Paper ST16

How to Customize Survival Curves in R and Add Interactivity

Oleksandr Babych, Intego Group LLC, Warsaw, Poland

ABSTRACT

Survival analysis plays a crucial role in clinical trials and quite often can be its primary objective. The basic tool of survival analysis is Kaplan-Meier estimates of survival curves. R is a well-known tool for data visualization which allows researchers to produce high-quality figures. The main advantage of R is that it provides a broad range of fully customizable data visualization options that are easily accessible even for beginners. In addition to that, modern data visualizations often require interactivity and R is a perfect tool to add one. This paper will focus on how Kaplan-Meier plots can be built and customized in R. Moreover, we will demonstrate which interactivity can be added to KM plots using plotly and highcahrts R packages.

HOW TO GENERATE A SURVIVAL CURVE IN R

To build Kaplan-Meier plots we will use the lung dataset from the `survival` package. The dataset contains survival data for patients with advanced lung cancer from the North Central Cancer Treatment Group. It was slightly updated to be more ADaM compliant and consist of common variables:

	USUBJID	SUBJID	SEX	AGE	AGEU	PARAMCD	PARAM	AVAL	AVALU	CNSR
1	01-001	001	1	74	YEARS	PFS	Progression Free Survival	306	DAYS	0
2	01-002	002	1	68	YEARS	PFS	Progression Free Survival	455	DAYS	0
3	01-003	003	1	56	YEARS	PFS	Progression Free Survival	1010	DAYS	1
4	01-004	004	1	57	YEARS	PFS	Progression Free Survival	210	DAYS	0
5	01-005	005	1	60	YEARS	PFS	Progression Free Survival	883	DAYS	0
6	01-006	006	1	74	YEARS	PFS	Progression Free Survival	1022	DAYS	1
7	01-007	007	2	68	YEARS	PFS	Progression Free Survival	310	DAYS	0
8	01-008	008	2	71	YEARS	PFS	Progression Free Survival	361	DAYS	0
9	01-009	009	1	53	YEARS	PFS	Progression Free Survival	218	DAYS	0
10	01-010	010	1	61	YEARS	PFS	Progression Free Survival	166	DAYS	0

We can see the time-to-event parameter in days (AVAL) with a censoring status (CNSR) where 1 corresponds to 'event' and 0 to 'censored'.

SURVIVAL PACKAGE

The core package for the survival analysis has already been mentioned and is called `survival`. The most vital functions are:

- `Surv()` – creates a survival object. The result of the `Surv()` is usually used as input by other survival package functions. By default, the status/event indicator expects the following format: 0=alive, 1=dead. Note that R functions are case-sensitive, so this function should always start with a capital "S".
- `survfit()` – function creates survival curves using the Kaplan-Meier method based on a formula. Usually, the `survfit()` function uses the `Surv` object as a response variable in the model formula.

So to build a survival object we will use the `Surv()` function and set two arguments `time` (AVAL) and `event` (CNSR). Since `Surv()` function by default considers '1' as the event value, we need to reassign the event argument to '0'. The object of type Surv contains the time information and whether the event occurred or not. The head function displays the first 10 elements of `surv_obj` as a vector where "+" sign marks the censored events. It can be double-checked by looking at the ADTTE dataset shown higher.

```
surv_obj <- Surv(time = ADTTE$AVAL, event = ADTTE$CNSR == 0)
head(surv_obj, 10)

[1]  306   455  1010+  210   883  1022+  310   361   218   166
```

The next step is to create survival curves. To fit the Kaplan-Meier curves we need to pass the Surv (`surv_obj`) object to the formula argument of the `survfit()` function. Here there are two equivalent `survfit()` calls that produce the same result. The argument name 'formula' can be omitted. The "~ 1" part in the formula means that we

run the model for overall progression free survival without considering any groupings. To add a stratification factor you just need to put a variable name instead of '1' like "~ SEX" which will be shown later.

```
surv_fit <- survfit(formula = Surv(time = AVAL, event = CNSR == 0) ~ 1, data = ADTTE)
surv_fit <- survfit(surv_obj ~ 1)
```

There are 3 main functions which help to investigate the survfit object. The summary of survfit object gives you a life table that contains survival statistics each time an event occurs.

```
summary(surv_fit, times = c(0,10,20,40,80))
```

```
 time n.risk n.event survival std.err lower 95% CI upper 95% CI
    0    228       0    1.000 0.00000        1.000        1.000
   10    227       1    0.996 0.00438        0.987        1.000
   20    220       7    0.965 0.01219        0.941        0.989
   40    217       3    0.952 0.01419        0.924        0.980
   80    205      12    0.899 0.01995        0.861        0.939
```

So, the summary table for each time step shows:

- `time` – the time step
- `n.risk` – the number of patients who were at risk (patients who didn't have an event and were not censored yet)
- `n.event` – patients who an event
- `survival` – the probability of not developing the event past that specific time (i.e. surviving till this particular moment in time)
- standard error and the confidence interval for that probability

The `print()` function allows to show the restricted mean survival time and its standard error:

```
print(surv_fit, print.rmean = TRUE)
```

```
    n events rmean* se(rmean) median 0.95LCL 0.95UCL
  228    165    376      19.7    310     285     363
 * restricted mean with upper limit =  1022
```

The `str()` function provides the list with detailed information regarding the survfit object structure:

```
str(surv_fit)
```

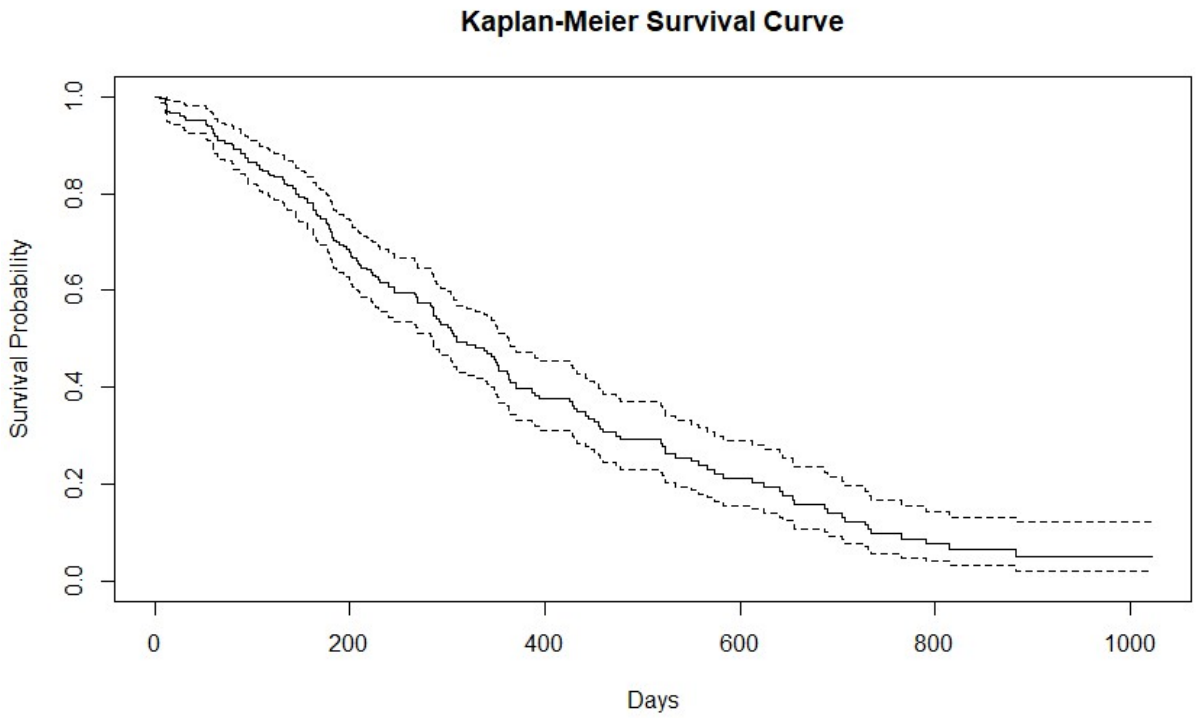
```
List of 16
 $ n        : int 228
 $ time     : num [1:186] 5 11 12 13 15 26 30 31 53 54 ...
 $ n.risk   : num [1:186] 228 227 224 223 221 220 219 218 217 215 ...
 $ n.event  : num [1:186] 1 3 1 2 1 1 1 1 2 1 ...
 $ n.censor : num [1:186] 0 0 0 0 0 0 0 0 0 0 ...
 $ surv     : num [1:186] 0.996 0.982 0.978 0.969 0.965 ...
 $ std.err  : num [1:186] 0.0044 0.00885 0.00992 0.01179 0.01263 ...
 $ cumhaz   : num [1:186] 0.00439 0.0176 0.02207 0.03103 0.03556 ...
 $ std.chaz : num [1:186] 0.00439 0.0088 0.00987 0.01173 0.01257 ...
 $ type     : chr "right"
 $ logse    : logi TRUE
 $ conf.int : num 0.95
 $ conf.type: chr "log"
 $ lower    : num [1:186] 0.987 0.966 0.959 0.947 0.941 ...
 $ upper    : num [1:186] 1 1 0.997 0.992 0.989 ...
 $ call     : language survfit(formula = Surv(time = AVAL, event = CNSR == 0) ~ 1, data = ADTTE)
 - attr(*, "class")= chr "survfit"
```

The important element here is `cumhaz` numeric vector which corresponds to cumulative hazard. It can be used to plot the cumulative hazard curve instead of the survival probability.

HOW TO PLOT THE KAPLAN-MEIER CURVES IN R

Once the survfit object is created (i.e. Kaplan-Meier estimates are fitted) – everything is ready to produce a graph. The survival probability can be visualized by the basic `plot()` function that draws the "Kaplan-Meier curve". Basically to visualize the survival curve you just need to pass the survfit object to the plot function: `plot(surv_fit)`. The `conf.int` argument allows to add the confidence interval (default value is TRUE). The `mark.time` arguments prints '+' sign on the curve for every event (default value is FALSE). The `fun` argument allows printing the cumulative events (i.e. mortality for our case) instead of the survival.

```
plot(surv_fit,                              # survfit object which contains survival curves
     conf.int = TRUE,                       # plots the confidence interval
     mark.time = FALSE,                     # removes '+' sign for every event
     main= "Kaplan-Meier Survival Curve",   # figure title
     xlab = "Days",                         # x-axis label
     ylab="Survival Probability"            # y-axis label
     # , fun = "event"                      # draws the cumulative events instead of the
                                            # survival (i.e mortality)
)
```

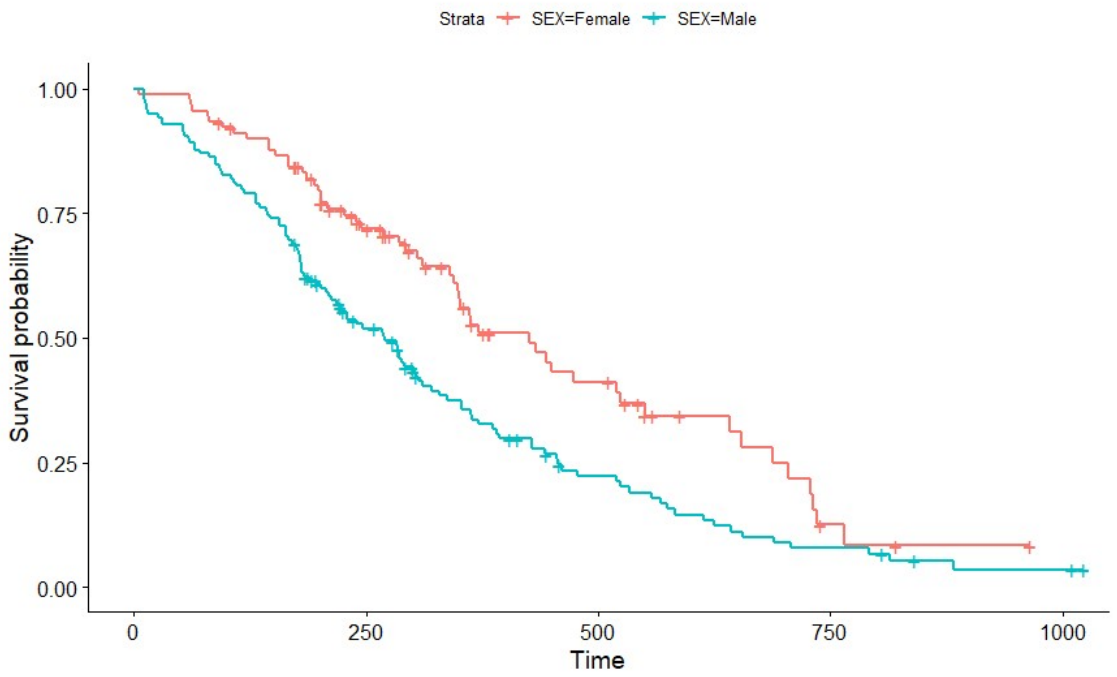


As a result, we receive a Kaplan-Meier plot which presents estimates of survival probability depending on days for patients with lung cancer. After all the basic `plot()` function produces the Kaplan-Meier curve which looks pretty "basic". This is not something we expect from R. Let's discuss more sophisticated R packages which were specifically written for plotting the Kaplan-Meier curves.

SURVMINER PACKAGE

One of the most common R packages for survival curve visualization is survminer. The most essential function of this package is `ggsurvplot()` which draws beautiful and publication-ready survival curves using ggplot2. Let's look at the most basic call. First of all, we will load the survminer package by calling the `library()` function (don't forget to install package first by calling `install.packages("survminer")`). Moreover, we will stratify the curve depending on the subject's gender (SEX variable).

```
library(survminer)
surv_fit2 <- survfit(Surv(time = AVAL, event = CNSR == 0) ~ SEX, data = ADTTE)
ggsurvplot(surv_fit2)
```



Here is what the basic `ggsurvplot()` looks like. At first sight, the graph looks more attractive than the one produced by the base package. The plot also shows estimates of survival probability depending on days for two groups of patients with lung cancer. Let's take a look at how we can enhance it.

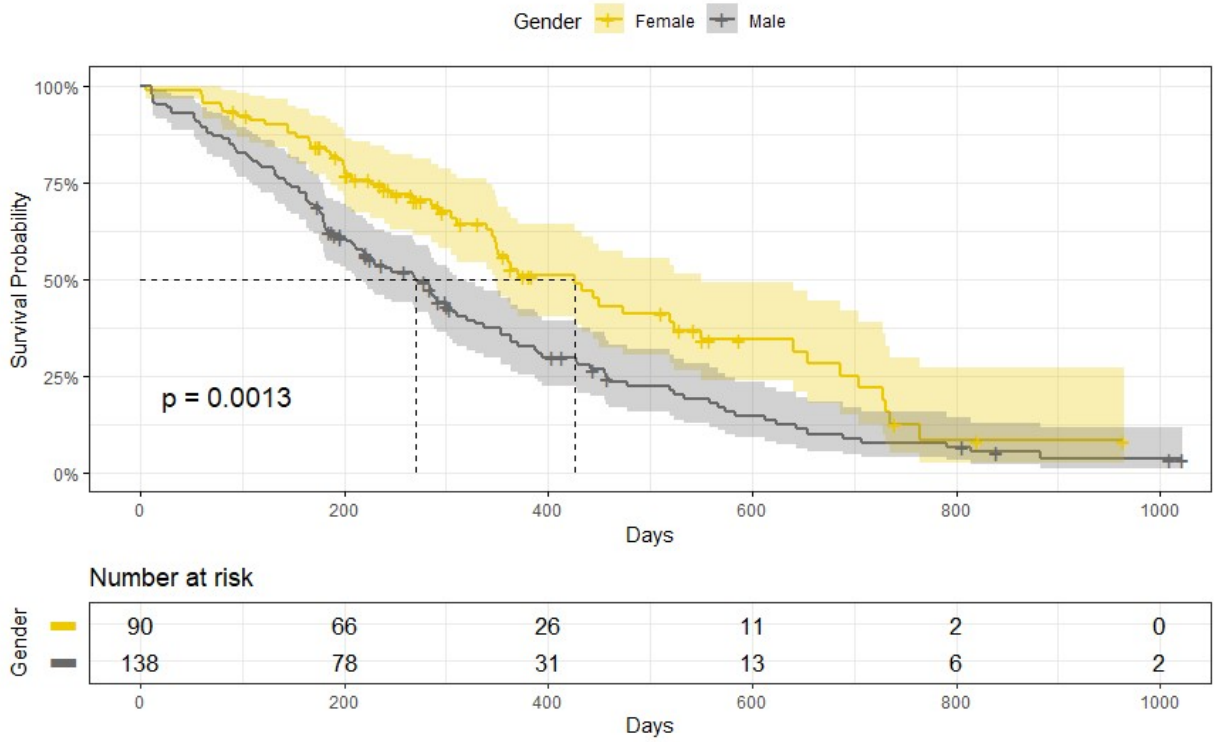
First of all, to make the plot more informative and elaborate we can add:

- a risk table (set the `risk.table` argument to `TRUE`)
- confidence intervals for point estimates (`conf.int = TRUE`)
- p-value of the log-rank test (`pval = TRUE`)
- horizontal and vertical lines for median for each group (`surv.median.line = "hv"` where "h" corresponds to the horizontal line and "v" to the vertical. To show only one line type the argument could be set to "h" or "v" accordingly)

The rest of the arguments influence on graphs' visual appearance:

- `surv.scale` displays the survival probability in percentages
- `break.time.by` affects the survival plot and the risk table by setting the interval between ticks.
- `risk.table.y.text.col` colors the risk table annotations and `risk.table.y.text` shows colored bars instead of group names.
- `xlab, ylab, legend.title, legend.labs` arguments define labels/names for corresponding components.
- `palette` argument set colors for each group. To familiarize yourself with colors and color names in R I recommend looking at the R color cheatsheet.
- `ggtheme` argument is used to set a graph theme. The `theme_bw()` is a standard R plot theme which makes white background, draws black lines around the plot and risk table and grey lines inside the plot.

```
ggsurvplot(
  surv_fit2,                     # survfit object with calculated statistics
  risk.table = TRUE,             # show risk table
  conf.int = TRUE,               # show confidence intervals for point estimates of KM curves
  pval = TRUE,                   # show p-value of log-rank test.
  surv.median.line = "hv",       # draw horizontal and vertical lines for median
  surv.scale = "percent",        # display probability in the y-axis in %
  break.time.by = 200,           # break x-axis in time intervals by 200.
  risk.table.y.text.col = TRUE,  # color risk table text annotations.
  risk.table.y.text = FALSE,     # show bars instead of names in text annotations
  xlab = "Days",                 # x-axis label
  ylab= "Survival Probability",  # y-axis label
  legend.title = "Gender",       # legend title
  legend.labs = c("Female", "Male"),# legend names
  palette = c("gold2","grey41"),    # color palette
  ggtheme = theme_bw()           # plot theme
)
```

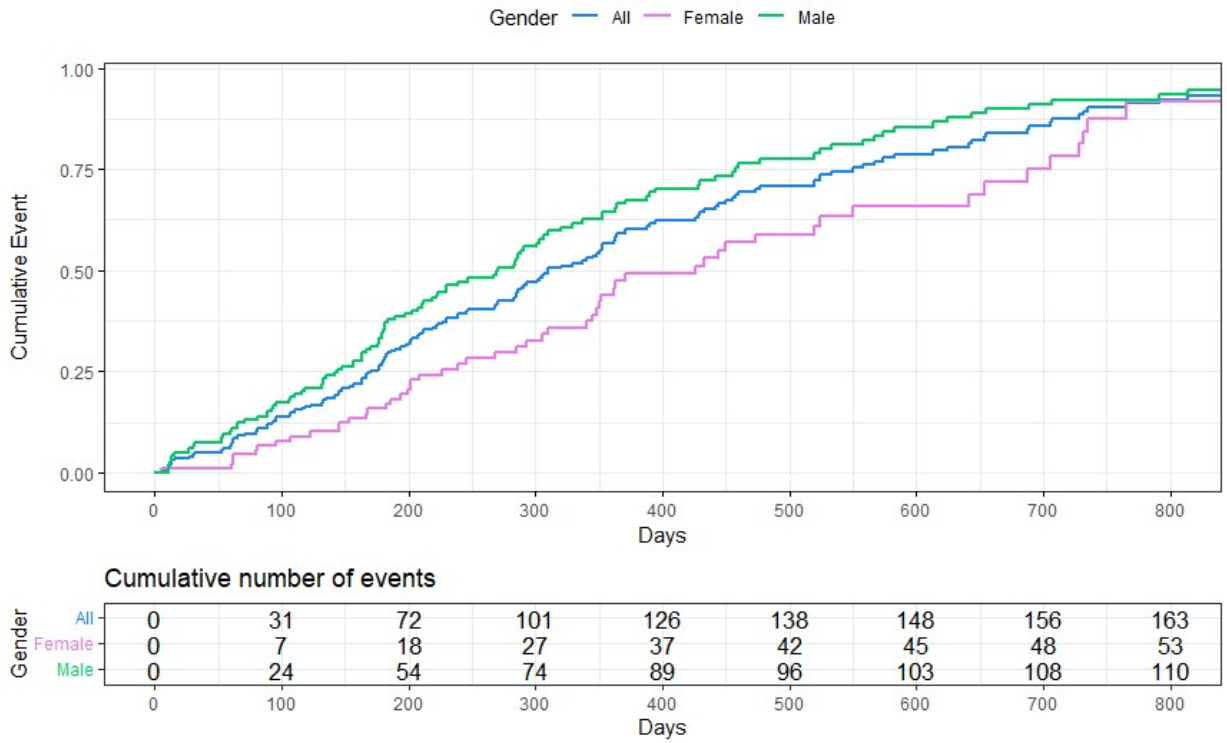


Another thing that `ggsurvplot()` allows doing is to transform the survival curves. The graph transformation can be specified by using a powerful parameter called “fun”. It can be set to “event” which means that the graph will plot the cumulative events (f(y) = 1-y) instead of survival probability, “cumhaz” plots the cumulative hazard function (f(y) = -log(y)) and “pct” for survival probability in percentages (which is similar to surv.scale = "percent" that was shown before). Here is the description of the rest of the arguments:

- `add.all` argument adds overall group
- `cumevets` show a table for the cumulative number of events
- `censor` – draws/removes censor signs on curves
- `xlim` argument defines borders of the x-axis. The important fact is that borders are changed without affecting survival estimates.

The remaining parameters were already described higher.

```
ggsurvplot(
  surv_fit2,                    # survfit object with calculated statistics
  fun = "event",                # defines transformation for survival curves
  add.all = TRUE,               # adds cumulative group
  cumevents = TRUE,             # shows table for cumulative number of events
  censor = FALSE,               # removes censor signs from the graph
  xlim = c(0,800),              # present narrower x-axis, without affecting survival estimates
  break.time.by = 100,
  xlab = "Days",
  ylab= "Cumulative Event",
  legend.title = "Gender",
  legend.labs = c("All","Female", "Male"),
  palette = c("dodgerblue2", "orchid2","springgreen3"),
  ggtheme = theme_bw()
)
```



So, we can see the graph which shows cumulative events. The overall group was added as well as the table of a cumulative number of events. Censor signs were removed and the x-axis is now spreading till 800 days (not 1000 as on the previous graph).

Consequently, we can see that survminer package is really a powerful tool for the visualization of Kaplan-Meier curves. Produced graphs look beautiful and attractive even without specifying arguments related to the visual appearance. Customizing graphs in various ways requires just a few lines of code, minimum R knowledge and is very straightforward. Of course, it is not the only thing the package can do. The survminer package also can plot graphs of the scaled Schoenfeld residuals, diagnostics graphs presenting goodness of Cox Proportional Hazards Model, forest plots for a Cox regression model and Adjusted Survival Curves for Cox Proportional Hazards Model.

HOW TO CREATE AN INTERACTIVE KAPLAN-MEIER PLOT IN R

It is absolutely clear that R usage is growing in the clinical trials industry. We see various initiatives like specific packages for the clinical trials industry or different R Shiny applications or dashboards that could help investigators to explore data or amend outputs to obtain answers much faster. Despite the fact that R has various elaborate packages for data visualizations even those that are specified on just one type of graph (such as `survminer`), R also provides the possibility to make graphs interactive. To share such graphs they can be inserted into R Shiny applications or simply opened in the web browser. Interactive graphs provide an ability to put more information on the plot which means that the number of questions it can answer increases. Interactivity gives you an opportunity to highlight the key points which trigger deeper explorations. In this article, we will demonstrate two packages for interactive data visualizations in R: ggplotly and highcharter.

GGPLOTLY PACKAGE

Plotly is a free and open-source graphing library for R. It creates interactive web-based graphs via the open-source JavaScript graphing library plotly.js. Plotly package provides two ways of creating an interactive graph:

- Set up the plotly object using the `plot_ly()` function.
- Get the plotly object by transforming the ggplot2 graph

We will focus on the second way because it is very easy and just amazing. The `ggplotly()` function interacts with ggplot2 (a well-known R package for data visualization in R) graphs. It simply takes the ggplot2 graph and makes it interactive. Unfortunately, it doesn't work with survminer package. So to produce an interactive plot using plotly firstly you need to create the plot using the ggplot2 package. But thanks to R and its extraordinary number of qualitative packages and extensions. One of them is called GGally – ggplot2 extension which facilitates the construction of

certain types of graphs (by diminishing the number of combining geoms) and one of them is a survival plot. The necessary function from the GGally package is `ggsurv()`, it produces Kaplan-Meier plots using the ggplot2 package. Let's gather all the information together and create an interactive graph.

To do this we will need two packages: plotly and GGally. Using the `ggsurv()` function we create a standard Kaplan-Meier plot:

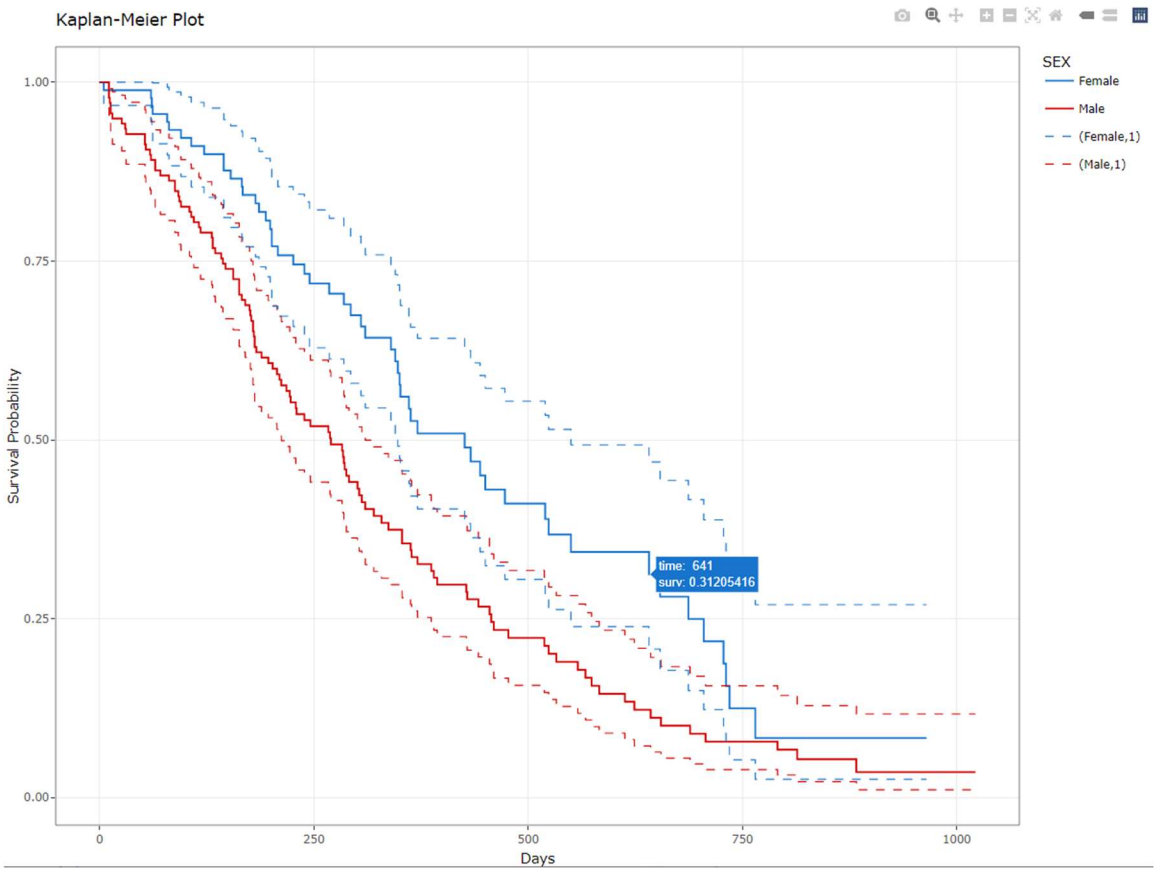
- The first and necessary argument is the survfit object (called `surv_fit2` in the example).
- `CI` argument specifies whether to plot the 95% CI.
- `plot.cens` argument defines whether to mark censored observations
- `surv.col` sets the colors of survival curves
- `size.ci` changes the width of 95% CI
- `ylab, xlab, main` – set the axis names and plot title respectively

In the example, we saved the graph into the ggplot2 object `int_kmplot`. Once the plot is created the next thing you need to do is just call the `ggplotly()` function and the interactive Kaplan-Meier plot is created.

```
library(plotly)
library(GGally)

int_kmplot <- ggsurv(surv_fit2,                    # survit object
             CI = TRUE,                            # plot CI or not
             plot.cens = FALSE,                    # marks censored observations
             surv.col = c("dodgerblue3","red3"),   # color of survival estimate
             size.ci = 0.3,                        # line width of the 95% CI
             xlab = "Days",                        # x-axis label
             ylab = "Survival Probability",        # y-axis label
             main = "Kaplan-Meier Plot") +         # plot title
  theme_bw()                                       # standard plot theme

ggplotly(int_kmplot)
```



Let's discuss the functionality that plotly package gives by default. First of all the pop-up appear once you point to any place on the graph with your mouse. The pop-up contains the exact values for x and y-axis. Of course, this pop-ups can be improved. The next thing is that you can deactivate lines by pressing on the legend. For example, if you don't need the CI interval, just press on the "- - " line in the legend and it will disappear. In the top right corner of the graph there is a menu which allows interactive manipulations with the graph: you can rescale the plot, zoom in and out, move the plot using the "Pan" button, make a screenshot of your insight and then revert back to the original plot by pressing the "Reset axes" button.

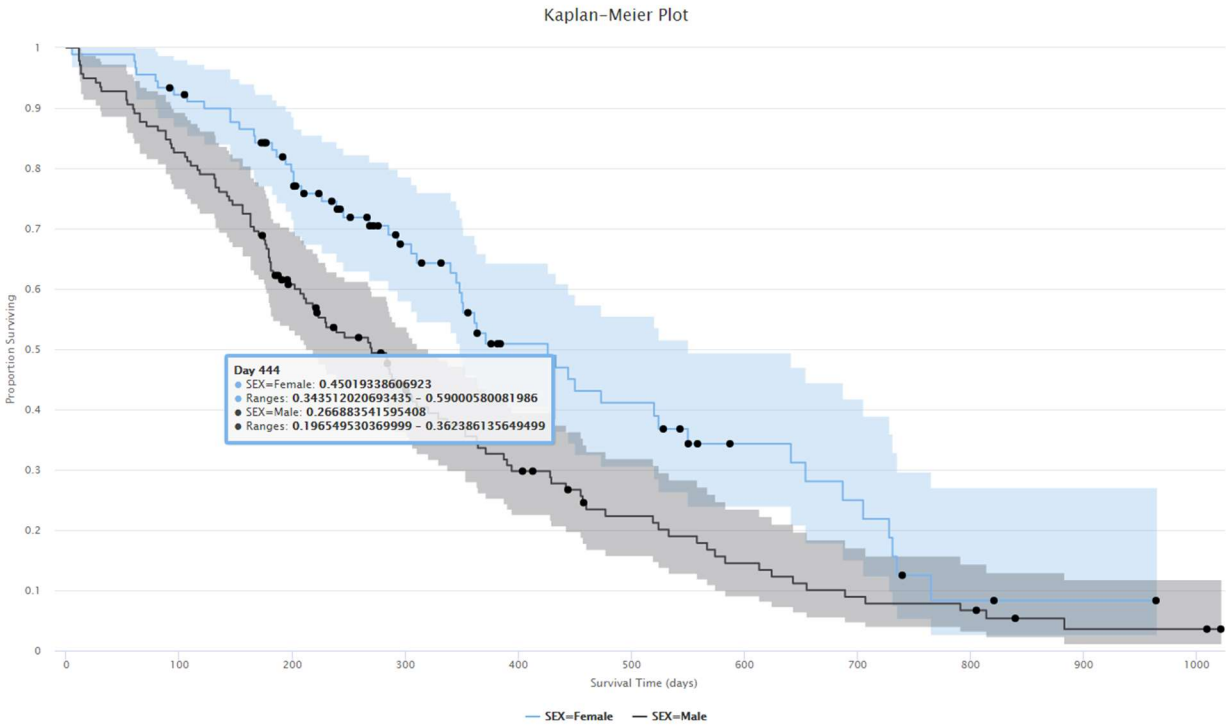
HIGHCHARTER PACKAGE

Highcharts is a modern multi-platform charting library written in pure JavaScript. Highcharts has main wrappers for several programming languages such as Python and R. Its wrapper for R is called highcarter. The main advantage of highcharts is its beautiful graphics, but is not free for commercial use.

To create a Kaplan-Meir plot using highcharter package you need to use `hchart()` function. As in the other packages you simply need to specify the survfit object and the graphs is ready. Also, we've set the ranges argument to TRUE to show the 95% CI. The highcharter package allows to use the pipe "%>%" operator to enhance the graph. In our example, the `hc_tooltip()` function is used to change the tooltip border width and then add a word "Day" into a header of a pop-up window. The `hc_title()` and `hc_xAxis()`, `hc_yAxis()` were used to set up title and axis names.

```
library(highcharter)

hchart(surv_fit2,
       ranges = TRUE) %>%
  hc_tooltip(headerFormat = "<b>Day {point.key} </b> <br>",
             borderWidth = 4) %>%
  hc_title(text = "Kaplan-Meier Plot") %>%
  hc_xAxis(title = list(text = "Survival Time (days)")) %>%
  hc_yAxis(title = list(text = "Proportion Surviving"))
```



The main advantage of highcharts packages in term of interactivity is its tooltip. It can be customized in various ways and show a lot of information. On this example it shows the day (i.e. x-axis value) and survival probability for each group with ranges. Even the default highcharter graph looks amazing due to its graphics. The paper version probably couldn't show it, but you will definitely see it if you have a chance to work with highcharts For more information about plotly and highchart please check my Phuse paper called "Make you data come alive".

CONCLUSION

Survival analysis plays crucial role in the clinical trial industry. Kaplan-Meier is the most frequent survival analysis method used in this field and usually is the first point of analysis of a clinical study. Commonly Kaplan-Meier estimates are used to visualize the treatment effect, so it is an essential skill for data analyst to produce such type of plots. In this paper we discussed the wide range of R packages that provide the ability to produce elaborate, detailed and presentation-quality graphs. Since R is used more frequently in clinical trials, any statistical programmer would benefit from using it or at least knowing its capability. The biggest advantage of R is that its syntax is very intuitive and allows to produce high-level outputs from literally right away. Get started using R and benefit from it.

RECOMMENDED READING

- Babych Oleksandr, 2020, PHUSE Conference (EU Connect) – "Make you data come alive".
- Babych Oleksandr, 2019, "The Power of Data Visualization in R"
- Batra, Neale, et al. The Epidemiologist R Handbook. 2021. (https://epirhandbook.com/)
- Emily C. Zabor, "Survival Analysis in R". (https://www.emilyzabor.com/tutorials/survival_analysis_in_r_tutorial.html)
- Survminer cheatsheet (https://rpkgs.datanovia.com/survminer/survminer_cheatsheet.pdf).
- Carson Sievert, 2019, "Interactive web-based data visualization with R, plotly, and shiny". Available at https://plotly-r.com/index.html
- Joshua Kunst, 2020, "First steps on highcharter package" (https://jkunst.com/highcharter/index.html)
- R color cheat sheet: https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Oleksandr Babych
Company: Intego Group LLC
Address: Grzybowska 78, 00-844 Warsaw, Poland
Email: oleksandr.babych@intego-group.com, oleksandr.babych1@gmail.com
Web: www.integogroup.com