Nome: Gina Szajnbok Harari

Tarefa 6 - Visualização de dados em Spider\_Plot

Fonte: Dados obtidos do DATASUS (2002), filtrando os dados de doentes internados e/ou atendidos na cidade analisada com problemas cardiovasculares (CID-I) ou com problemas respiratórios (CID-J), no ano de 2008.

MARQUES, J. Associações significativas entre dados de temperatura e de saúde em cidades litorâneas e não litorâneas. Tese apresentada ao Instituto de Energia e Ambiente, no Programa de Ciência Ambiental, Universidade de São Paulo para obtenção do título de Doutora em Ciências. São Paulo, 2019, 357 p.

Observações:

Descobri que todos os arquivos que estão no diretório onde roda o Matlab são considerados includes, mesmo sem declaração.

Achei que isso facilita bastante.

Faltou pesquisar como alterar a cor da legenda.

Laranja – cardiológicos

Azul - Pulmonares

Matlab

|  |
| --- |
| %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  % MESTRADO 2020  % UPM - Universidade Presbiteriana Mackenzie  % 72008075 - HARARI, G. S.  % GNU Octave(R) - Version 5.2.0  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  % Module : spiderPlot\_Tarefa\_6\_GSH\_Card\_Pulm\_2008\_Module\_v\_20200411.m  % Input : spiderPlot\_Tarefa\_6\_GSH\_Card\_Pulm\_2008\_Data\_v\_20200411.dat  % Output : spiderPlot\_Tarefa\_6\_GSH\_Card\_Pulm\_2008\_Fig\_v\_20200411.JPG  % Process: 12th, Apr 2020  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  %  clc; clear all; close all;  %  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  % Carga da Base de Dados SUS que já possui a media dos  % meses do ano de 2008, para internações card e pulm  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  %  load spiderPlot\_Tarefa\_6\_GSH\_Card\_Pulm\_2008\_Data\_v\_20200411.dat;  %  card\_pulm\_2008=spiderPlot\_Tarefa\_6\_GSH\_Card\_Pulm\_2008\_Data\_v\_20200411;  %  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  % Gráfico de SpiderPlot - radar  % Função Chamada a Função spider\_plot  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  %  spider\_plot(card\_pulm\_2008, ...  'AxesLabels', {'janeiro', 'fevereiro', 'março', 'abril', 'maio', 'junho', ...  'julho', 'agosto', 'setembro', 'outubro', 'novembro', 'dezembro'});  title(' Média de Internações no SUS a cada mês (2008) ',...  'FontSize', 14);  legend\_str = {'Problemas Cardiológicos', 'Problemas Pulmonares'};  legend(legend\_str, 'Location', 'southoutside');  print -djpg spiderPlot\_Tarefa\_6\_GSH\_Card\_Pulm\_012008\_Fig\_v\_20200411; |
| ## Copyright (C) 2020 mauro  ##  ## This program is free software: you can redistribute it and/or modify it  ## under the terms of the GNU General Public License as published by  ## the Free Software Foundation, either version 3 of the License, or  ## (at your option) any later version.  ##  ## This program is distributed in the hope that it will be useful, but  ## WITHOUT ANY WARRANTY; without even the implied warranty of  ## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  ## GNU General Public License for more details.  ##  ## You should have received a copy of the GNU General Public License  ## along with this program. If not, see  ## <https://www.gnu.org/licenses/>.  ## -\*- texinfo -\*-  ## @deftypefn {} {@var{retval} =} spider\_plot (@var{input1}, @var{input2})  ##  ## @seealso{}  ## @end deftypefn  ## Author: mauro <mauro@GOLDA-19610808>  ## Created: 2020-04-20  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  % Grafico de SpiderPlot - radar  % Função spider\_plot()  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  function spider\_plot(P, varargin)  %spider\_plot Create a spider or radar plot with individual axes.  %  % Syntax:  % spider\_plot(P)  % spider\_plot(P, Name, Value, ...)  %  % Input Arguments:  % (Required)  % P - The data points used to plot the spider chart. The  % rows are the groups of data and the columns are the  % data points. The axes labels and axes limits are  % automatically generated if not specified.  % [vector | matrix]  %  % Name-Value Pair Arguments:  % (Optional)  % AxesLabels - Used to specify the label each of the axes.  % [auto-generated (default) | cell of strings | 'none']  %  % AxesInterval - Used to change the number of intervals displayed  % between the webs.  % [3 (default) | integer]  %  % AxesPrecision - Used to change the precision level on the value  % displayed on the axes.  % [1 (default) | integer]  %  % AxesDisplay - Used to change the number of axes in which the  % axes text are displayed. 'None' or 'one' can be used  % to simplify the plot appearance for normalized data.  % ['all' (default) | 'none' | 'one']  %  % AxesLimits - Used to manually set the axes limits. A matrix of  % 2 x size(P, 2). The top row is the minimum axes  % limits and the bottow row is the maximum axes limits.  % [auto-scaled (default) | matrix]  %  % FillOption - Used to toggle color fill option.  % ['off' (default) | 'on']  %  % FillTransparency - Used to set color fill transparency.  % [0.1 (default) | scalar in range (0, 1)]  %  % Color - Used to specify the line color, specified as an RGB  % triplet. The intensities must be in the range (0, 1).  % [MATLAB colors (default) | RGB triplet]  %  % LineStyle - Used to change the line style of the plots.  % ['-' (default) | '--' | ':' | '-.' | 'none' | cell array of character vectors]  %  % LineWidth - Used to change the line width, where 1 point is  % 1/72 of an inch.  % [0.5 (default) | positive value | vector]  %  % Marker - Used to change the marker symbol of the plots.  % ['o' (default) | 'none' | '\*' | 's' | 'd' | ... | cell array of character vectors]  %  % MarkerSize - Used to change the marker size, where 1 point is  % 1/72 of an inch.  % [8 (default) | positive value | vector]  %  % AxesFontSize - Used to change the font size of the values  % displayed on the axes.  % [10 (default) | scalar value greater than zero]  %  % LabelFontSize - Used to change the font size of the labels.  % [10 (default) | scalar value greater than zero]  %  % Direction - Used to change the direction of rotation of the  % plotted data and axis labels.  % [clockwise (default) | counterclockwise]  %  % AxesLabelsOffset - Used to adjust the position offset of the axes  % labels.  % [0.1 (default) | positive value]  %  % AxesScaling - Used to change the scaling of the axes.  % ['linear' (default) | 'log']  %  % Examples:  % % Example 1: Minimal number of arguments. All non-specified, optional  % arguments are set to their default values. Axes labels  % and limits are automatically generated and set.  %  % D1 = [5 3 9 1 2];  % D2 = [5 8 7 2 9];  % D3 = [8 2 1 4 6];  % P = [D1; D2; D3];  % spider\_plot(P);  % legend('D1', 'D2', 'D3', 'Location', 'southoutside');  %  % % Example 2: Manually setting the axes limits. All non-specified,  % optional arguments are set to their default values.  %  % D1 = [5 3 9 1 2];  % D2 = [5 8 7 2 9];  % D3 = [8 2 1 4 6];  % P = [D1; D2; D3];  % spider\_plot(P,...  % 'AxesLimits', [1, 2, 1, 1, 1; 10, 8, 9, 5, 10]); % [min axes limits; max axes limits]  %  % % Example 3: Set fill option on. The fill transparency can be adjusted.  %  % D1 = [5 3 9 1 2];  % D2 = [5 8 7 2 9];  % D3 = [8 2 1 4 6];  % P = [D1; D2; D3];  % spider\_plot(P,...  % 'AxesLabels', {'S1', 'S2', 'S3', 'S4', 'S5'},...  % 'AxesInterval', 2,...  % 'FillOption', 'on',...  % 'FillTransparency', 0.1);  %  % % Example 4: Maximum number of arguments.  %  % D1 = [5 3 9 1 2];  % D2 = [5 8 7 2 9];  % D3 = [8 2 1 4 6];  % P = [D1; D2; D3];  % spider\_plot(P,...  % 'AxesLabels', {'S1', 'S2', 'S3', 'S4', 'S5'},...  % 'AxesInterval', 4,...  % 'AxesPrecision', 0,...  % 'AxesDisplay', 'one',...  % 'AxesLimits', [1, 2, 1, 1, 1; 10, 8, 9, 5, 10],...  % 'FillOption', 'on',...  % 'FillTransparency', 0.2,...  % 'Color', [1, 0, 0; 0, 1, 0; 0, 0, 1],...  % 'LineStyle', {'--', '-', '--'},...  % 'LineWidth', [1, 2, 3],...  % 'Marker', {'o', 'd', 's'},...  % 'MarkerSize', [8, 10, 12],...  % 'AxesFontSize', 12,...  % 'LabelFontSize', 10,...  % 'Direction', 'clockwise',...  % 'AxesLabelsOffset', 0,...  % 'AxesScaling', 'linear');  %  % % Example 5: Excel-like radar charts.  %  % D1 = [5 0 3 4 4];  % D2 = [2 1 5 5 4];  % P = [D1; D2];  % spider\_plot(P,...  % 'AxesInterval', 5,...  % 'AxesPrecision', 0,...  % 'AxesDisplay', 'one',...  % 'AxesLimits', [0, 0, 0, 0, 0; 5, 5, 5, 5, 5],...  % 'FillOption', 'on',...  % 'FillTransparency', 0.1,...  % 'Color', [139, 0, 0; 240, 128, 128]/255,...  % 'LineWidth', 4,...  % 'Marker', 'none',...  % 'AxesFontSize', 14,...  % 'LabelFontSize', 10);  % title('Excel-like Radar Chart',...  % 'FontSize', 14);  % legend\_str = {'D1', 'D2'};  % legend(legend\_str, 'Location', 'southoutside');  %  % % Example 6: Logarithimic scale on all axes. Axes limits and axes  % intervals are automatically set to factors of 10.  %  % D1 = [-1 10 1 500];  % D2 = [-10 20 1000 60];  % D3 = [-100 30 10 7];  % P = [D1; D2; D3];  % spider\_plot(P,...  % 'AxesPrecision', 2,...  % 'AxesDisplay', 'one',...  % 'AxesScaling', 'log');  % legend('D1', 'D2', 'D3', 'Location', 'northeast');  %  % % Example 7: Spider plot with subplot feature.  %  % D1 = [5 3 9 1 2];  % D2 = [5 8 7 2 9];  % D3 = [8 2 1 4 6];  % P = [D1; D2; D3];  % subplot(1, 2, 1)  % spider\_plot(P,...  % 'AxesInterval', 1,...  % 'AxesPrecision', 0);  % subplot(1, 2, 2)  % spider\_plot(P,...  % 'AxesInterval', 1,...  % 'AxesPrecision', 0);  %  % Author:  % Moses Yoo, (jyoo at hatci dot com)  % 2020-03-26: Added feature to allow different line styles, line width,  % marker type, and marker sizes for the data groups.  % 2020-02-12: Fixed condition and added error checking for when only one  % data group is plotted.  % 2020-01-27: Corrected bug where only 7 entries were allowed in legend.  % 2020-01-06: Added support for subplot feature.  % 2019-11-27: Add option to change axes to logarithmic scale.  % 2019-11-15: Add feature to customize the plot rotational direction and  % the offset position of the axis labels.  % 2019-10-23: Minor revision to set starting axes as the vertical line.  % Add customization option for font sizes and axes display.  % 2019-10-16: Minor revision to add name-value pairs for customizing  % color, marker, and line settings.  % 2019-10-08: Another major revision to convert to name-value pairs and  % add color fill option.  % 2019-09-17: Major revision to improve speed, clarity, and functionality  %  % Special Thanks:  % Special thanks to Gabriela Andrade, Andrés Garcia, Alex Grenyer,  % Tobias Kern, Zafar Ali, Christophe Hurlin, & Roman for their feature  % recommendations and suggested bug fixes.  %%% Data Properties %%%  % Point properties  [num\_data\_groups, num\_data\_points] = size(P);  % Number of optional arguments  numvarargs = length(varargin);  % Check for even number of name-value pair argments  if mod(numvarargs, 2) == 1  error('Error: Please check name-value pair arguments');  end #if mod(numvarargs, 2) == 1  % Create default labels  axes\_labels = cell(1, num\_data\_points);  % Iterate through number of data points  for ii = 1:num\_data\_points  % Default axes labels  axes\_labels{ii} = sprintf('Label %i', ii);  end # for ii = 1:num\_data\_points  % Default arguments  axes\_interval = 3;  axes\_precision = 1;  axes\_display = 'all';  axes\_limits = [];  fill\_option = 'off';  fill\_transparency = 0.2;  colors = lines(num\_data\_groups);  line\_style = '-';  line\_width = 2;  marker\_type = 'o';  marker\_size = 8;  axes\_font\_size = 10;  label\_font\_size = 10;  direction = 'clockwise';  axes\_labels\_offset = 0.1;  axes\_scaling = 'linear';  % Check if optional arguments were specified  if numvarargs > 1  % Initialze name-value arguments  name\_arguments = varargin(1:2:end);  value\_arguments = varargin(2:2:end);    % Iterate through name-value arguments  for ii = 1:length(name\_arguments)  % Set value arguments depending on name  switch lower(name\_arguments{ii})  case 'axeslabels'  axes\_labels = value\_arguments{ii};  case 'axesinterval'  axes\_interval = value\_arguments{ii};  case 'axesprecision'  axes\_precision = value\_arguments{ii};  case 'axesdisplay'  axes\_display = value\_arguments{ii};  case 'axeslimits'  axes\_limits = value\_arguments{ii};  case 'filloption'  fill\_option = value\_arguments{ii};  case 'filltransparency'  fill\_transparency = value\_arguments{ii};  case 'color'  colors = value\_arguments{ii};  case 'linestyle'  line\_style = value\_arguments{ii};  case 'linewidth'  line\_width = value\_arguments{ii};  case 'marker'  marker\_type = value\_arguments{ii};  case 'markersize'  marker\_size = value\_arguments{ii};  case 'axesfontsize'  axes\_font\_size = value\_arguments{ii};  case 'labelfontsize'  label\_font\_size = value\_arguments{ii};  case 'direction'  direction = value\_arguments{ii};  case 'axeslabelsoffset'  axes\_labels\_offset = value\_arguments{ii};  case 'axesscaling'  axes\_scaling = value\_arguments{ii};  otherwise  error('Error: Please enter in a valid name-value pair.');  end #switch lower(name\_arguments{ii})  end #for ii = 1:length(name\_arguments)    end #if numvarargs > 1  %%% Error Check %%%  % Check if axes labels is a cell  if iscell(axes\_labels)  % Check if the axes labels are the same number as the number of points  if length(axes\_labels) ~= num\_data\_points  error('Error: Please make sure the number of labels is the same as the number of points.');  end #if length(axes\_labels) ~= num\_data\_points  else  % Check if valid string entry  if ~contains(axes\_labels, 'none')  error('Error: Please enter in valid labels or "none" to remove labels.');  end #if ~contains(axes\_labels, 'none')  end #if iscell(axes\_labels)  % Check if axes limits is not empty  if ~isempty(axes\_limits)  % Check if the axes limits same length as the number of points  if size(axes\_limits, 1) ~= 2 || size(axes\_limits, 2) ~= num\_data\_points  error('Error: Please make sure the min and max axes limits match the number of data points.');  end #if size(axes\_limits, 1) ~= 2 || size(axes\_limits, 2) ~= num\_data\_points    % Lower and upper limits  lower\_limits = axes\_limits(1, :);  upper\_limits = axes\_limits(2, :);    % Difference in upper and lower limits  diff\_limits = upper\_limits - lower\_limits;    % Check to make sure upper limit is greater than lower limit  if any(diff\_limits < 0)  error('Error: Please make sure max axes limits are greater than the min axes limits.');  end #if any(diff\_limits < 0)    % Check the range of axes limits  if any(diff\_limits == 0)  error('Error: Please make sure the min and max axes limits are different.');  end #if any(diff\_limits == 0)  end #if ~isempty(axes\_limits)  % Check if axes properties are an integer  if floor(axes\_interval) ~= axes\_interval || floor(axes\_precision) ~= axes\_precision  error('Error: Please enter in an integer for the axes properties.');  end #if floor(axes\_interval) ~= axes\_interval || floor(axes\_precision) ~= axes\_precision  % Check if axes properties are positive  if axes\_interval < 1 || axes\_precision < 0  error('Error: Please enter a positive for the axes properties.');  end #if axes\_interval < 1 || axes\_precision < 0  % Check if axes display is valid string entry  if ~ismember(axes\_display, {'all', 'none', 'one'})  error('Error: Invalid axes display entry. Please enter in "all", "none", or "one" to set axes text.');  end #if ~ismember(axes\_display, {'all', 'none', 'one'})  % Check if not a valid fill option arguement  if ~ismember(fill\_option, {'off', 'on'})  error('Error: Please enter either "off" or "on" for fill option.');  end #if ~ismember(fill\_option, {'off', 'on'})  % Check if fill transparency is valid  if fill\_transparency < 0 || fill\_transparency > 1  error('Error: Please enter a transparency value between [0, 1].');  end #if fill\_transparency < 0 || fill\_transparency > 1  % Check if font size is greater than zero  if axes\_font\_size <= 0 || label\_font\_size <= 0  error('Error: Please enter a font size greater than zero.');  end #if axes\_font\_size <= 0 || label\_font\_size <= 0  % Check if direction is valid string entry  if ~ismember(direction, {'counterclockwise', 'clockwise'})  error('Error: Invalid direction entry. Please enter in "counterclockwise" or "clockwise" to set direction of rotation.');  end #if ~ismember(direction, {'counterclockwise', 'clockwise'})  % Check if axes labels offset is positive  if axes\_labels\_offset < 0  error('Error: Please enter a positive for the axes labels offset.');  end # if axes\_labels\_offset < 0  % Check if axes scaling is valid  if ~ismember(axes\_scaling, {'linear', 'log'})  error('Error: Invalid axes scaling entry. Please enter in "linear" or "log" to set axes scaling.');  end #if ~ismember(axes\_scaling, {'linear', 'log'})  % Check axis limits if num\_data\_groups is one  if num\_data\_groups == 1 && isempty(axes\_limits)  error('Error: For one data group, please enter in a range for the axes limits.');  end #if num\_data\_groups == 1 && isempty(axes\_limits)  % Check if line style is a char  if ischar(line\_style)  % Convert to cell array of char  line\_style = cellstr(line\_style);    % Repeat cell to number of data groups  line\_style = repmat(line\_style, num\_data\_groups, 1);  elseif iscellstr(line\_style)  % Check is length is one  if length(line\_style) == 1  % Repeat cell to number of data groups  line\_style = repmat(line\_style, num\_data\_groups, 1);  elseif length(line\_style) ~= num\_data\_groups  error('Error: Please specify the same number of line styles as number of data groups.');  end #if length(line\_style) == 1  else  error('Error: Please make sure the line style is a char or a cell array of char.');  end #if ischar(line\_style)  % Check if line width is numeric  if isnumeric(line\_width)  % Check is length is one  if length(line\_width) == 1  % Repeat array to number of data groups  line\_width = repmat(line\_width, num\_data\_groups, 1);  elseif length(line\_width) ~= num\_data\_groups  error('Error: Please specify the same number of line width as number of data groups.');  end #if length(line\_width) == 1  else  error('Error: Please make sure the line width is a numeric value.');  end #if isnumeric(line\_width)  % Check if marker type is a char  if ischar(marker\_type)  % Convert to cell array of char  marker\_type = cellstr(marker\_type);    % Repeat cell to number of data groups  marker\_type = repmat(marker\_type, num\_data\_groups, 1);  elseif iscellstr(marker\_type)  % Check is length is one  if length(marker\_type) == 1  % Repeat cell to number of data groups  marker\_type = repmat(marker\_type, num\_data\_groups, 1);  elseif length(marker\_type) ~= num\_data\_groups  error('Error: Please specify the same number of line styles as number of data groups.');  end #if length(marker\_type) == 1  else  error('Error: Please make sure the line style is a char or a cell array of char.');  end #if ischar(marker\_type)  % Check if line width is numeric  if isnumeric(marker\_size)  if length(marker\_size) == 1  % Repeat array to number of data groups  marker\_size = repmat(marker\_size, num\_data\_groups, 1);  elseif length(marker\_size) ~= num\_data\_groups  error('Error: Please specify the same number of line width as number of data groups.');  end #if length(marker\_size) == 1  else  error('Error: Please make sure the line width is numeric.');  end #if isnumeric(marker\_size)  %%% Axes Scaling Properties %%%  % Check axes scaling option  if strcmp(axes\_scaling, 'log')  % Logarithm of base 10, account for numbers less than 1  P = sign(P) .\* log10(abs(P));    % Minimum and maximun log limits  min\_limit = min(min(fix(P)));  max\_limit = max(max(floor(P)));    % Update axes interval  axes\_interval = max\_limit - min\_limit;    % Update axes limits  axes\_limits = zeros(2, num\_data\_points);  axes\_limits(1, :) = min\_limit;  axes\_limits(2, :) = max\_limit;  end #if strcmp(axes\_scaling, 'log')  %%% Figure Properties %%%  % Grab current figure  fig = gcf;  % Set figure background  %fig.Color = 'white';  % Reset axes  cla reset;  % Current axes handle  ax = gca;  % Axis limits  hold on;  axis square;  axis([-1, 1, -1, 1] \* 1.3);  % Axis properties  %ax.XTickLabel = [];  %ax.YTickLabel = [];  %ax.XColor = 'none';  %ax.YColor = 'none';  % Plot colors  grey = [0.5, 0.5, 0.5];  % Polar increments  theta\_increment = 2\*pi/num\_data\_points;  rho\_increment = 1/(axes\_interval+1);  %%% Scale Data %%%  % Pre-allocation  P\_scaled = zeros(size(P));  axes\_range = zeros(3, num\_data\_points);  % Iterate through number of data points  for ii = 1:num\_data\_points  % Group of points  group\_points = P(:, ii);    % Automatically the range of each group  min\_value = min(group\_points);  max\_value = max(group\_points);  range = max\_value - min\_value;    % Check if axes\_limits is empty  if isempty(axes\_limits)  % Scale points to range from [rho\_increment, 1]  P\_scaled(:, ii) = ((group\_points - min\_value) / range) \* (1 - rho\_increment) + rho\_increment;  else  % Manually set the range of each group  min\_value = axes\_limits(1, ii);  max\_value = axes\_limits(2, ii);  range = max\_value - min\_value;    % Check if the axes limits are within range of points  if min\_value > min(group\_points) || max\_value < max(group\_points)  error('Error: Please make the manually specified axes limits are within range of the data points.');  end #if min\_value > min(group\_points) || max\_value < max(group\_points)    % Scale points to range from [rho\_increment, 1]  P\_scaled(:, ii) = ((group\_points - min\_value) / range) \* (1 - rho\_increment) + rho\_increment;  end #if isempty(axes\_limits)    % Store to array  axes\_range(:, ii) = [min\_value; max\_value; range];  end #for ii = 1:num\_data\_points  %%% Polar Axes %%%  % Polar coordinates  rho = 0:rho\_increment:1;  % Check rotational direction  switch direction  case 'counterclockwise'  % Shift by pi/2 to set starting axis the vertical line  theta = (0:theta\_increment:2\*pi) + (pi/2);  case 'clockwise'  % Shift by pi/2 to set starting axis the vertical line  theta = (0:-theta\_increment:-2\*pi) + (pi/2);  end #switch direction  % Remainder after using a modulus of 2\*pi  theta = mod(theta, 2\*pi);  % Iterate through each theta  for ii = 1:length(theta)-1  % Convert polar to cartesian coordinates  [x\_axes, y\_axes] = pol2cart(theta(ii), rho);    % Plot webs  h = plot(x\_axes, y\_axes,...  'LineWidth', 1.5,...  'Color', grey);    % Turn off legend annotation  % h.Annotation.LegendInformation.IconDisplayStyle = 'off';  end #for ii = 1:length(theta)-1  % Iterate through each rho  for ii = 2:length(rho)  % Convert polar to cartesian coordinates  [x\_axes, y\_axes] = pol2cart(theta, rho(ii));    % Plot axes  h = plot(x\_axes, y\_axes,...  'Color', grey);    % Turn off legend annotation  % h.Annotation.LegendInformation.IconDisplayStyle = 'off';  end #for ii = 2:length(rho)  % Set end index depending on axes display argument  switch axes\_display  case 'all'  theta\_end\_index = length(theta)-1;  case 'one'  theta\_end\_index = 1;  case 'none'  theta\_end\_index = 0;  end #switch axes\_display  % Iterate through each theta  for ii = 1:theta\_end\_index  % Convert polar to cartesian coordinates  [x\_axes, y\_axes] = pol2cart(theta(ii), rho);    % Iterate through points on isocurve  for jj = 2:length(rho)  % Axes increment value  min\_value = axes\_range(1, ii);  range = axes\_range(3, ii);  axes\_value = min\_value + (range/axes\_interval) \* (jj-2);    % Check axes scaling option  if strcmp(axes\_scaling, 'log')  % Exponent to the tenth power  axes\_value = 10^axes\_value;  end #if strcmp(axes\_scaling, 'log')    % Display axes text  text\_str = sprintf(sprintf('%%.%if', axes\_precision), axes\_value);  text(x\_axes(jj), y\_axes(jj), text\_str,...  'Units', 'Data',...  'Color', 'k',...  'FontSize', axes\_font\_size,...  'HorizontalAlignment', 'center',...  'VerticalAlignment', 'middle');  end #for jj = 2:length(rho)  end #for ii = 1:theta\_end\_index  %%% Plot %%%  % Iterate through number of data groups  for ii = 1:num\_data\_groups  % Convert polar to cartesian coordinates  [x\_points, y\_points] = pol2cart(theta(1:end-1), P\_scaled(ii, :));    % Make points circular  x\_circular = [x\_points, x\_points(1)];  y\_circular = [y\_points, y\_points(1)];    % Plot data points  plot(x\_circular, y\_circular,...  'LineStyle', line\_style{ii},...  'Marker', marker\_type{ii},...  'Color', colors(ii, :),...  'LineWidth', line\_width(ii),...  'MarkerSize', marker\_size(ii),...  'MarkerFaceColor', colors(ii, :));    % Check if fill option is toggled on  if strcmp(fill\_option, 'on')  % Fill area within polygon  h = patch(x\_circular, y\_circular, colors(ii, :),...  'EdgeColor', 'none',...  'FaceAlpha', fill\_transparency);    % Turn off legend annotation  h.Annotation.LegendInformation.IconDisplayStyle = 'off';  end #if strcmp(fill\_option, 'on')  end #for ii = 1:num\_data\_groups  % Find object handles  %text\_handles = findobj(ax.Children,...  % 'Type', 'Text');  %patch\_handles = findobj(ax.Children,...  % 'Type', 'Patch');  %isocurve\_handles = findobj(ax.Children,...  % 'Color', grey,...  % '-and', 'Type', 'Line');  %plot\_handles = findobj(ax.Children, '-not',...  % 'Color', grey,...  % '-and', 'Type', 'Line');  % Manually set the stack order  %uistack(text\_handles, 'bottom');  %uistack(plot\_handles, 'bottom');  %uistack(patch\_handles, 'bottom');  %uistack(isocurve\_handles, 'bottom');  %%% Labels %%%  % Check labels argument  if ~strcmp(axes\_labels, 'none')  % Convert polar to cartesian coordinates  [x\_axes, y\_axes] = pol2cart(theta, rho(end));    % Iterate through number of data points  for ii = 1:length(axes\_labels)  % Angle of point in radians  theta\_point = theta(ii);    % Find out which quadrant the point is in  if theta\_point == 0  quadrant = 0;  elseif theta\_point == pi/2  quadrant = 1.5;  elseif theta\_point == pi  quadrant = 2.5;  elseif theta\_point == 3\*pi/2  quadrant = 3.5;  elseif theta\_point == 2\*pi  quadrant = 0;  elseif theta\_point > 0 && theta\_point < pi/2  quadrant = 1;  elseif theta\_point > pi/2 && theta\_point < pi  quadrant = 2;  elseif theta\_point > pi && theta\_point < 3\*pi/2  quadrant = 3;  elseif theta\_point > 3\*pi/2 && theta\_point < 2\*pi  quadrant = 4;  end #if theta\_point == 0    % Adjust label alignment depending on quadrant  switch quadrant  case 0  horz\_align = 'left';  vert\_align = 'middle';  x\_pos = axes\_labels\_offset;  y\_pos = 0;  case 1  horz\_align = 'left';  vert\_align = 'bottom';  x\_pos = axes\_labels\_offset;  y\_pos = axes\_labels\_offset;  case 1.5  horz\_align = 'center';  vert\_align = 'bottom';  x\_pos = 0;  y\_pos = axes\_labels\_offset;  case 2  horz\_align = 'right';  vert\_align = 'bottom';  x\_pos = -axes\_labels\_offset;  y\_pos = axes\_labels\_offset;  case 2.5  horz\_align = 'right';  vert\_align = 'middle';  x\_pos = -axes\_labels\_offset;  y\_pos = 0;  case 3  horz\_align = 'right';  vert\_align = 'top';  x\_pos = -axes\_labels\_offset;  y\_pos = -axes\_labels\_offset;  case 3.5  horz\_align = 'center';  vert\_align = 'top';  x\_pos = 0;  y\_pos = -axes\_labels\_offset;  case 4  horz\_align = 'left';  vert\_align = 'top';  x\_pos = axes\_labels\_offset;  y\_pos = -axes\_labels\_offset;  end #switch quadrant    % Display text label  text(x\_axes(ii)+x\_pos, y\_axes(ii)+y\_pos, axes\_labels{ii},...  'Units', 'Data',...  'HorizontalAlignment', horz\_align,...  'VerticalAlignment', vert\_align,...  'EdgeColor', 'k',...  'BackgroundColor', 'w',...  'FontSize', label\_font\_size);  end #for ii = 1:length(axes\_labels)  end #if ~strcmp(axes\_labels, 'none')  endfunction #function spider\_plot (P, varargin) |

Mapa colorido com texto preto sobre fundo branco

Descrição gerada automaticamente