Dear Mr. Markus,

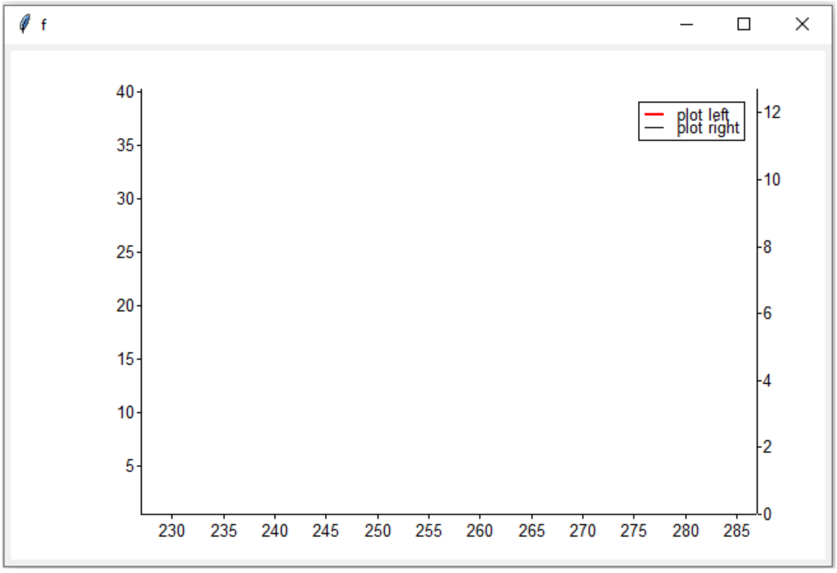
I observed a bug (which is not limited to the current master version on GitHub) if a widget contains two plots, whereby one plot is orientated on the right y-axis by using "*Plotchart::createRightAxis*".
When I draw the legend, only the left y-axis orientated plot is displayed in right colour and line width.
The right y-axis plot is always black coloured and with line width equal to one.
So I analysed the behaviour of the code.
The plotchart code uses different variables to handle the configuration and data.
As consequence, there is the variable *data_series* which uses different widgets to handle left and right y-axis orientated plots and there is the variable legend, which combines all plots into one widget.

Widgets names for right y-axis orientated plots have a leading letter "r".
As soon as plotchart is requested to draw the legend, the method "*ActuallyDrawLegend*" is accessing the
the variable *data_series*, which contains the configuration information for each plot.
Unfortunately, the aspect for left and right y-axis is not correctly implemented in this method.
The leading letter "r" is filtered, both in method "*DrawLegend*" and "*ActuallyDrawLegend*". So the access to *data_series* in the loop over the series comes to nothing for right y-axis orientated plots.

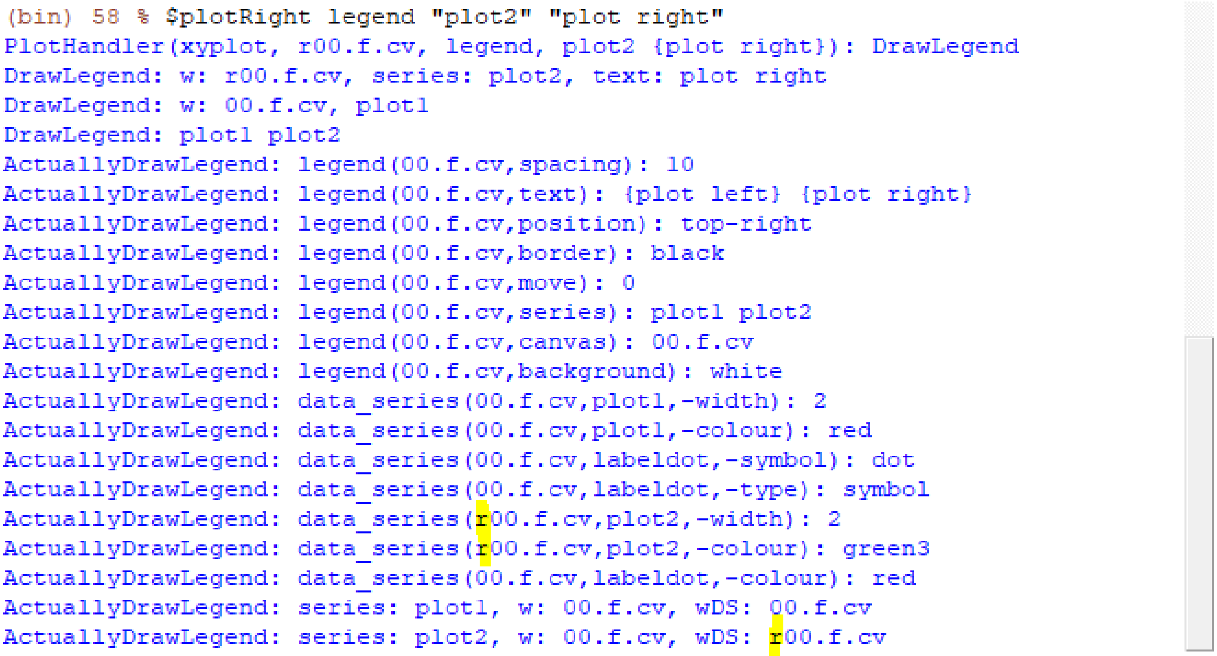
Sequence to force the failure:

```
package require Plotchart
set wnd [toplevel .f]
canvas $wnd.cv -width 640 -height 400 -background white
grid $wnd.cv -sticky news -padx 3 -pady 3
set plotLeft [Plotchart::createXYPlot $wnd.cv {227 287 5} {0.5 40.2 5}]
$plotLeft dataconfig "plot1" -colour "red" -width 2
$plotLeft legend "plot1" "plot left"
set plotRight [Plotchart::createRightAxis $plotLeft {0 12.7 2}]
$plotRight dataconfig "plot2" -colour "green3" -width 2
$plotRight legend "plot2" "plot right"
```

If this sequence is executed, you will get the following diagram.

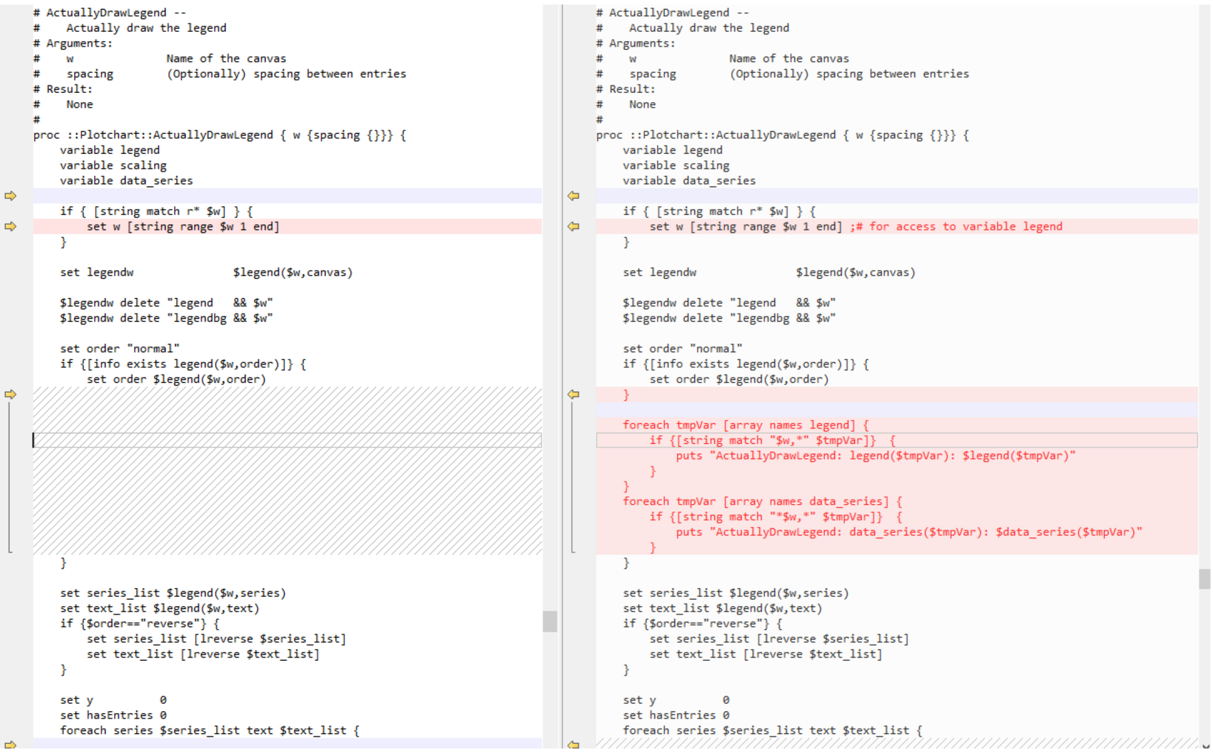


I added some own code to get outputs of the variable data.



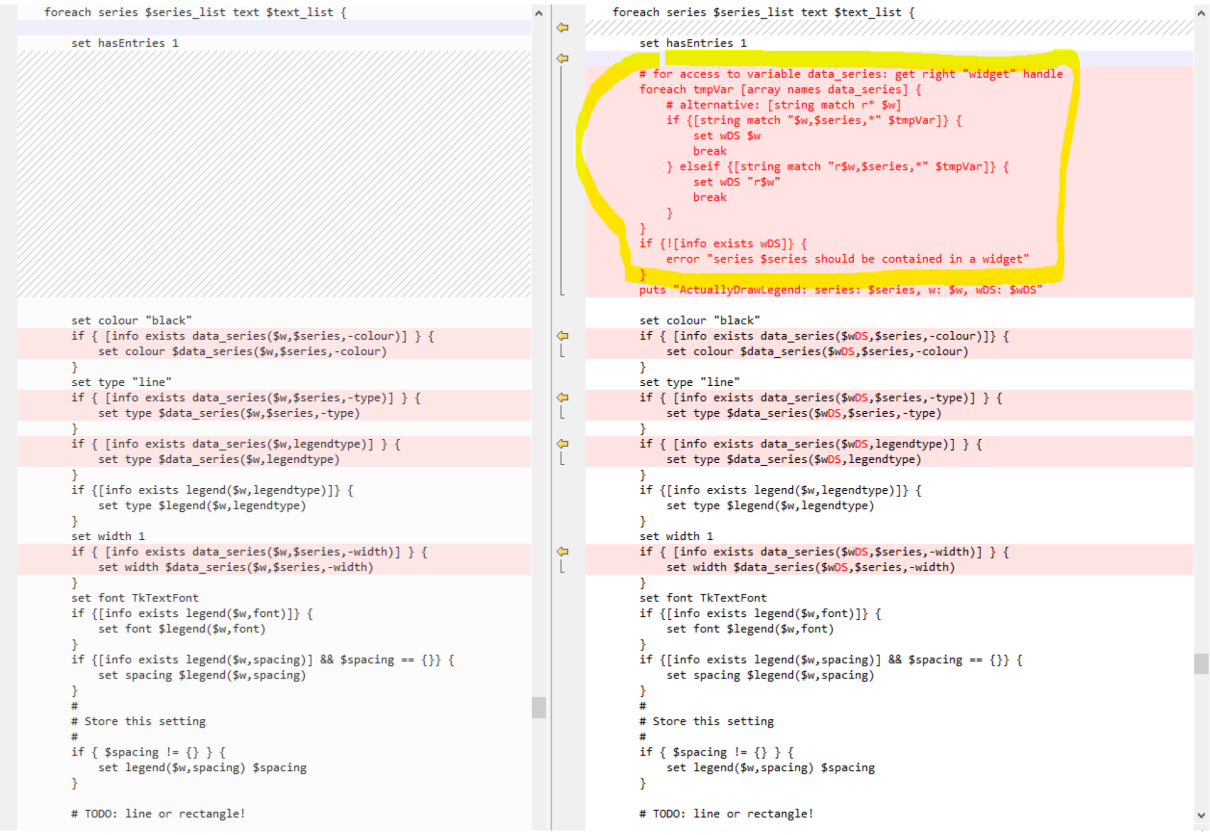
```
(bin) 58 % $plotRight legend "plot2" "plot right"
PlotHandler(xyplot, r00.f.cv, legend, plot2 {plot right}): DrawLegend
DrawLegend: w: r00.f.cv, series: plot2, text: plot right
DrawLegend: w: 00.f.cv, plot1
DrawLegend: plot1 plot2
ActuallyDrawLegend: legend(00.f.cv,spacing): 10
ActuallyDrawLegend: legend(00.f.cv,text): {plot left} {plot right}
ActuallyDrawLegend: legend(00.f.cv,position): top-right
ActuallyDrawLegend: legend(00.f.cv,border): black
ActuallyDrawLegend: legend(00.f.cv,move): 0
ActuallyDrawLegend: legend(00.f.cv,series): plot1 plot2
ActuallyDrawLegend: legend(00.f.cv,canvas): 00.f.cv
ActuallyDrawLegend: legend(00.f.cv,background): white
ActuallyDrawLegend: data_series(00.f.cv,plot1,-width): 2
ActuallyDrawLegend: data_series(00.f.cv,plot1,-colour): red
ActuallyDrawLegend: data_series(00.f.cv,labeldot,-symbol): dot
ActuallyDrawLegend: data_series(00.f.cv,labeldot,-type): symbol
ActuallyDrawLegend: data_series(r00.f.cv,plot2,-width): 2
ActuallyDrawLegend: data_series(r00.f.cv,plot2,-colour): green3
ActuallyDrawLegend: data_series(00.f.cv,labeldot,-colour): red
ActuallyDrawLegend: series: plot1, w: 00.f.cv, wDS: 00.f.cv
ActuallyDrawLegend: series: plot2, w: 00.f.cv, wDS: r00.f.cv
```

The following changes have been made. The yellow marked sequence adds the leading letter "r", if it is a right y-axis orientated plot for the access to the variable *data_series*.



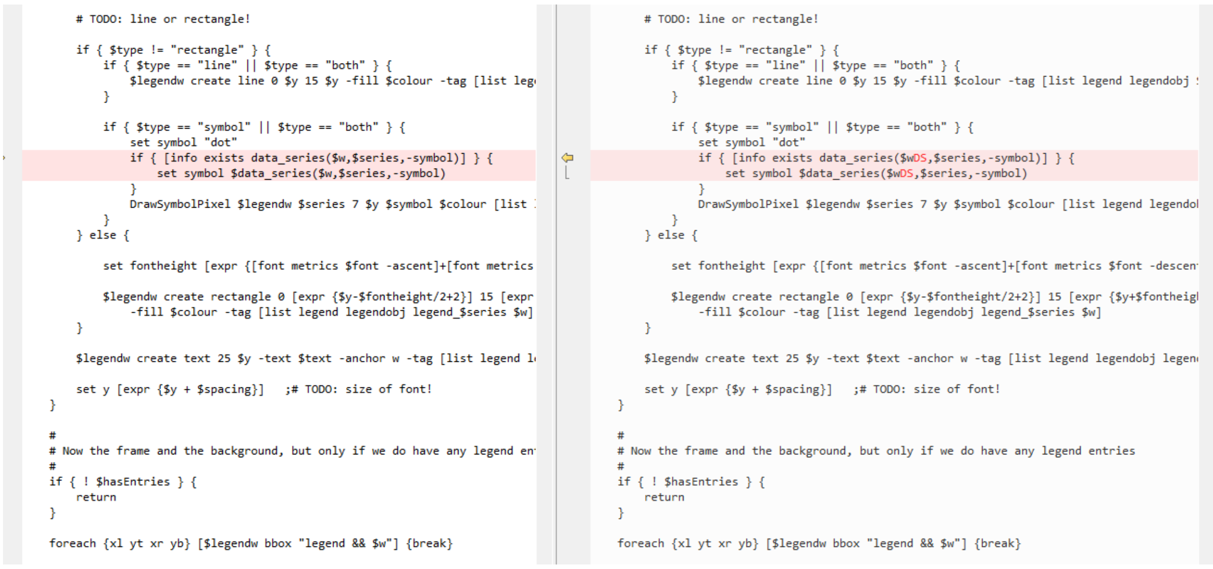
```
# ActuallyDrawLegend --
#    Actually draw the legend
# Arguments:
#    w               Name of the canvas
#    spacing         (Optionally) spacing between entries
# Result:
#    None
#
proc ::Plotchart::ActuallyDrawLegend { w {spacing {}}} {
    variable legend
    variable scaling
    variable data_series

    if { [string match r* $w] } {
        set w [string range $w 1 end]
    }

    set legendw              $legend($w,canvas)

    $legendw delete "legend   && $w"
    $legendw delete "legendbg && $w"

    set order "normal"
    if {[info exists legend($w,order)]} {
        set order $legend($w,order)
    }

    set series_list $legend($w,series)
    set text_list $legend($w,text)
    if {$order=="reverse"} {
        set series_list [lreverse $series_list]
        set text_list [lreverse $text_list]
    }

    set y         0
    set hasEntries 0
    foreach series $series_list text $text_list {
```

```
# ActuallyDrawLegend --
#    Actually draw the legend
# Arguments:
#    w               Name of the canvas
#    spacing         (Optionally) spacing between entries
# Result:
#    None
#
proc ::Plotchart::ActuallyDrawLegend { w {spacing {}}} {
    variable legend
    variable scaling
    variable data_series

    if { [string match r* $w] } {
        set w [string range $w 1 end] ;# for access to variable legend
    }

    set legendw              $legend($w,canvas)

    $legendw delete "legend   && $w"
    $legendw delete "legendbg && $w"

    set order "normal"
    if {[info exists legend($w,order)]} {
        set order $legend($w,order)
    }

    foreach tmpVar [array names legend] {
        if {[string match "$w,*" $tmpVar]}  {
            puts "ActuallyDrawLegend: legend($tmpVar): $legend($tmpVar)"
        }
    }
    foreach tmpVar [array names data_series] {
        if {[string match "*$w,*" $tmpVar]}  {
            puts "ActuallyDrawLegend: data_series($tmpVar): $data_series($tmpVar)"
        }
    }

    set series_list $legend($w,series)
    set text_list $legend($w,text)
    if {$order=="reverse"} {
        set series_list [lreverse $series_list]
        set text_list [lreverse $text_list]
    }

    set y         0
    set hasEntries 0
    foreach series $series_list text $text_list {
```



```
    foreach series $series_list text $text_list {

        set hasEntries 1

        set colour "black"
        if { [info exists data_series($w,$series,-colour)] } {
            set colour $data_series($w,$series,-colour)
        }
        set type "line"
        if { [info exists data_series($w,$series,-type)] } {
            set type $data_series($w,$series,-type)
        }
        if { [info exists data_series($w,legendtype)] } {
            set type $data_series($w,legendtype)
        }
        if {[info exists legend($w,legendtype)]} {
            set type $legend($w,legendtype)
        }
        set width 1
        if { [info exists data_series($w,$series,-width)] } {
            set width $data_series($w,$series,-width)
        }
        set font TkTextFont
        if {[info exists legend($w,font)]} {
            set font $legend($w,font)
        }
        if {[info exists legend($w,spacing)] && $spacing == {}} {
            set spacing $legend($w,spacing)
        }
        #
        # Store this setting
        #
        if { $spacing != {} } {
            set legend($w,spacing) $spacing
        }

        # TODO: line or rectangle!
```

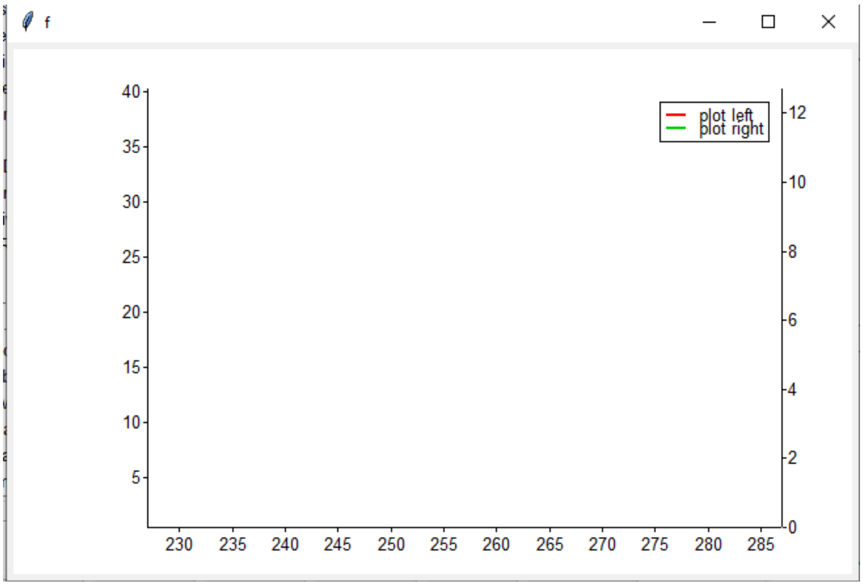
```
    foreach series $series_list text $text_list {

        set hasEntries 1

        # for access to variable data_series: get right "widget" handle
        foreach tmpVar [array names data_series] {
            # alternative: [string match r* $w]
            if {[string match "$w,$series,*" $tmpVar]} {
                set wDS $w
                break
            } elseif {[string match "r$w,$series,*" $tmpVar]} {
                set wDS "r$w"
                break
            }
        }
        if {![info exists wDS]} {
            error "series $series should be contained in a widget"
        }
        puts "ActuallyDrawLegend: series: $series, w: $w, wDS: $wDS"

        set colour "black"
        if { [info exists data_series($wDS,$series,-colour)]} {
            set colour $data_series($wDS,$series,-colour)
        }
        set type "line"
        if { [info exists data_series($wDS,$series,-type)] } {
            set type $data_series($wDS,$series,-type)
        }
        if { [info exists data_series($wDS,legendtype)] } {
            set type $data_series($wDS,legendtype)
        }
        if {[info exists legend($w,legendtype)]} {
            set type $legend($w,legendtype)
        }
        set width 1
        if { [info exists data_series($wDS,$series,-width)] } {
            set width $data_series($wDS,$series,-width)
        }
        set font TkTextFont
        if {[info exists legend($w,font)]} {
            set font $legend($w,font)
        }
        if {[info exists legend($w,spacing)] && $spacing == {}} {
            set spacing $legend($w,spacing)
        }
        #
        # Store this setting
        #
        if { $spacing != {} } {
            set legend($w,spacing) $spacing
        }

        # TODO: line or rectangle!
```



```
        # TODO: line or rectangle!

        if { $type != "rectangle" } {
            if { $type == "line" || $type == "both" } {
                $legendw create line 0 $y 15 $y -fill $colour -tag [list leg
            }

            if { $type == "symbol" || $type == "both" } {
                set symbol "dot"
                if { [info exists data_series($w,$series,-symbol)] } {
                    set symbol $data_series($w,$series,-symbol)
                }
                DrawSymbolPixel $legendw $series 7 $y $symbol $colour [list
            }
        } else {

            set fontheight [expr {[font metrics $font -ascent]+[font metrics

            $legendw create rectangle 0 [expr {$y-$fontheight/2+2}] 15 [expr
                -fill $colour -tag [list legend legendobj legend_$series $w]
        }

        $legendw create text 25 $y -text $text -anchor w -tag [list legend l

        set y [expr {$y + $spacing}]   ;# TODO: size of font!
    }

    #
    # Now the frame and the background, but only if we do have any legend en
    #
    if { ! $hasEntries } {
        return
    }

    foreach {xl yt xr yb} [$legendw bbox "legend && $w"] {break}
```

```
        # TODO: line or rectangle!

        if { $type != "rectangle" } {
            if { $type == "line" || $type == "both" } {
                $legendw create line 0 $y 15 $y -fill $colour -tag [list legend legendobj
            }

            if { $type == "symbol" || $type == "both" } {
                set symbol "dot"
                if { [info exists data_series($wDS,$series,-symbol)] } {
                    set symbol $data_series($wDS,$series,-symbol)
                }
                DrawSymbolPixel $legendw $series 7 $y $symbol $colour [list legend legendo
            }
        } else {

            set fontheight [expr {[font metrics $font -ascent]+[font metrics $font -descen

            $legendw create rectangle 0 [expr {$y-$fontheight/2+2}] 15 [expr {$y+$fontheig
                -fill $colour -tag [list legend legendobj legend_$series $w]
        }

        $legendw create text 25 $y -text $text -anchor w -tag [list legend legendobj legen

        set y [expr {$y + $spacing}]   ;# TODO: size of font!
    }

    #
    # Now the frame and the background, but only if we do have any legend entries
    #
    if { ! $hasEntries } {
        return
    }

    foreach {xl yt xr yb} [$legendw bbox "legend && $w"] {break}
```

The following diagram is plotted after code changes.



I hope you could follow my problem description.

Best regards,

Markus Freiberg