTAT will be plotted, this input
%   argument is optional.

% -------------------------------------------------------------------------
% Check whether the input is valid.
% -------------------------------------------------------------------------
if (nu1 <= 0)
    uiwait(warndlg(['The degrees of freedom of the first population' ...
        ' should be larger than zero.']));
    return
elseif (nu2 <= 0)
    uiwait(warndlg(['The degrees of freedom of the second population' ...
        ' should be larger than zero.']));
    return
elseif (mod(nu1, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of' ...
        ' freedom of the first population.']));
    return
elseif (mod(nu2, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of' ...
        ' freedom of the second population.']));
    return
elseif (alpha <= 0 || alpha >= 1)
    uiwait(warndlg(['Please fill in a value of alpha between zero and' ...
        ' one.']));
    return
end

% -------------------------------------------------------------------------
% Check if the user wants to plot the test statistic and make sure that
% the test statistic is nonnegative valued.
% -------------------------------------------------------------------------
if (nargin == 4)
    if (tstat < 0)
        uiwait(warndlg(['The test statistic cannot be negative valued,'...
            ' please make sure that the test statistic is correctly' ...
            ' calculated.']));
        return
    end
    FTR.Display = 1;
else
    FTR.Display = 0;
end

```

```
% ----------------------------------------------------------------------------
% Calculating the critical value.
% ----------------------------------------------------------------------------
FTR.CV = icdf('F', 1-alpha, nu1, nu2);

% ----------------------------------------------------------------------------
% Determining the length of the horizontal axis, which depends on the
% critical value as having only one or two degrees of freedom can result
% in large critical value. In most cases however, the critical value is
% relatively small and [0, 9] is a good interval to display the curve of
% the F-distribution and the crititcal value.
% ----------------------------------------------------------------------------
FTR.xmin = 0;
FTR.xmax = max([FTR.CV+3 9]);
FTR.x = FTR.xmin:0.01:FTR.xmax;

% ----------------------------------------------------------------------------
% Creating the density.
% ----------------------------------------------------------------------------
FTR.y = pdf('F', FTR.x, nu1, nu2);

% ----------------------------------------------------------------------------
% Calculating the rejection region, as the area needs to be shown in the
% plot.
% ----------------------------------------------------------------------------
FTR.xright = FTR.CV:0.001:FTR.xmax;
FTR.yright = pdf('F', FTR.xright, nu1, nu2);

% ----------------------------------------------------------------------------
% Setting up the plot. To create a subtitle consisting of two lines, the
% sprintf() function is used in the subtitle() function. The subtitle will
% be split up in two lines when the user wants to plot the test statistic
% and calculate the p value. FTR.alphadec is used to determine the number
% of decimals for displaying alpha, which depends on the user and hence is
% dynamic. FTR.nodec is used for the degrees of freedom, which don't have
% decimals. The code then asks for the size of the monitor of the user to
% calculate the size (in pixels) of the graph. The standard 4:3 ratio is
% used. FTR.scale scales the graph with respect to the monitor size of the
% user. xticks is used as it is necessary to show the exact critical value
% on the horizontal axis.
% ----------------------------------------------------------------------------
FTR.alphadec = sprintf('%%.%df', ...
    length(char(extractAfter(string(alpha),'.'))));
FTR.nodec = sprintf('%%.%df', 0);
FTR.variables = sprintf(['\\alpha = ', FTR.alphadec, ', \\nu_{1} = ', ...
    FTR.nodec, ', \\nu_{2} = ' FTR.nodec], alpha, nu1, nu2);

FTR.mp = get(0, 'MonitorPositions');
FTR.mwidth = FTR.mp(1, 3);
FTR.mheight = FTR.mp(1, 4);
FTR.scale = 0.45;

FTR.gwidth = FTR.scale*FTR.mwidth;
FTR.gheight = 0.75*FTR.gwidth;
```

```
FTR.x0 = 0.5*FTR.mwidth*(1 - FTR.scale);
FTR.y0 = (FTR.mheight - FTR.gheight - 84)*0.5;

figure
plot(FTR.x,FTR.y,'-black');
xticks([FTR.CV]);
title("F-distribution")
subtitle({FTR.variables}, 'Interpreter', 'tex');
xlabel("F-value");
ylabel("Density");
FTR.fig = gcf;
FTR.fig.Position = [FTR.x0, FTR.y0, FTR.gwidth, FTR.gheight];
hold on

% ----------------------------------------------------------------------
% Marking the critical value in the plot.
% ----------------------------------------------------------------------
xline([FTR.CV], 'LineStyle', ':', 'Color', '#9a9afc', 'LineWidth', 1.4);

% ----------------------------------------------------------------------
% Filling the area of the rejection region.
% ----------------------------------------------------------------------
FTR.ar = area(FTR.xright, FTR.yright);
FTR.ar.FaceColor = 'blue';
FTR.ar.FaceAlpha = 0.15;
FTR.ar.EdgeColor = 'none';

% ----------------------------------------------------------------------
% The user has the option to also plot the self calculated test statistic
% and compute the corresponding p value. This part of the code will only
% run when there is input for the fourth argument (the value of the test
% statistic). In the case that there is no input for the fourth argument,
% the p value will not be calculated and the function has finished
% running.
%
% The first nested if else statement will check, on the condition that the
% user gave input for the fourth argument, if the test statistic is
% smaller than the critical value. If this is the case, the null
% hypothesis cannot be rejected and a vertical light purple dotted line
% corresponding to the value of the test statistic and a light purple
% shaded area will be added to the plot. Else, the null can be rejected
% and a purple vertical dotted line corresponding to the value of the
% test statistic and a dark purple shaded area will be plotted.
%
% Afterwards the subtitle is updated and the code has finished
% running.
% ----------------------------------------------------------------------
if (FTR.Display == 1)
    if (tstat < FTR.CV)
        xline(tstat, 'LineStyle', ':', 'Color', '#ae9ab5', 'LineWidth',...
            1.4);
        FTR.tint = tstat:0.001:FTR.CV;
        FTR.ty = pdf('F', FTR.tint, nu1, nu2);
        FTR.tar = area(FTR.tint, FTR.ty);
```

```
        FTR.tar.FaceColor = '#8a22b3';
        FTR.tar.FaceAlpha = 0.04;
        FTR.tar.EdgeColor = 'none';
    else
        xline(tstat, 'LineStyle', ':', 'Color','#8a22b3', 'LineWidth', ...
            1.4);
        FTR.tint = tstat:0.001:FTR.xmax;
        FTR.ty = pdf('F', FTR.tint, nu1, nu2);
        FTR.tar = area(FTR.tint, FTR.ty);
        FTR.tar.FaceColor = '#8a22b3';
        FTR.tar.FaceAlpha = 1;
        FTR.tar.EdgeColor = 'none';
    end
    FTR.pval = 1-cdf('F', tstat, nu1, nu2);
    FTR.empdec = sprintf('%%.%df', 4);
    FTR.pdec = sprintf('%%.%df', 4);
    FTR.tp = sprintf(['Test statistic = ', FTR.empdec,', p value = ', ...
        FTR.pdec], tstat,FTR.pval);
    subtitle({FTR.variables, FTR.tp}, 'Interpreter', 'tex');
end
```

2.4 Left tailed test

When the alternative hypothesis H_1 is defined as $\sigma_1^2 < \sigma_2^2$, the function file `FTestLeft.m` can be used.

```
function FTL = FTestLeft(nu1, nu2, alpha, tstat)
%FTESTLEFT Visualize a left sided F-test
%   FTL = FTestLeft(NU1, NU2, ALPHA, TSTAT) plots the theoretical
%   F-distribution with NU1 and NU2 degrees of freedom. It calculates the
%   critical value corresponding to a left sided F-test with NU1 and NU2
%   degrees of freedom at an ALPHA level of significance and plots the
%   related rejection region. A vertical line representing the manually
%   calculated test statistic valued TSTAT will be plotted, this input
%   argument is optional.

% -------------------------------------------------------------------------
% Check whether the input is valid.
% -------------------------------------------------------------------------
if (nu1 <= 0)
    uiwait(warndlg(['The degrees of freedom of the first population' ...
        ' should be larger than zero.']));
    return
elseif (nu2 <= 0)
    uiwait(warndlg(['The degrees of freedom of the second population' ...
        ' should be larger than zero.']));
    return
elseif (mod(nu1, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of' ...
        ' freedom of the first population.']));
    return
elseif (mod(nu2, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of' ...
```

```
        ' freedom of the second population.']));
    return
elseif (alpha <= 0 || alpha >= 1)
    uiwait(warndlg(['Please fill in a value of alpha between zero and' ...
        ' one.']));
    return
end

% ------------------------------------------------------------------------
% Check if the user wants to plot the test statistic and make sure that
% the test statistic is nonnegative valued.
% ------------------------------------------------------------------------
if (nargin == 4)
    if (tstat < 0)
        uiwait(warndlg(['The test statistic cannot be negative valued,'...
            ' please make sure that the test statistic is correctly' ...
            ' calculated.']));
        return
    end
    FTL.Display = 1;
else
    FTL.Display = 0;
end

% ------------------------------------------------------------------------
% Calculating the critical value.
% ------------------------------------------------------------------------
FTL.CV = icdf('F', alpha, nu1, nu2);

% ------------------------------------------------------------------------
% Determining the length of the horizontal axis, which depends on the
% critical value as having only one or two degrees of freedom can result
% in large critical value. In most cases however, the critical value is
% relatively small and [0, 9] is a good interval to display the curve of
% the F-distribution and the crititcal value. To display the distribution
% better in a dynamic way, the right end value of the interval depends on
% the critical value of a right sided F-test. This is because the critical
% value of a left sided F-test does not increase as much as the thickness
% of the right sided tail of the F-distribution when the degrees of
% freedom increase.
% ------------------------------------------------------------------------
FTL.betterright = icdf('F', 1-alpha, nu2, nu1);
FTL.xmin = 0;
FTL.xmax = min([FTL.betterright+3 9]);
FTL.x = FTL.xmin:0.01:FTL.xmax;

% ------------------------------------------------------------------------
% Creating the density.
% ------------------------------------------------------------------------
FTL.y = pdf('F', FTL.x, nu1, nu2);

% ------------------------------------------------------------------------
% Calculating the rejection region, as the area needs to be shown in the
% plot.
```

```
% ------------------------------------------------------------------------
FTL.xleft = FTL.xmin:0.001:FTL.CV;
FTL.yleft = pdf('F', FTL.xleft, nu1, nu2);

% ------------------------------------------------------------------------
% Setting up the plot. To create a subtitle consisting of two lines, the
% sprintf() function is used in the subtitle() function. The subtitle will
% be split up in two lines when the user wants to plot the test statistic
% and calculate the p value. FTL.alphadec is used to determine the number
% of decimals for displaying alpha, which depends on the user and hence is
% dynamic. FTL.nodec is used for the degrees of freedom, which don't have
% decimals. The code then asks for the size of the monitor of the user to
% calculate the size (in pixels) of the graph. The standard 4:3 ratio is
% used. FTL.scale scales the graph with respect to the monitor size of the
% user. xticks is used as it is necessary to show the exact critical value
% on the horizontal axis.
% ------------------------------------------------------------------------
FTL.alphadec = sprintf('%%.%df', ...
    length(char(extractAfter(string(alpha),'.'))));
FTL.nodec = sprintf('%%.%df', 0);
FTL.variables = sprintf(['\\alpha = ', FTL.alphadec, ', \\nu_{1} = ', ...
    FTL.nodec, ', \\nu_{2} = ' FTL.nodec], alpha, nu1, nu2);

FTL.mp = get(0, 'MonitorPositions');
FTL.mwidth = FTL.mp(1, 3);
FTL.mheight = FTL.mp(1, 4);
FTL.scale = 0.45;

FTL.gwidth = FTL.scale*FTL.mwidth;
FTL.gheight = 0.75*FTL.gwidth;
FTL.x0 = 0.5*FTL.mwidth*(1 - FTL.scale);
FTL.y0 = (FTL.mheight - FTL.gheight - 84)*0.5;

figure
plot(FTL.x,FTL.y,'-black');
xticks([FTL.CV]);
title("F-distribution");
subtitle({FTL.variables}, 'Interpreter', 'tex');
xlabel("F-value");
ylabel("Density");
FTL.fig = gcf;
FTL.fig.Position = [FTL.x0, FTL.y0, FTL.gwidth, FTL.gheight];
hold on

% ------------------------------------------------------------------------
% Marking the critical value in the plot.
% ------------------------------------------------------------------------
xline([FTL.CV], 'LineStyle', ':', 'Color', '#9a9afc', 'LineWidth', 1.4);

% ------------------------------------------------------------------------
% Filling the area of the rejection region.
% ------------------------------------------------------------------------
FTL.ar = area(FTL.xleft, FTL.yleft);
FTL.ar.FaceColor = 'blue';
```

```
FTL.ar.FaceAlpha = 0.15;
FTL.ar.EdgeColor = 'none';

% -------------------------------------------------------------------------
% The user has the option to also plot the self calculated test statistic
% and compute the corresponding p value. This part of the code will only
% run when there is input for the fourth argument (the value of the test
% statistic). In the case that there is no input for the fourth argument,
% the p value will not be calculated and the function has finished
% running.
%
% The first nested if else statement will check, on the condition that the
% user gave input for the fourth argument, if the test statistic is larger
% than the critical value. If this is the case, the null hypothesis cannot
% be rejected and a vertical light purple dotted line corresponding to the
% value of the test statistic and a light purple shaded area will be added
% to the plot. Else, the null can be rejected and a purple vertical dotted
% line corresponding to the value of the test statistic and a dark purple
% shaded area will be plotted.
%
% Afterwards the subtitle is updated and the code has finished
% running.
% -------------------------------------------------------------------------
if (FTL.Display == 1)
    if (tstat > FTL.CV)
        xline(tstat, 'LineStyle', ':', 'Color', '#ae9ab5', 'LineWidth',...
            1.4);
        FTL.tint = FTL.CV:0.001:tstat;
        FTL.ty = pdf('F', FTL.tint, nu1, nu2);
        FTL.tar = area(FTL.tint, FTL.ty);
        FTL.tar.FaceColor = '#8a22b3';
        FTL.tar.FaceAlpha = 0.04;
        FTL.tar.EdgeColor = 'none';
    else
        xline(tstat, 'LineStyle', ':', 'Color','#8a22b3', 'LineWidth', ...
            1.4);
        FTL.tint = FTL.xmin:0.001:tstat;
        FTL.ty = pdf('F', FTL.tint, nu1, nu2);
        FTL.tar = area(FTL.tint, FTL.ty);
        FTL.tar.FaceColor = '#8a22b3';
        FTL.tar.FaceAlpha = 1;
        FTL.tar.EdgeColor = 'none';
    end
    FTL.pval = cdf('F', tstat, nu1, nu2);
    FTL.empdec = sprintf('%%.%df', 4);
    FTL.pdec = sprintf('%%.%df', 4);
    FTL.tp = sprintf(['Test statistic = ', FTL.empdec,', p value = ', ...
        FTL.pdec], tstat, FTL.pval);
    subtitle({FTL.variables, FTL.tp}, 'Interpreter', 'tex');
end
```

2.5 Two tailed test

Lastly, it is also possible that the F-test is two tailed. This occurs when we have that $H_1 : \sigma_1^2 \neq \sigma_2^2$. The function file `FTestTwoSided.m` has been written to visualize the two tailed F-test.

Similar to the two tailed t-test, the p value of the test statistic is derived by doubling the p value. However, since the F-distribution is not symmetric and its support is $[0, \infty)$, it is not possible to find the 'counterpart' of the test statistic by multiplying its value by negative one.

In the case that the degrees of freedom ν_1 and ν_2 are equal to each other, the 'counterpart' of the test statistic can be found by deriving its reciprocal.

If the two degrees of freedom are not equal to each other, taking the reciprocal will not give the correct value. This problem has been solved by noting that the p value is equally divided in the two tails. The code determines whether the percentile of the test statistic is larger than or equal to the $50th$ percentile. If this is the case, the 'counterpart' is somewhere below the $50th$ percentile. The code will then continue by calculating the p value of the tail which contains the test statistic. Depending on in which tail the 'counterpart' should be, the inverse cumulative function is used to derive the value of the 'counterpart'. The code has now gathered all required information to shade the p value in both tails. This method also works when the two degrees of freedom are equal to each other.

```matlab
function FTTS = FTestTwoSided(nu1, nu2, alpha, tstat)
%FTESTTWOSIDED Visualize a two sided F-test
%   FTTS = FTestTwoSided(NU, ALPHA, TSTAT) plots the theoretical
%   F-distribution with NU1 and NU2 degrees of freedom. It calculates the
%   two critical values corresponding to a two sided F-test with NU1 and
%   NU2 degrees of freedom at an ALPHA level of significance and plots the
%   related rejection regions. A vertical line representing the manually
%   calculated test statistic valued TSTAT will be plotted, this input
%   argument is optional.

% ------------------------------------------------------------------------
% Check whether the input is valid.
% ------------------------------------------------------------------------
if (nu1 <= 0)
    uiwait(warndlg(['The degrees of freedom of the first population' ...
        ' should be larger than zero.']));
    return
elseif (nu2 <= 0)
    uiwait(warndlg(['The degrees of freedom of the second population' ...
        ' should be larger than zero.']));
    return
elseif (mod(nu1, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of' ...
        ' freedom of the first population.']));
    return
elseif (mod(nu2, 1) ~= 0)
    uiwait(warndlg(['Please fill in an integer for the degrees of' ...
        ' freedom of the second population.']));
    return
elseif (alpha <= 0 || alpha >= 1)
    uiwait(warndlg(['Please fill in a value of alpha between zero and' ...
        ' one.']));
    return
end
```

```
% ---------------------------------------------------------------------------
% Check if the user wants to plot the test statistic and make sure that
% the test statistic is nonnegative valued.
% ---------------------------------------------------------------------------
if (nargin == 4)
    if (tstat < 0)
        uiwait(warndlg(['The test statistic cannot be negative valued,'...
            ' please make sure that the test statistic is correctly' ...
            ' calculated.']));
        return
    end
    FTTS.Display = 1;
else
    FTTS.Display = 0;
end

% ---------------------------------------------------------------------------
% Calculating the critical values.
% ---------------------------------------------------------------------------
FTTS.CVleft = icdf('F', alpha/2, nu1, nu2);
FTTS.CVright = icdf('F', 1-alpha/2, nu1, nu2);

% ---------------------------------------------------------------------------
% Determining the length of the horizontal axis, which depends on the
% critical values as having only one or two degrees of freedom can result
% in large critical value. In most cases however, the critical values are
% relatively small and [0, 9] is a good interval to display the curve of
% the F-distribution and the crititcal values.
% ---------------------------------------------------------------------------
FTTS.xmin = 0;
FTTS.xmax = min([FTTS.CVright+3 9]);
FTTS.x = FTTS.xmin:0.01:FTTS.xmax;

% ---------------------------------------------------------------------------
% Creating the density.
% ---------------------------------------------------------------------------
FTTS.y = pdf('F', FTTS.x, nu1, nu2);

% ---------------------------------------------------------------------------
% Calculating the rejection regions, as these areas need to be shown in
% the plot.
% ---------------------------------------------------------------------------
FTTS.xleft = FTTS.xmin:0.01:FTTS.CVleft;
FTTS.xright = FTTS.CVright:0.01:FTTS.xmax;
FTTS.yleft = pdf('F', FTTS.xleft, nu1, nu2);
FTTS.yright = pdf('F', FTTS.xright, nu1, nu2);

% ---------------------------------------------------------------------------
% Setting up the plot. To create a subtitle consisting of two lines, the
% sprintf() function is used in the subtitle() function. The subtitle will
% be split up in two lines when the user wants to plot the test statistic
% and calculate the p value. FTTS.alphadec is used to determine the number
% of decimals for displaying alpha, which depends on the user and hence is
```

```
% dynamic. FTTS.nodec is used for the degrees of freedom, which don't have
% decimals. The code then asks for the size of the monitor of the user to
% calculate the size (in pixels) of the graph. The standard 4:3 ratio is
% used. FTTS.scale scales the graph with respect to the monitor size of
% the user. xticks is used as it is necessary to show the exact critical
% value on the horizontal axis.
% ------------------------------------------------------------------------
FTTS.alphadec = sprintf('%%.%df', ...
    length(char(extractAfter(string(alpha),'.'))));
FTTS.nodec = sprintf('%%.%df', 0);
FTTS.variables = sprintf(['\\alpha = ', FTTS.alphadec, ', \\nu_{1} = ',...
    FTTS.nodec, ', \\nu_{2} = ' FTTS.nodec], alpha, nu1, nu2);

FTTS.mp = get(0, 'MonitorPositions');
FTTS.mwidth = FTTS.mp(1, 3);
FTTS.mheight = FTTS.mp(1, 4);
FTTS.scale = 0.45;

FTTS.gwidth = FTTS.scale*FTTS.mwidth;
FTTS.gheight = 0.75*FTTS.gwidth;
FTTS.x0 = 0.5*FTTS.mwidth*(1 - FTTS.scale);
FTTS.y0 = (FTTS.mheight - FTTS.gheight - 84)*0.5;

figure
plot(FTTS.x,FTTS.y,'-black');
xticks([FTTS.CVleft FTTS.CVright]);
title("F-distribution");
subtitle({FTTS.variables}, 'Interpreter', 'tex');
xlabel("F-value");
ylabel("Density");
xlim([0 FTTS.xmax]);
FTTS.fig = gcf;
FTTS.fig.Position = [FTTS.x0, FTTS.y0, FTTS.gwidth, FTTS.gheight];
hold on

% ------------------------------------------------------------------------
% Marking the critical values in the plot.
% ------------------------------------------------------------------------
xline([FTTS.CVleft FTTS.CVright], 'LineStyle', ':', 'Color', '#9a9afc',...
    'LineWidth', 1.4);

% ------------------------------------------------------------------------
% Filling the areas of the rejection region.
% ------------------------------------------------------------------------
FTTS.arleft = area(FTTS.xleft, FTTS.yleft);
FTTS.arleft.FaceColor = 'blue';
FTTS.arleft.FaceAlpha = 0.15;
FTTS.arleft.EdgeColor = 'none';

FTTS.arright = area(FTTS.xright, FTTS.yright);
FTTS.arright.FaceColor = 'blue';
FTTS.arright.FaceAlpha = 0.15;
FTTS.arright.EdgeColor = 'none';

```

```matlab
% ------------------------------------------------------------------------
% The user has the option to also plot the self calculated test statistic
% and compute the corresponding p value. This part of the code will only
% run when there is input for the fourth argument (the value of the test
% statistic). In the case that there is no input for the fourth argument,
% the p value will not be calculated and the function has finished
% running.
%
% The first nested if else statement will check, on the condition that the
% user gave input for the fourth argument, if the value of the test
% statistic lies between the two critical values. If this is the case, the
% null hypothesis cannot be rejected and a vertical light purple dotted
% line corresponding to the value of the test statistic and a light purple
% shaded area will be added to the plot. Else, the null can be rejected
% and a purple vertical dotted line corresponding to the value of the test
% statistic and a dark purple shaded area will be plotted.
%
% Afterwards the subtitle is updated and the code has finished running.
% ------------------------------------------------------------------------
if (FTTS.Display == 1)
    % Check if null can be rejected
    if (tstat > FTTS.CVleft && tstat < FTTS.CVright)
        xline(tstat, 'LineStyle', ':', 'Color', '#ae9ab5', 'LineWidth', ...
            1.4);
        % Determine the interval of the light purple shaded area
        if (tstat >= icdf('F', 0.5, nu1, nu2))
            FTTS.otherpside = 1 - cdf('F', tstat, nu1, nu2);
            FTTS.otherstat = icdf('F', FTTS.otherpside, nu1, nu2);
            FTTS.tintleft = FTTS.CVleft:0.001:FTTS.otherstat;
            FTTS.tintright = tstat:0.001:FTTS.CVright;
            FTTS.pval = (1 - cdf('F', tstat, nu1, nu2))*2;
        else
            FTTS.otherpside = cdf('F', tstat, nu1, nu2);
            FTTS.otherstat = icdf('F', 1 - FTTS.otherpside, nu1, nu2);
            FTTS.tintleft = FTTS.CVleft:0.001:tstat;
            FTTS.tintright = FTTS.otherstat:0.001:FTTS.CVright;
            FTTS.pval = 2*cdf('F', tstat, nu1, nu2);
        end
        FTTS.tyl = pdf('F', FTTS.tintleft, nu1, nu2);
        FTTS.tlar = area(FTTS.tintleft, FTTS.tyl);
        FTTS.tlar.FaceColor = '#8a22b3';
        FTTS.tlar.FaceAlpha = 0.04;
        FTTS.tlar.EdgeColor = 'none';

        FTTS.tyr = pdf('F', FTTS.tintright, nu1, nu2);
        FTTS.trar = area(FTTS.tintright, FTTS.tyr);
        FTTS.trar.FaceColor = '#8a22b3';
        FTTS.trar.FaceAlpha = 0.04;
        FTTS.trar.EdgeColor = 'none';
    else
        xline(tstat, 'LineStyle', ':', 'Color','#8a22b3', 'LineWidth', ...
            1.4);
        % Determine the interval of the dark purple shaded area
        if (tstat >= FTTS.CVright)
```

```
            FTTS.otherpside = 1 - cdf('F', tstat, nu1, nu2);
            FTTS.otherstat = icdf('F', FTTS.otherpside, nu1, nu2);
            FTTS.tintleft = FTTS.xmin:0.001:FTTS.otherstat;
            FTTS.tintright = tstat:0.001:FTTS.xmax;
            FTTS.pval = (1 - cdf('F', tstat, nu1, nu2))*2;
        else
            FTTS.otherpside = cdf('F', tstat, nu1, nu2);
            FTTS.otherstat = icdf('F', 1 - FTTS.otherpside, nu1, nu2);
            FTTS.tintleft = FTTS.xmin:0.001:tstat;
            FTTS.tintright = FTTS.otherstat:0.001:FTTS.xmax;
            FTTS.pval = 2*cdf('F', tstat, nu1, nu2);
        end
        FTTS.tyl = pdf('F', FTTS.tintleft, nu1, nu2);
        FTTS.tlar = area(FTTS.tintleft, FTTS.tyl);
        FTTS.tlar.FaceColor = '#8a22b3';
        FTTS.tlar.FaceAlpha = 1;
        FTTS.tlar.EdgeColor = 'none';

        FTTS.tyr = pdf('F', FTTS.tintright, nu1, nu2);
        FTTS.trar = area(FTTS.tintright, FTTS.tyr);
        FTTS.trar.FaceColor = '#8a22b3';
        FTTS.trar.FaceAlpha = 1;
        FTTS.trar.EdgeColor = 'none';
    end
    FTTS.empdec = sprintf('%%.%df', 4);
    FTTS.pdec = sprintf('%%.%df', 4);
    FTTS.tp = sprintf(['Test statistic = ', FTTS.empdec,', p value = ', ...
        FTTS.pdec], tstat, FTTS.pval);
    subtitle({FTTS.variables, FTTS.tp}, 'Interpreter', 'tex');
end
```