White Axes, line labels and transparency in Survival Curves

Calum Polwart

Background

I don't know if I'm actually going to use the results of this yet, but having spent more time than I'd hoped to trying to figure out how to do this, I thought I should document it. I'm producing a survival curve using ggsurvplot, but I want to present them on a dark background. That means I want to make my plot background transparent and make all the black stuff (lines, text etc) white so that it has good contrast on the dark background. I've almost never seen examples of this in the wild - either because in design terms someone says its a bad thing, /or/ its hard to do in R so people don't try?

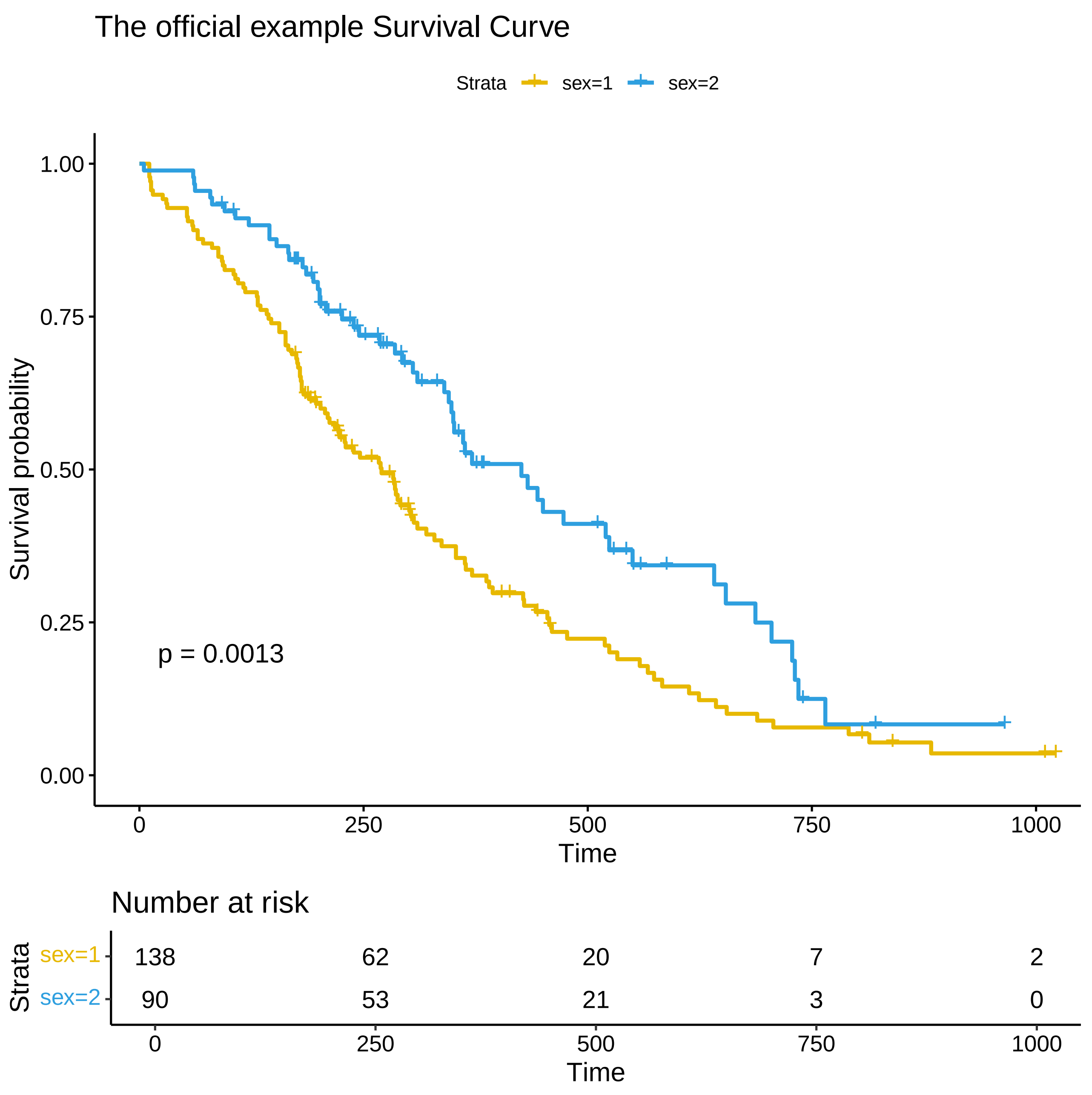
I've read enough data science blogs to know that having to move your eyes from a line to a legend to work out which one is which is also considered bad, and so I'm also going to label my lines.

Lets set up a basic plot

My plots have risk tables on them which makes life all the more 'exciting'. We will start with the official example plot - survival in the lung cancer data-set. I've made some tweaks to the standard off the shelf plot to get us started.

```r
require(survival)
require(survminer)
fit<- survfit(Surv(time, status) ~ sex, data = lung)

# Basic survival curves
ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",   ①
    palette = c("#E7B800", "#2E9FDF"),               ②
    pval = TRUE,                                     ③
    risk.table = TRUE,                               ④
    tables.height = 0.2                              ⑤
    )
```

① Add a title
② Change the colours for the two survival lines
③ Add a p-value annotation to the plot
④ Add the risk table below the plot
⑤ Set the risk table to be 20% of the plot height.



Strata	0	250	500	750	1000
sex=1	138	62	20	7	2
sex=2	90	53	21	3	0

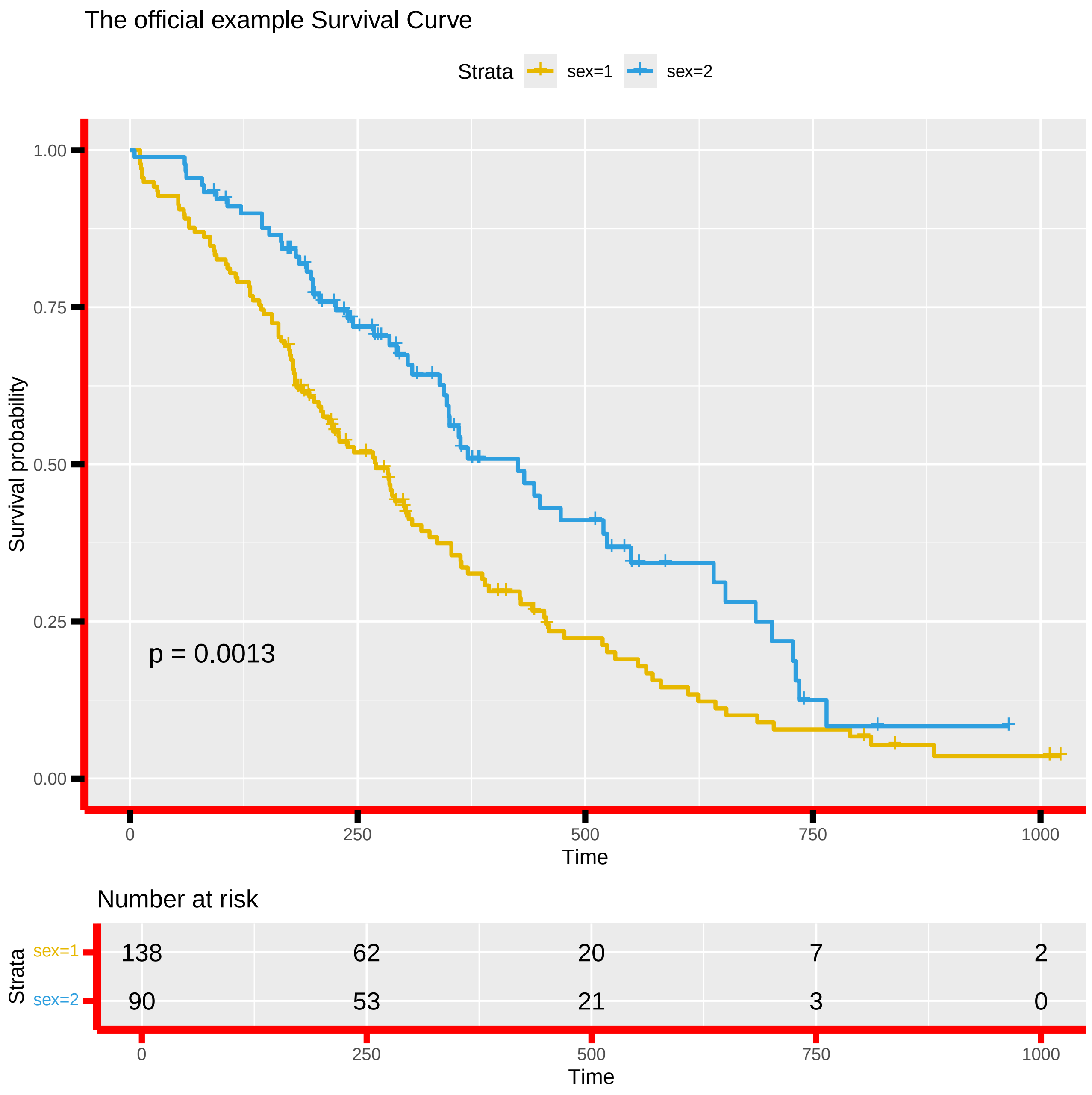
Now to customise that plot theme

ggsurvplot works using ggplot and so you might expect that you can take its result and throw a few theme() statements at it and magically make things look different. Of course if it was that easy, I'd not be writing a blog about it! ggsurvplot knows this is an issue so it lets you define the theme /within/ the call to the plot. The reason this strange approach is used, is that ggsurvplot is actually creating *two* plots - the risk table is a plot as well.

So for simplicity, lets say I wanted to change the axis to be red and a bit thicker I can use this:

```
ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = c("#E7B800", "#2E9FDF"),
    pval = TRUE,
    risk.table = TRUE,
    tables.height = 0.2,
    ggtheme = theme(                                                             ①
            axis.line = element_line(colour = "red", linewidth=2),
                        axis.ticks = element_line(colour = "red", linewidth = 1.5),
                        axis.ticks.length=unit(.25, "cm")
                     )
    )
```

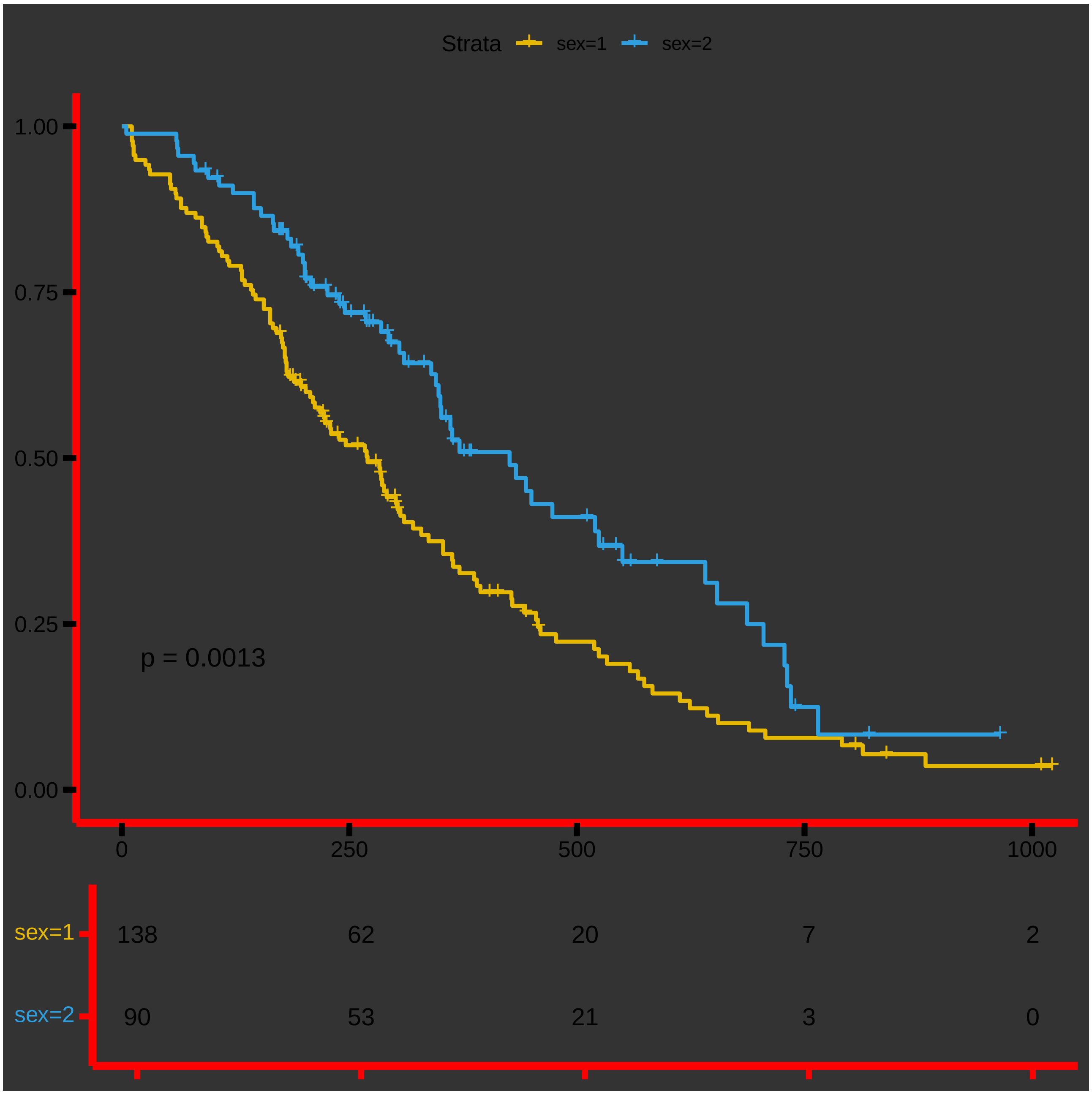
① Add a theme statement to change the axis.line to red, and the tick colour, width and length.



This is all fairly well documented and straightforward so far. And we can add a transparent background using the theme_transparent() in the ggtheme statement.

```
ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = c("#E7B800", "#2E9FDF"),
    pval = TRUE,
    risk.table = TRUE,
    tables.height = 0.2,
    ggtheme = theme_transparent() +                                          ①
            theme(
                axis.line = element_line(colour = "red", linewidth=2),
                            axis.ticks = element_line(colour = "red", linewidth = 1.5),
                            axis.ticks.length=unit(.25, "cm")
                        )
    )
```

① Add a theme_transparent statement much like you would with ggplot2 normally, but within the ggtheme statement.

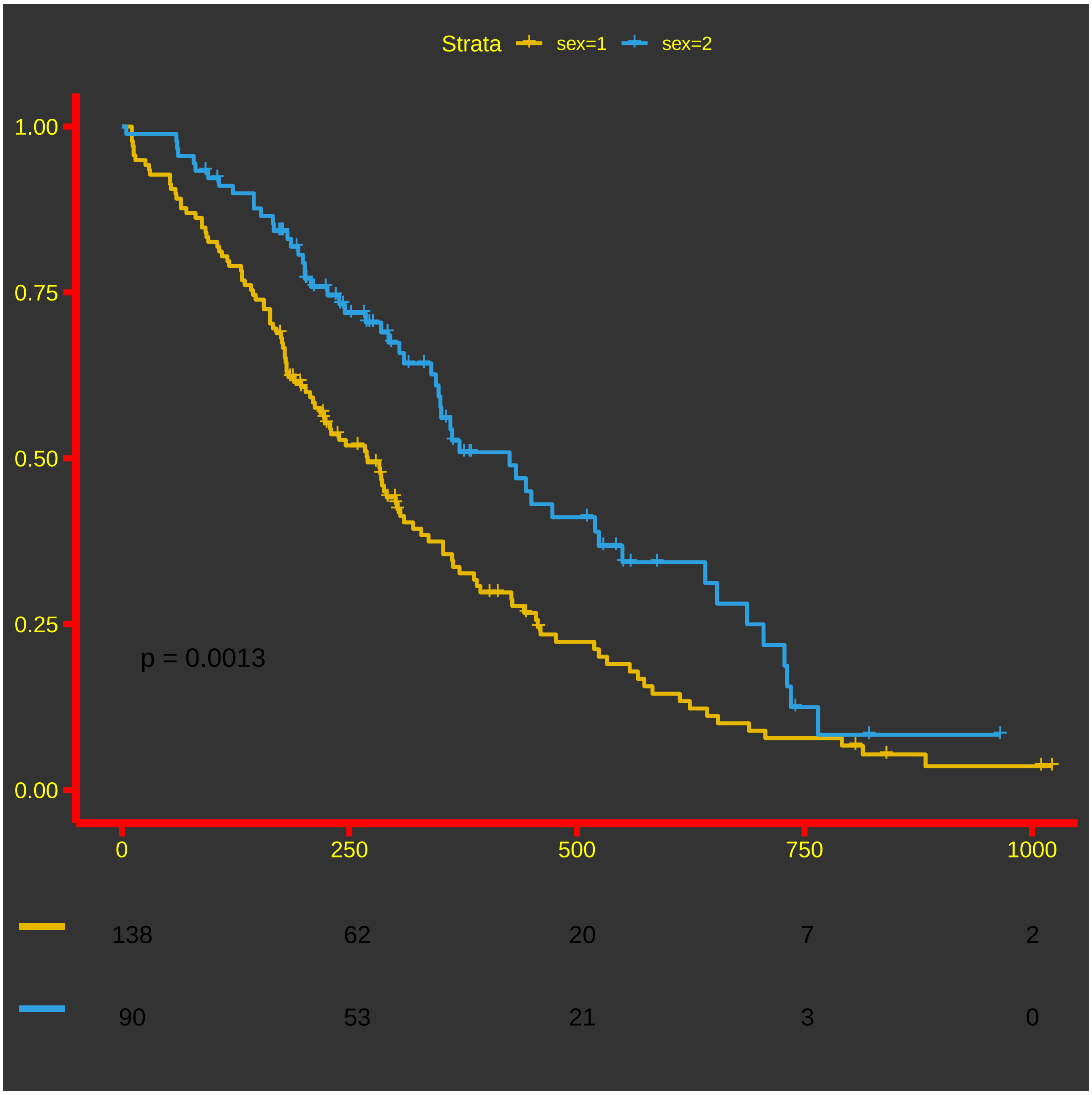


If you've not already got dark mode enabled on the site, now might be a good time to enable it!

You can use the slider at the top right of the screen. However, already there are a few things that even then aren't behaving as you might expect! The title has vanished. The tick-marks (on the main plot) are in black not red and my decision to thicken the axis has suddenly made the risk table axis really obvious. You can add a tables.theme statement as well which would let you handle its axis differently. I'm also going to switch off the text labels with - tables.y.text = FALSE.

```
ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = c("#E7B800", "#2E9FDF"),
    pval = TRUE,
    risk.table = TRUE,
    tables.height = 0.2,
    tables.y.text = FALSE,                                                 ①
    ggtheme = theme_transparent() +
            theme( text = element_text(colour = "yellow"),                 ②
                axis.line = element_line(colour = "red", linewidth=2),
                            axis.ticks.x = element_line(colour = "red", linewidth = 1.5)
                axis.ticks.y = element_line(colour = "red", linewidth = 1.5),
                            axis.ticks.length=unit(.25, "cm")
                        ),
    tables.theme = theme_cleantable()                                      ④
    )
```

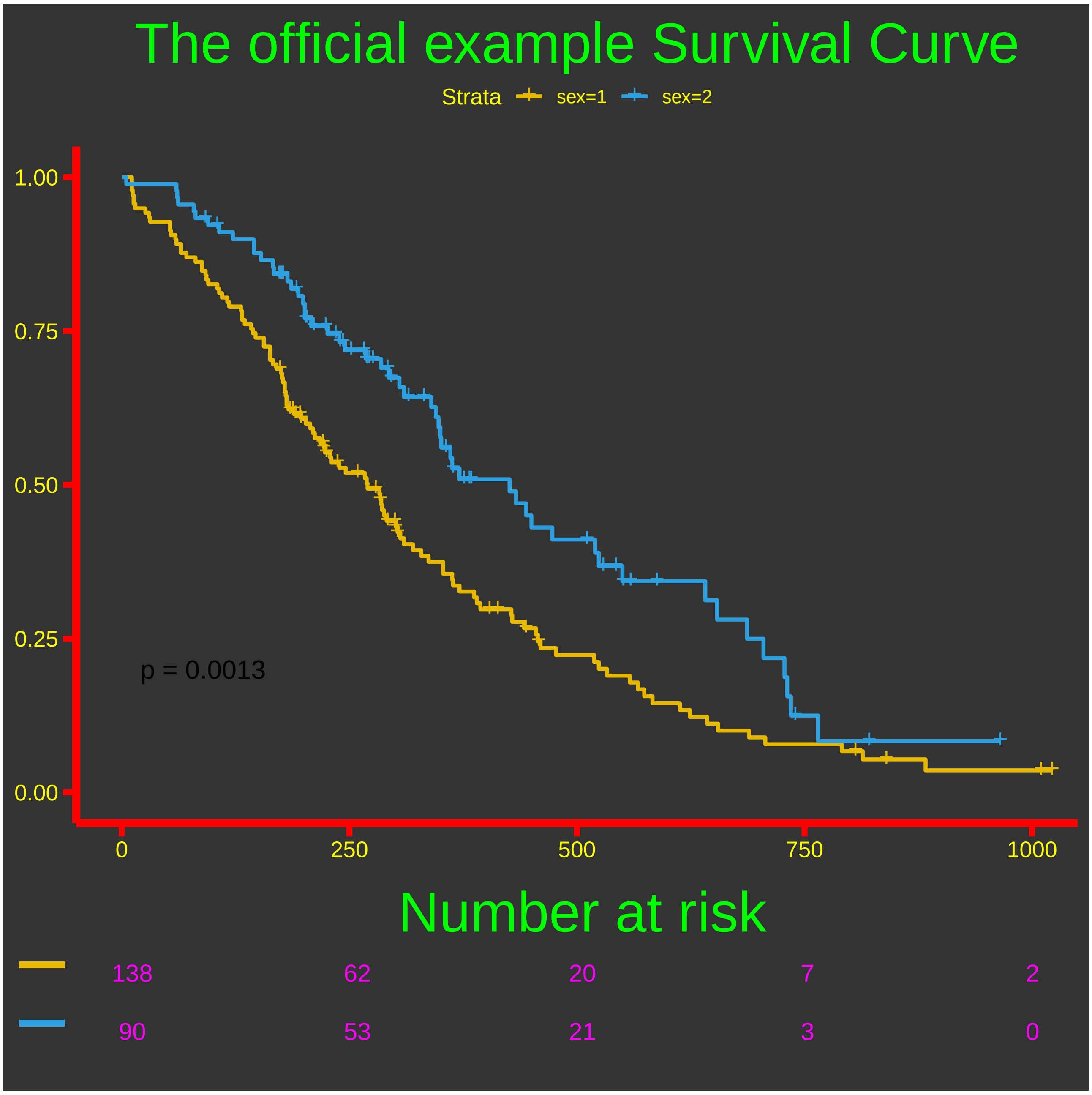
① Removing the text labels just for tidiness
② Make the text a different colour. When I'm experimenting with graphs like this I often use a mix of horrible colours to check what I'm changing, then flip to my final colour I want later.
③ Specify the axis ticks for x and y separately as they must be defined separately elsewhere in the ggsurvplot theme :-(
④ Adding a different theme to the risk table



You will notice we still have black text on the p value and the risk numbers and that we are still missing the title and the title on the risk table. The title is fairly easy to fix. The black text for the p-value is however not so simple. These bits of text are actually geom_text and the colour is specified when they are placed rather than using a theme. ggsurvplot has provided a method for the numbers at risk so we can use that.

```
ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = c("#E7B800", "#2E9FDF"),
    pval = TRUE,
    risk.table = TRUE,
    tables.height = 0.2,
    tables.y.text = FALSE,
    tables.col = "magenta",                                    (1)
    ggtheme = theme_transparent() +
            theme( text = element_text(colour = "yellow"),
                plot.title = element_text(colour = "green", size = 30),       ②
                                plot.tag=element_text(colour="orange"),       ③
                                plot.subtitle = element_text(colour = "blue", size = 25
                axis.line = element_line(colour = "red", linewidth=2),
                            axis.ticks.x = element_line(colour = "red", linewidth = 1.5
                axis.ticks.y = element_line(colour = "red", linewidth = 1.5),
                            axis.ticks.length=unit(.25, "cm")
                        ),
    tables.theme = theme_cleantable()
    )
```

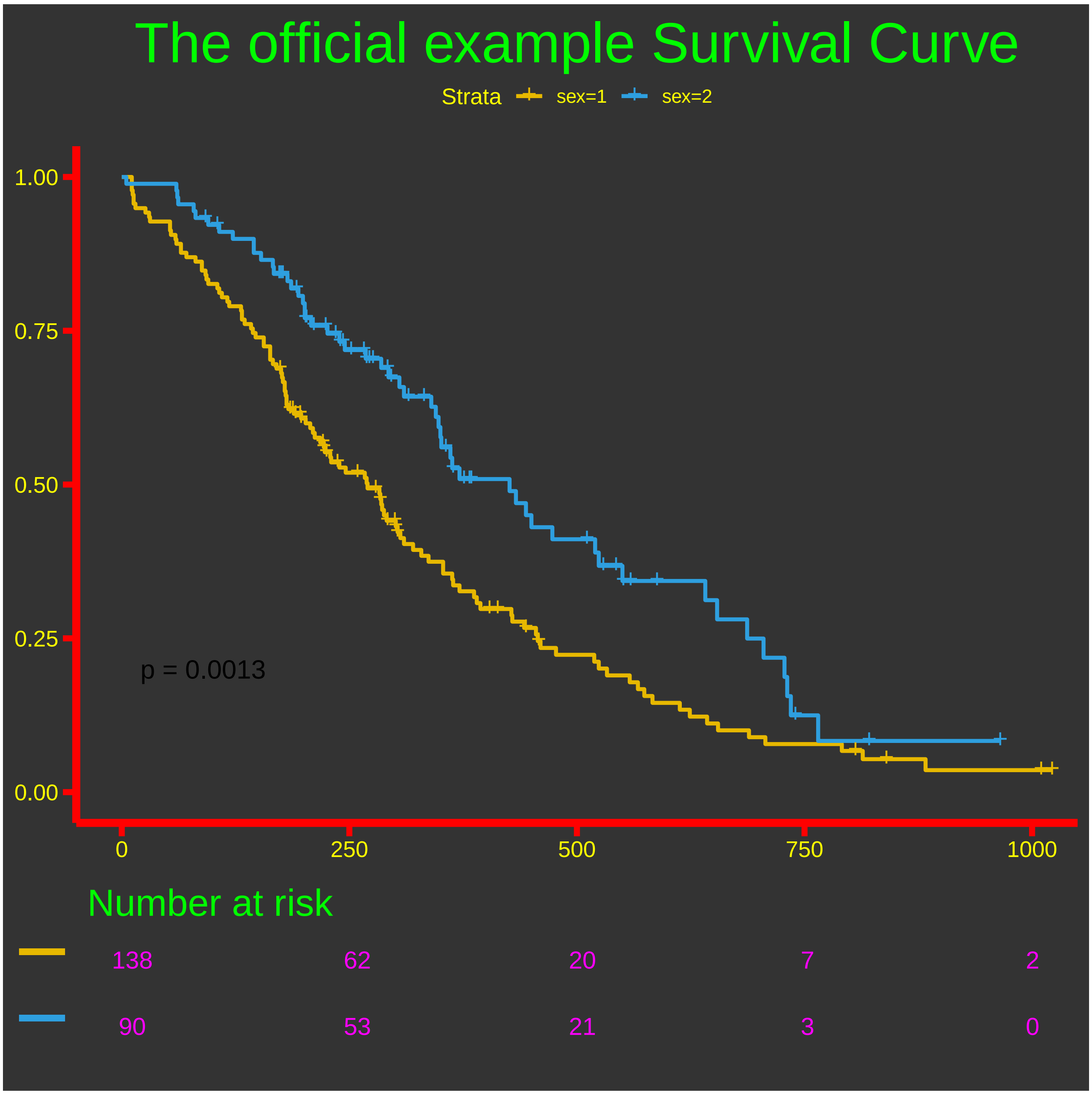
① Use tables.col to specify the text colour, or alternatively you could use tables.col = "strata" to get the line colours as the text colour.
② Specifying a plot title, and in this example a size will restore the title.
③ If you are using a subtitle, or plot tags you will need to specify these too.



If you'd like the Number at Risk to be smaller and on the left we can add it as a theme to tables.theme()

```
ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = c("#E7B800", "#2E9FDF"),
    pval = TRUE,
    risk.table = TRUE,
    tables.height = 0.2,
    tables.y.text = FALSE,
    tables.col = "magenta",
    ggtheme = theme_transparent() +
            theme( text = element_text(colour = "yellow"),
                plot.title = element_text(colour = "green", size = 30),
                                plot.tag=element_text(colour="orange"),
                                plot.subtitle = element_text(colour = "blue", size = 25
                axis.line = element_line(colour = "red", linewidth=2),
                            axis.ticks.x = element_line(colour = "red", linewidth = 1.5
                axis.ticks.y = element_line(colour = "red", linewidth = 1.5),
                            axis.ticks.length=unit(.25, "cm")
                        ),
    tables.theme = theme_cleantable() +
            theme(
                plot.title = element_text(size = 20, hjust = 0 )              ①
            )
    )
```

① Reduce the size and horizontally justify (left align) the title

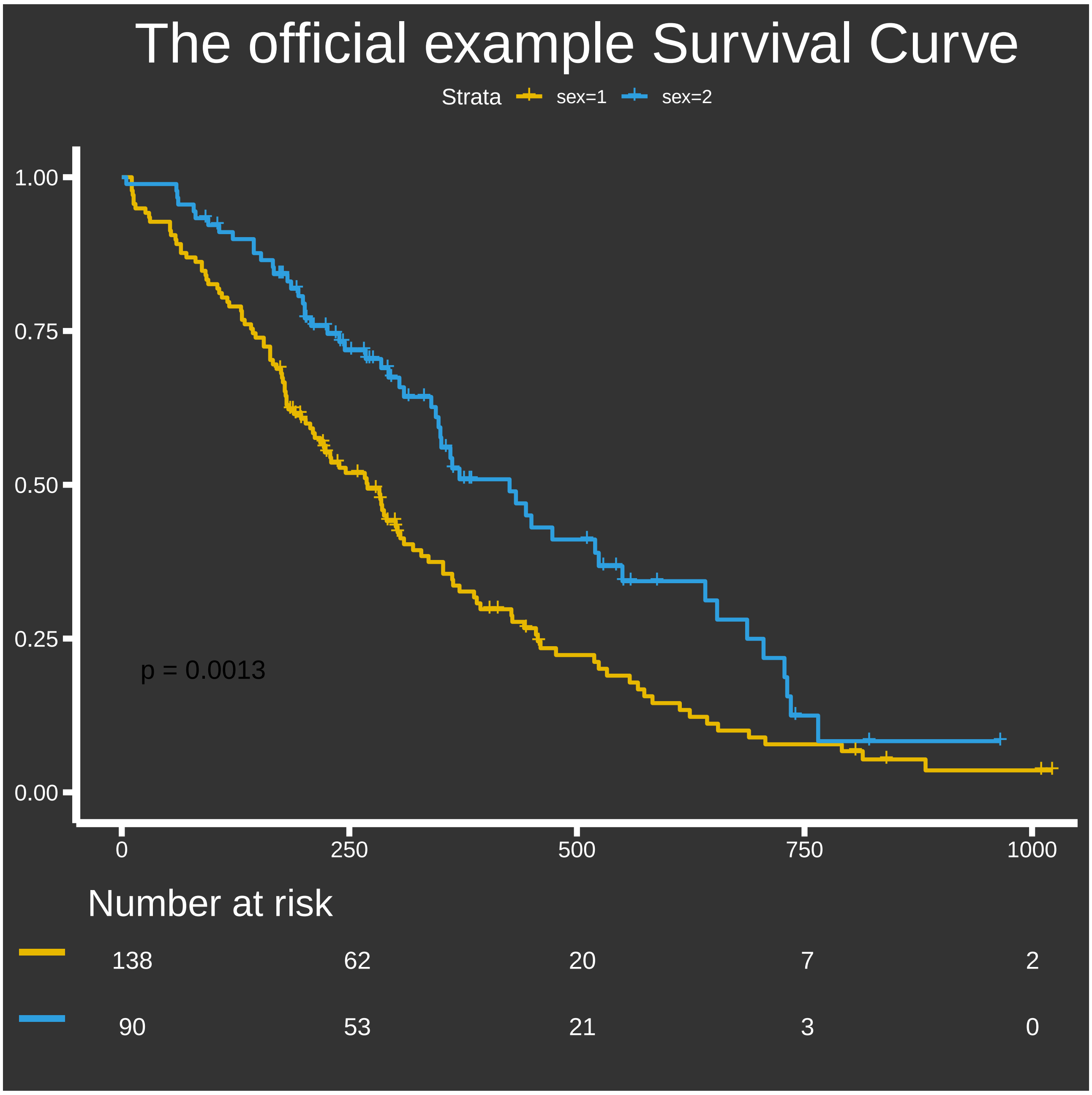


Hacking the p-value colour

We've now done as much as we can achieve within ggsurvplot directly. So lets plot this all with white instead of all these hideous colours. We will save this as an object so that we can hack the p value colour.

```
par(bg = "#000000", fg = "#FFFFFF")
ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = c("#E7B800", "#2E9FDF"),
    pval = TRUE,
    risk.table = TRUE,
    tables.height = 0.2,
    tables.y.text = FALSE,
    tables.col = "white",                                                    ①
    ggtheme = theme_transparent() +
            theme( text = element_text(colour = "white"),
                plot.title = element_text(colour = "white", size = 30),
                                plot.tag=element_text(colour="white"),
                                plot.subtitle = element_text(colour = "white", size = 25
                axis.line = element_line(colour = "white", linewidth=2),
                            axis.ticks.x = element_line(colour = "white", linewidth = 1.
                axis.ticks.y = element_line(colour = "white", linewidth = 1.5),
                            axis.ticks.length=unit(.25, "cm")
                        ),
    tables.theme = theme_cleantable() +
            theme(
                plot.title = element_text(size = 20, hjust = 0 )
            )
    ) -> myPlot                                                              ②
print(myPlot)                                                                ③
```

① Change all the colours to white
② Save the plot to an object called myPlot
③ Use print to view the plot



If we now examine the structure of myPlot with summary(myPlot) we get:

```
summary(myPlot)
```

```
               Length Class      Mode
plot           11     gg         list
table          11     gg         list
data.survplot  10     data.frame list
data.survtable 10     data.frame list
```

Which tells us there is 4 parts to this object. Two data tables containing the data, and two gg class objects (plots in essence) once called 'plot' which has the survival curve, the other 'table' which has the risk table. Examining the plot will give us more information:

```
summary(myPlot$plot)
```

```
data: time, n.risk, n.event, n.censor, surv, std.err, upper, lower,
  strata, sex [208x10]
```

```
mapping:  x = ~time, y = ~surv
scales:   y, ymin, ymax, yend, yintercept, ymin_final, ymax_final, lower, middle, upper,
faceting: <ggproto object: Class FacetNull, Facet, gg>
    compute_layout: function
    draw_back: function
    draw_front: function
    draw_labels: function
    draw_panels: function
    finish_data: function
    init_scales: function
    map_data: function
    params: list
    setup_data: function
    setup_params: function
    shrink: TRUE
    train_scales: function
    vars: function
    super:  <ggproto object: Class FacetNull, Facet, gg>
-----------------------------------
mapping: colour = ~strata
geom_step: direction = hv, na.rm = FALSE
stat_identity: na.rm = FALSE
position_identity

mapping: x = ~x, y = ~y
geom_blank: na.rm = FALSE
stat_identity: na.rm = FALSE
position_identity

mapping: colour = ~strata
geom_point: na.rm = FALSE
stat_identity: na.rm = FALSE
position_identity

mapping: x = ~x, y = ~y
geom_text: na.rm = FALSE
stat_identity: na.rm = FALSE
position_identity
```

This shows there are 4 parts to this particular plot (your plots may vary!). Importantly the 4th part is geom_text - this likely contains our p-value information. These sit in something called a layer and we should look at the 4th Layer. We are interested in the aesthetics so we need to look at the aes_params:

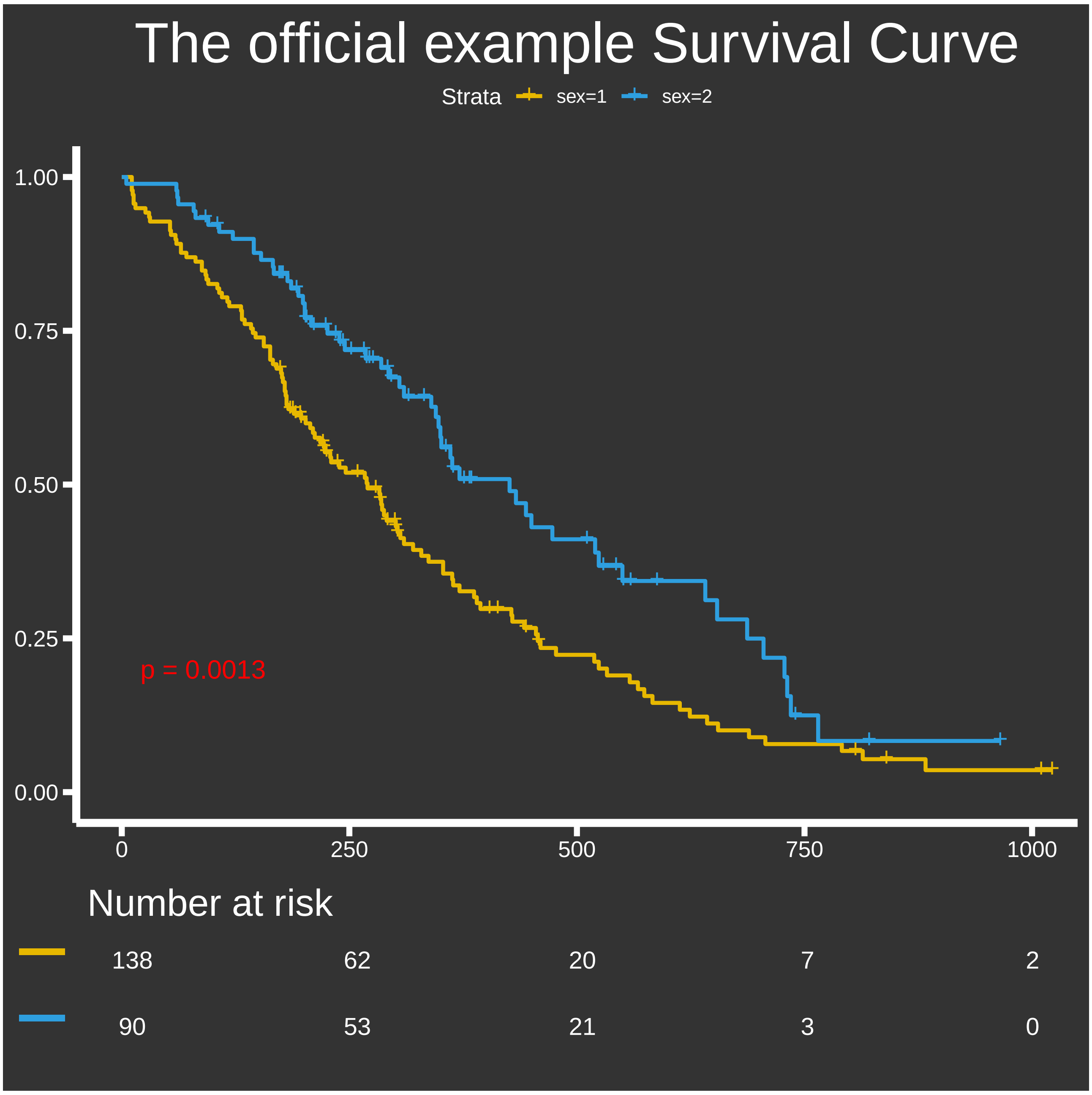
```
myPlot$plot$layers[[4]]$aes_params
```

```
$label
[1] "p = 0.0013"

$size
[1] 5
```

```
$hjust
[1] 0

$family
[1] ""
```

This confirms we are looking that p-value and shows that no colour has been specified. Adding a colour is as easy as:

```
myPlot$plot$layers[[4]]$aes_params$colour <- "red"
print(myPlot)
```



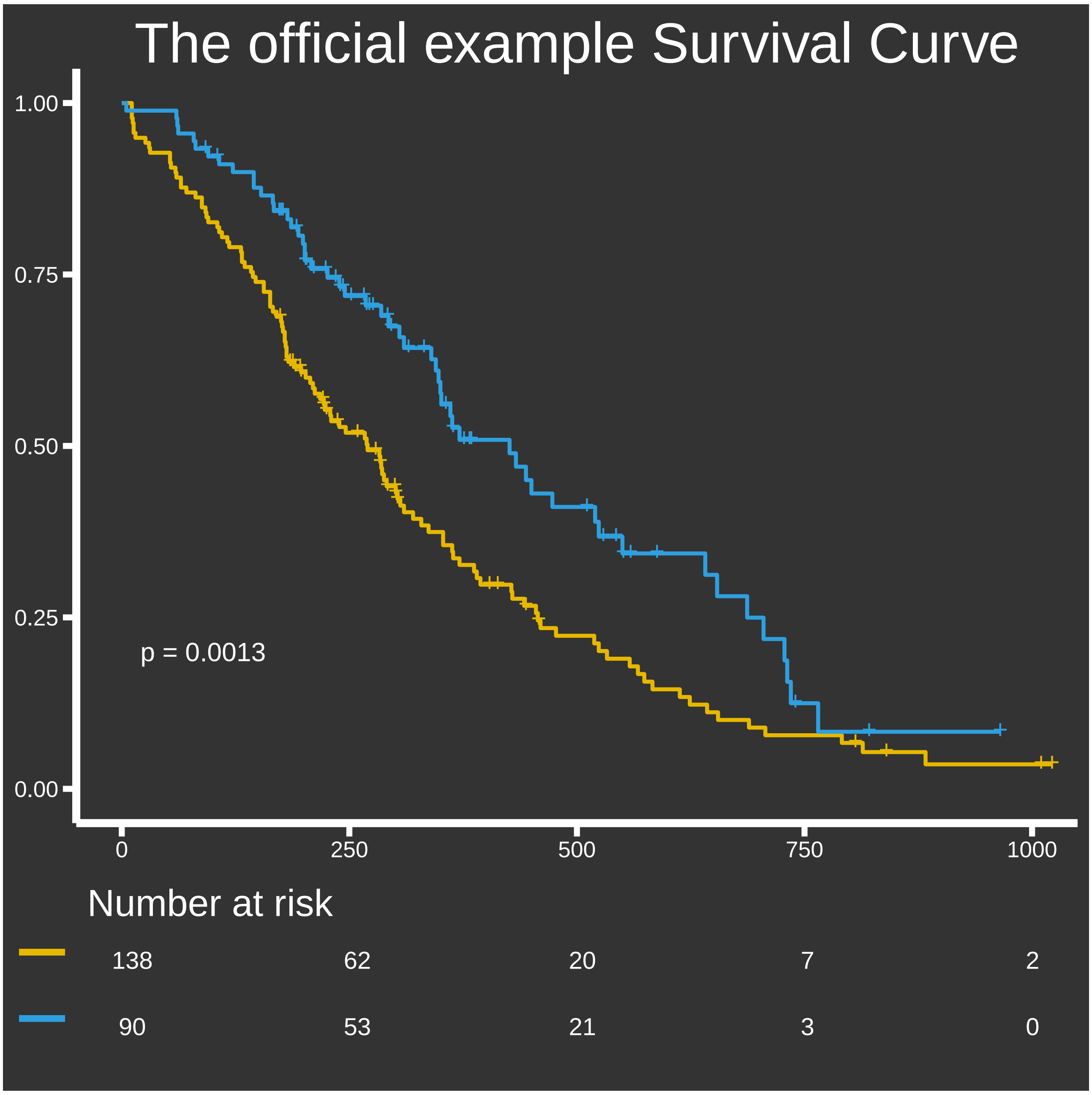
	0	250	500	750	1000
🟨	138	62	20	7	2
🟦	90	53	21	3	0

Labelling the line instead of a legend

I also wanted to edit the legend to annotate those alongside my lines rather than at the top of the window as a legend. We should therefore return to our ggsurvplot command to disable the legend:

```
ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = c("#E7B800", "#2E9FDF"),
    pval = TRUE,
    legend = "none",                                                        ①
    risk.table = TRUE,
    tables.height = 0.2,
    tables.y.text = FALSE,
    tables.col = "white",
    ggtheme = theme_transparent() +
            theme( text = element_text(colour = "white"),
                plot.title = element_text(colour = "white", size = 30),
                                plot.tag=element_text(colour="white"),
                                plot.subtitle = element_text(colour = "white", size = 25
                axis.line = element_line(colour = "white", linewidth=2),
                            axis.ticks.x = element_line(colour = "white", linewidth = 1.
                axis.ticks.y = element_line(colour = "white", linewidth = 1.5),
                            axis.ticks.length=unit(.25, "cm")
                        ),
    tables.theme = theme_cleantable() +
            theme(
                plot.title = element_text(size = 20, hjust = 0 )
            )
    ) -> myPlot
myPlot$plot$layers[[4]]$aes_params$colour <- "white"                        ②
print(myPlot)
```

① Disable the legend
② Update the p-value colour



As myPlot$plot contains the survival curve we can use ggannotate to, erm, annotate it! Strata are ordered in the order of the factor for their grouping. So in our example at the very start we created a survival object with:

```
fit<- survfit(Surv(time, status) ~ sex, data = lung)
```

and 'sex' is where the strata comes from. Unhelpfully they are coded as sex = 1 or 2. I gather that the correct allocation is 1 = Male and 2 = Female. It would probably be better to fix this *before* we create the survival object. So lets go back and do that:

```
require(forcats)                                                    ①

lung$sex |>
    as.character() |>                                               ②
    fct_recode ( "Male" = "1", "Female" = "2") -> lung$sex          ③

fit<- survfit(Surv(time, status) ~ sex, data = lung)                ④
```

```
ggsurvplot(                                                                      ⑤
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = c("#E7B800", "#2E9FDF"),
    pval = TRUE,
    risk.table = TRUE,
    tables.height = 0.2,
    tables.y.text = FALSE
    )
```

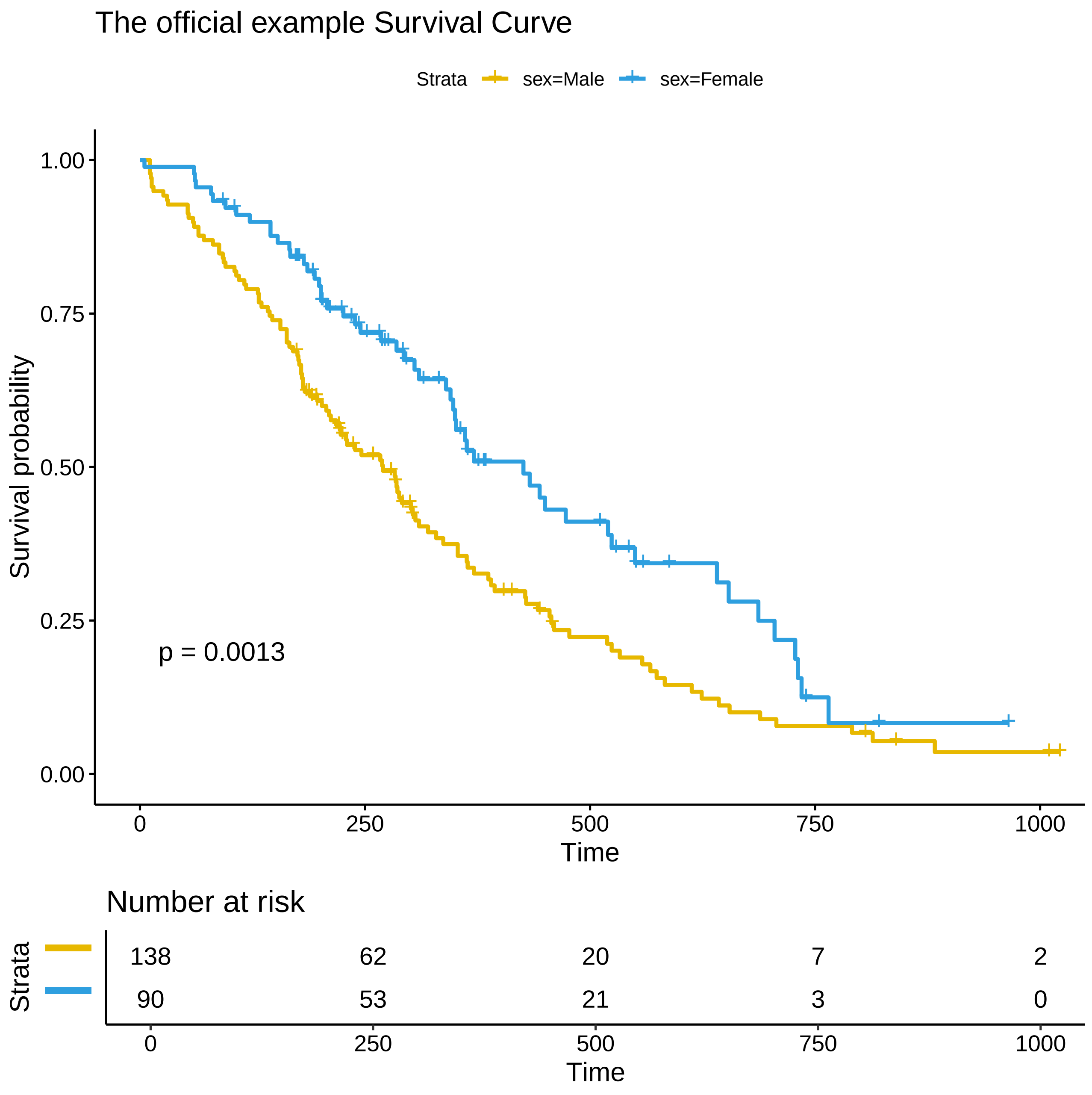
① I prefer to use tidyverse forcats package for factors, but you could use base R to achieve the same

② The data is currently stored as a number and not a factor. forcats can't process a number it needs a character or factor so lets make it a character

③ Re-code the data and save it back into the lung$sex data column. You could of course save it elsewhere and use that if you didn't want to affect the source data

④ Refit the data with the coded values

⑤ Re run the graph, I've used a shortened version here with the legend re-enabled to show which is which for colours - yellow is male and blue is female



To annotate the graphs we will need to know:

- Where we want the labels to be
- The label colours
- The label names

The label colours is 'easy' we are already defining that with the line palette = c("#E7B800", "#2E9FDF") in the plot and it would be sensible to use labels of the same colour as the line.

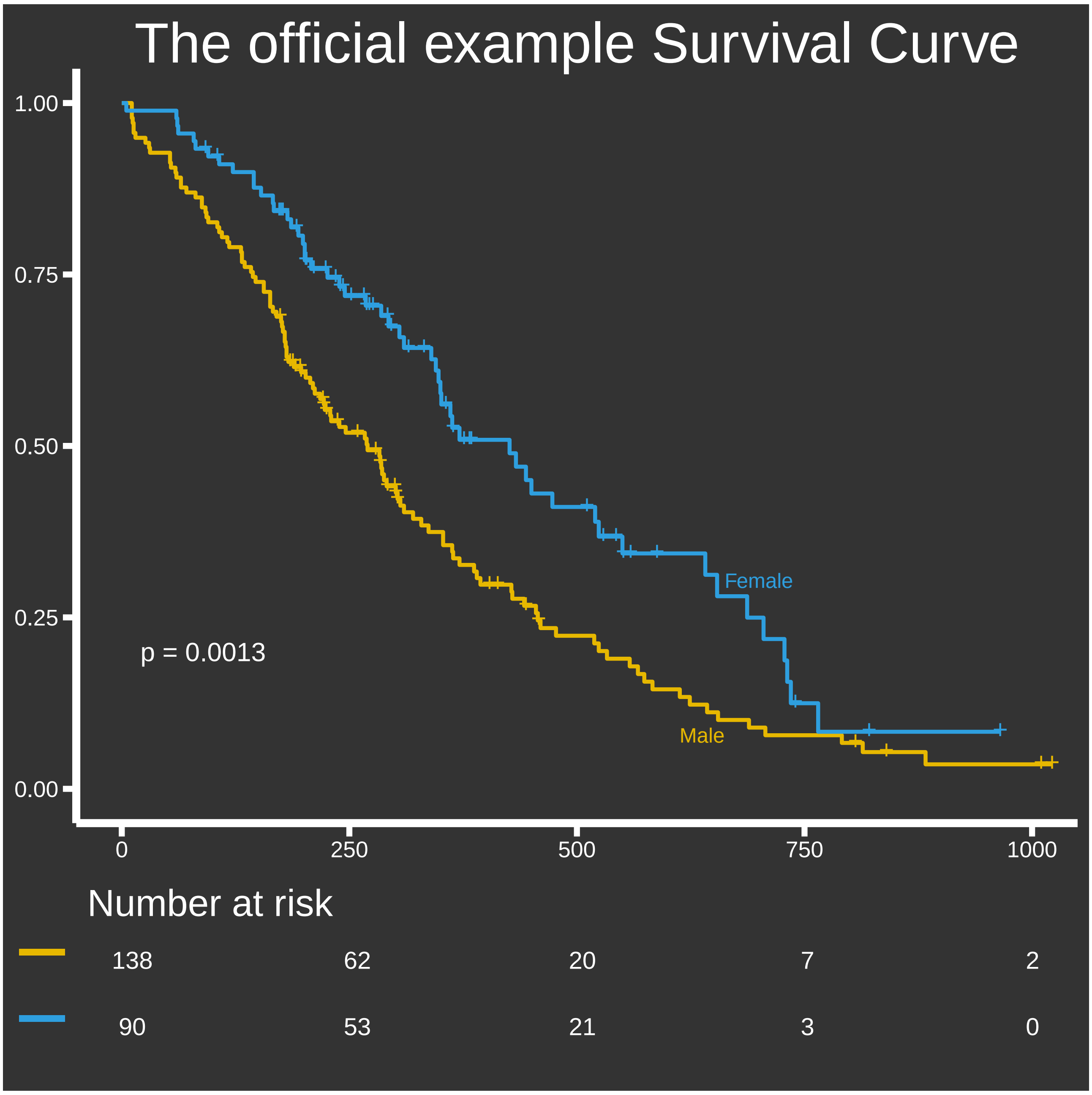
The label names is also 'easy' as we simply need to take the factor levels. These will be applied to the colours in order. So the first factor gets #E7B800 as a colour.

Positioning by code is harder. You may simply decide to do it using some hard entered numbers of the co-ordinates you want to use. But I thought this little bit of code might work:

```
line_colours =  c("#E7B800", "#2E9FDF")     ①
label_names = levels(lung$sex)              ②
```

```
# Get the lower line name
lower_line <- surv_median(fit)$strata[surv_median(fit)$median == min(surv_median(fit)$me
upper_line <- surv_median(fit)$strata[surv_median(fit)$median == max(surv_median(fit)$me

# establish 75% of x-axis
sum_fit <- summary(fit)
x75 <- max(sum_fit$time, na.rm=T) * 0.75

# get the lower line level at 75% of the way along the line
lower_line_y <- sum_fit$surv[sum_fit$time >= x75 & sum_fit$strata == lower_line][1]

# get the upper line level at 75% of the way along the line
upper_line_y <- sum_fit$surv[sum_fit$time >= x75 & sum_fit$strata == upper_line][1]

# annotate the lower line with the right hand to corner of the label at 75% & lower_line

myPlot$plot <- myPlot$plot + annotate(                                              ④
            "text",
            x= x75,
            y = lower_line_y,
            label= label_names[surv_median(fit)$strata ==lower_line],
            colour=line_colours[surv_median(fit)$strata ==lower_line],
            hjust = 1,
            vjust = 1
            )

# annotate the upper line above the line

myPlot$plot <- myPlot$plot + annotate(
            "text",
            x= x75,
            y = upper_line_y,
            label= label_names[surv_median(fit)$strata ==upper_line],
            colour=line_colours[surv_median(fit)$strata ==upper_line],
            hjust = 0,
            vjust = -2
            )

print(myPlot)
```

① define the line colours as a vector
② define the label names for the graph
③ establish where to plot the annotations
④ plot the annotations



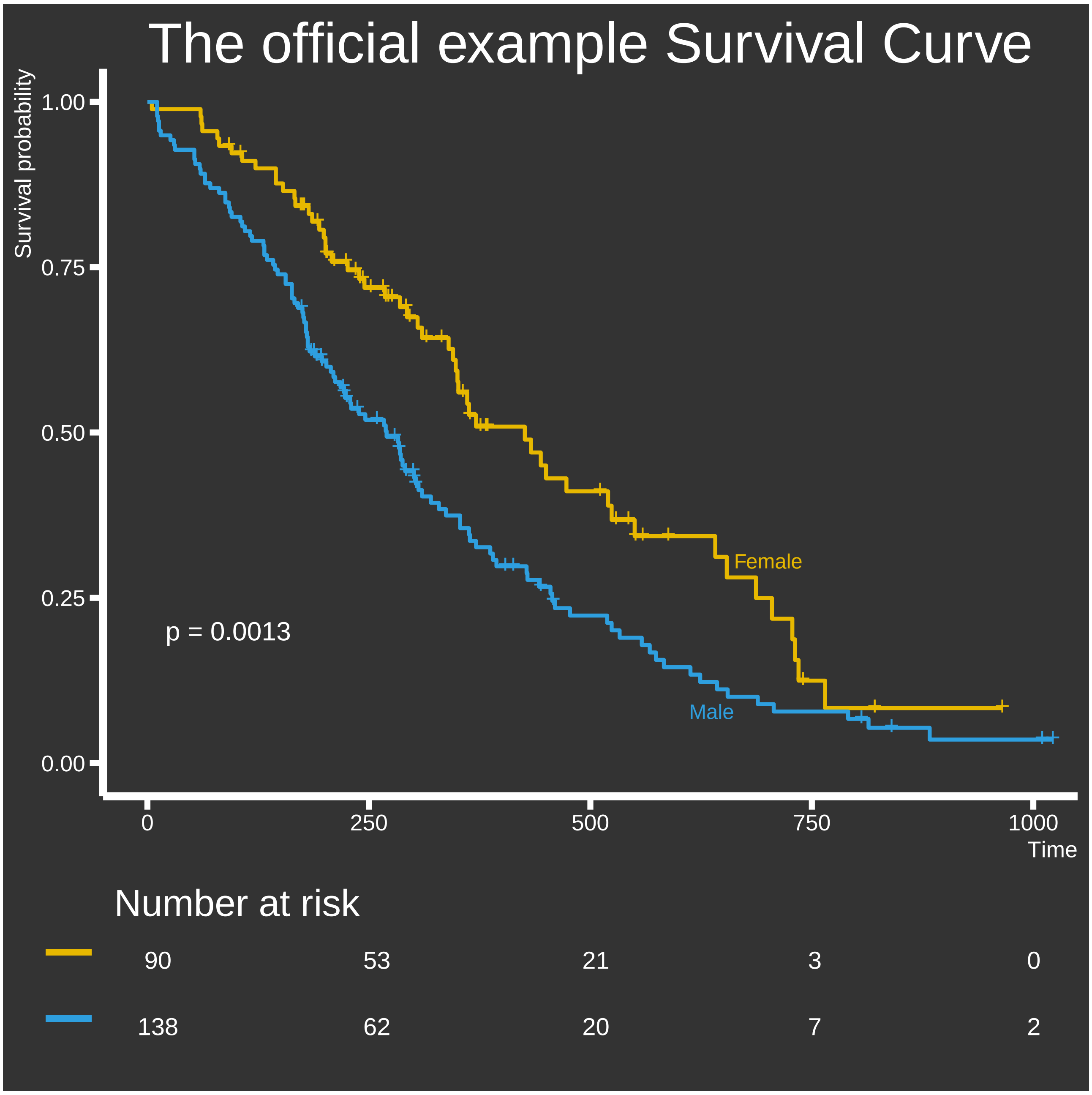
For completeness - we can create this all as a single piece of code, and we will replace the line colouration inf the ggsurvplot function so that it couldn't be swapped around in error.

```
require(survival)
require(survminer)
require(forcats)

data(lung)                                                                    ①

lung$sex |>
    as.character() |>
    fct_recode ( "Male" = "1", "Female" = "2") -> lung$sex

fit<- survfit(Surv(time, status) ~ sex, data = lung)

line_colours =  c("#E7B800", "#2E9FDF")
```

```
label_names = levels(lung$sex)

ggsurvplot(
    fit,
    data = lung,
    title = "The official example Survival Curve",
    palette = line_colours,                                              ②
    pval = TRUE,
    legend = "none",
    risk.table = TRUE,
    tables.height = 0.2,
    tables.y.text = FALSE,
    tables.col = "white",
    ggtheme = theme_transparent() +
            theme( text = element_text(colour = "white"),
                plot.title = element_text(colour = "white", size = 30),
                                plot.tag=element_text(colour="white"),
                                plot.subtitle = element_text(colour = "white", size = 25
                axis.line = element_line(colour = "white", linewidth=2),
                            axis.ticks.x = element_line(colour = "white", linewidth = 1.
                axis.title.x = element_text(colour = "white", hjust=1),      ③
                axis.title.y = element_text(colour = "white", hjust=1),
                axis.ticks.y = element_line(colour = "white", linewidth = 1.5),
                            axis.ticks.length=unit(.25, "cm")
                        ),
    tables.theme = theme_cleantable() +
            theme(
                plot.title = element_text(size = 20, hjust = 0 )
            )
    ) -> myPlot

# change the p-value colour
myPlot$plot$layers[[4]]$aes_params$colour <- "white"

# Get the lower line name
lower_line <- surv_median(fit)$strata[surv_median(fit)$median == min(surv_median(fit)$me
upper_line <- surv_median(fit)$strata[surv_median(fit)$median == max(surv_median(fit)$me

# establish 75% of x-axis
sum_fit <- summary(fit)
x75 <- max(sum_fit$time, na.rm=T) * 0.75

# get the lower line level at 75% of the way along the line
lower_line_y <- sum_fit$surv[sum_fit$time >= x75 & sum_fit$strata == lower_line][1]

# get the upper line level at 75% of the way along the line
upper_line_y <- sum_fit$surv[sum_fit$time >= x75 & sum_fit$strata == upper_line][1]

# annotate the lower line with the right hand to corner of the label at 75% & lower_line
```

```
myPlot$plot <- myPlot$plot + annotate(
            "text",
            x= x75,
            y = lower_line_y,
            label= label_names[surv_median(fit)$strata ==lower_line],
            colour=line_colours[surv_median(fit)$strata ==lower_line],
            hjust = 1,
            vjust = 1
            )

# annotate the upper line above the line

myPlot$plot <- myPlot$plot + annotate(
            "text",
            x= x75,
            y = upper_line_y,
            label= label_names[surv_median(fit)$strata ==upper_line],
            colour=line_colours[surv_median(fit)$strata ==upper_line],
            hjust = 0,
            vjust = -2 # this is a bit unpredictable
            )

print(myPlot)
```

① reload the lung data-set as we've been messing with it
② insert the line colour vector
③ expose the axis labels which I missed earlier!



Finally, if we want to save the plot as a vector or png, we need to tell ggsave not to apply a background colour!

```
ggsave("myPlot.png", bg="transparent")
```

```
grateful::cite_packages(output = "paragraph", out.dir = ".")
```

We used R version 4.3.0[1] and the following R packages: cowplot v. 1.1.3[2], DT v. 0.33[3], glue v. 1.8.0[4], knitr v. 1.50[5–7], qicharts2 v. 0.7.5[8], rmarkdown v. 2.29[9–11], survival v. 3.8.3[12,13], survminer v. 0.5.0[14], tidyverse v. 2.0.0[15].

References

1. R Core Team. (2023). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing. https://www.R-project.org/
2. Wilke, C. O. (2024). *cowplot: Streamlined plot theme and plot annotations for "ggplot2".* https://CRAN.R-project.org/package=cowplot

3. Xie, Y., Cheng, J., & Tan, X. (2024). *DT: A wrapper of the JavaScript library "DataTables"*. https://CRAN.R-project.org/package=DT
4. Hester, J., & Bryan, J. (2024). *glue: Interpreted string literals*. https://CRAN.R-project.org/package=glue
5. Xie, Y. (2014). knitr: A comprehensive tool for reproducible research in R. In V. Stodden, F. Leisch, & R. D. Peng (Eds.), *Implementing reproducible computational research*. Chapman; Hall/CRC.
6. Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Chapman; Hall/CRC. https://yihui.org/knitr/
7. Xie, Y. (2025). *knitr: A general-purpose package for dynamic report generation in R*. https://yihui.org/knitr/
8. Anhoej, J. (2024). *qicharts2: Quality improvement charts*. https://CRAN.R-project.org/package=qicharts2
9. Xie, Y., Allaire, J. J., & Grolemund, G. (2018). *R markdown: The definitive guide*. Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown
10. Xie, Y., Dervieux, C., & Riederer, E. (2020). *R markdown cookbook*. Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook
11. Allaire, J., Xie, Y., Dervieux, C., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W., & Iannone, R. (2024). *rmarkdown: Dynamic documents for r*. https://github.com/rstudio/rmarkdown
12. Terry M. Therneau, & Patricia M. Grambsch. (2000). *Modeling survival data: Extending the Cox model*. Springer.
13. Therneau, T. M. (2024). *A package for survival analysis in r*. https://CRAN.R-project.org/package=survival
14. Kassambara, A., Kosinski, M., & Biecek, P. (2024). *survminer: Drawing survival curves using "ggplot2"*. https://CRAN.R-project.org/package=survminer
15. Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686