PP12

SAS!...R(aaah), Saviours of the Data Vis

Sophie Shapcott & Jack Finch | sophie.shapcott@gemprogramming.com & jack.finch@gemprogramming.com

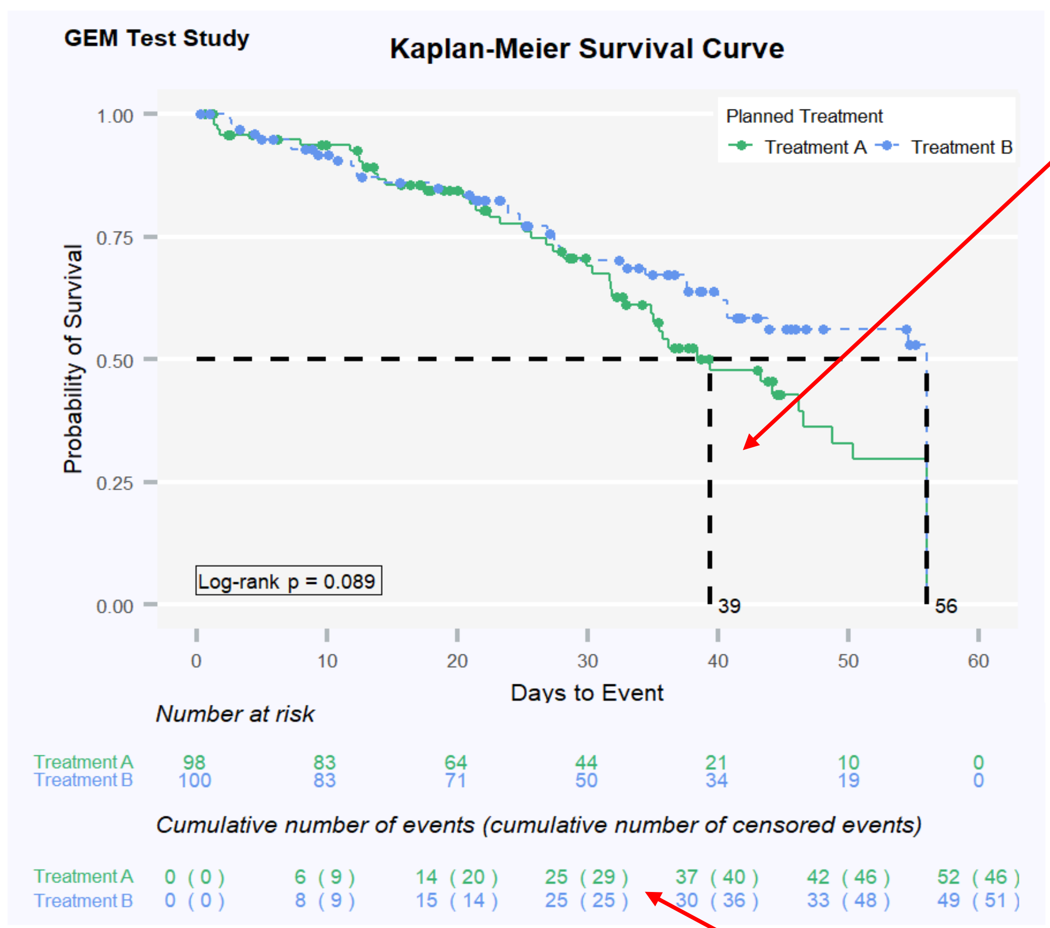
Overview

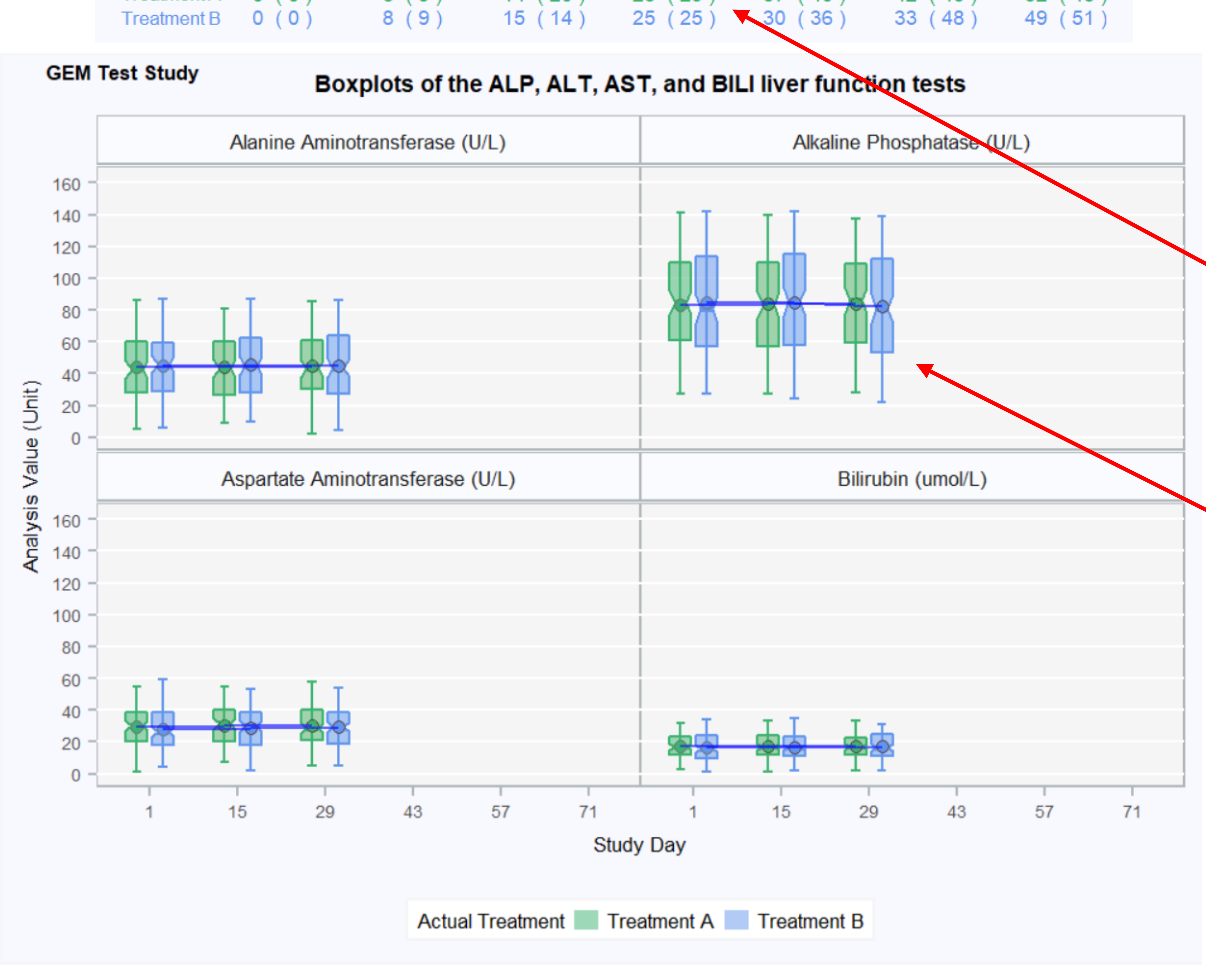
R and Python are words we hear being uttered more and more within the pharmaceutical programming community. But what are the important differences between SAS® and these open-source software packages when choosing how to visualise your data?

SAS is the de facto software for the Pharmaceutical Industry. It provides a comprehensive validated framework with several user-friendly interfaces. There is, however, an increasing interest in open-source software, such as R and Python. Many pharmaceutical organisations are now accelerating discussions around the use of open-source software packages. This poster compares the use of SAS versus R, focusing on data visualisation and graphic capabilities.

An animated lab boxplot and an annotated Kaplan-Meier (KM) plot will be generated using SAS and R to compare ease of use, appearance, annotations, and animation. The SAS version used is 9.4M6 and the R version is 4.1.0.

R





To produce the median droplines in R, all that is needed is the following argument in the `ggsurvplot` function: `surv.median.line="hv“`, where hv = horizontal and vertical.

To get the median survival times from PROC LIFETEST use the following:

```
ods output quartiles=quartiles;
```

Create variables in the dataset with these values and include this statement in the PROC SGPLOT:

```
text x=xlabel_a y=ylabel text=lab_a /
textattrs=(size=8);
```

Use this statement to get the values for the log-rank p-value with α=0.05:

```
ods output homtests=tests;
```

Create a macro variable with this p-value:

```
proc sql noprint;
  select PROBCHISQ into: pval trimmed
  from tests;
quit;
```

Then include the following statement in the PROC SGPLOT:

```
inset "Log-rank p=&pval." /
position=bottomleft textattrs=(size=12)
border;
```

Using the data from the PROC LIFETEST when the following option is added:

```
(atrisk=0 to 60 by 10)
```

add in this statement to the PROC SGPLOT:

```
xaxistable atrisk / x=tatrisk class=stratum
colorgroup=stratum attrid=trtmt
location=outside;
```

The number of events are concatenated using:

```
surv_plot$data.survtable$cum.n.event <-
paste(surv_plot$data.survtable$cum.n.event,
" (“,surv_plot$data.survtable$cum.n.censor,
")“).
```

The following piece of code creates the table:

```
event_tbl <- ggsurvtable (fit=
surv_plot$data.survtable, survtable =
"cumevents", ggtheme=theme(...),
tables.theme=theme(...), ...)
```

Select the variable to base the animation on:

```
animated_plot <- stationary_plot +
transition_reveal(ADY, keep_last = TRUE)
```

Animate the plot and save as a GIF with the following function:

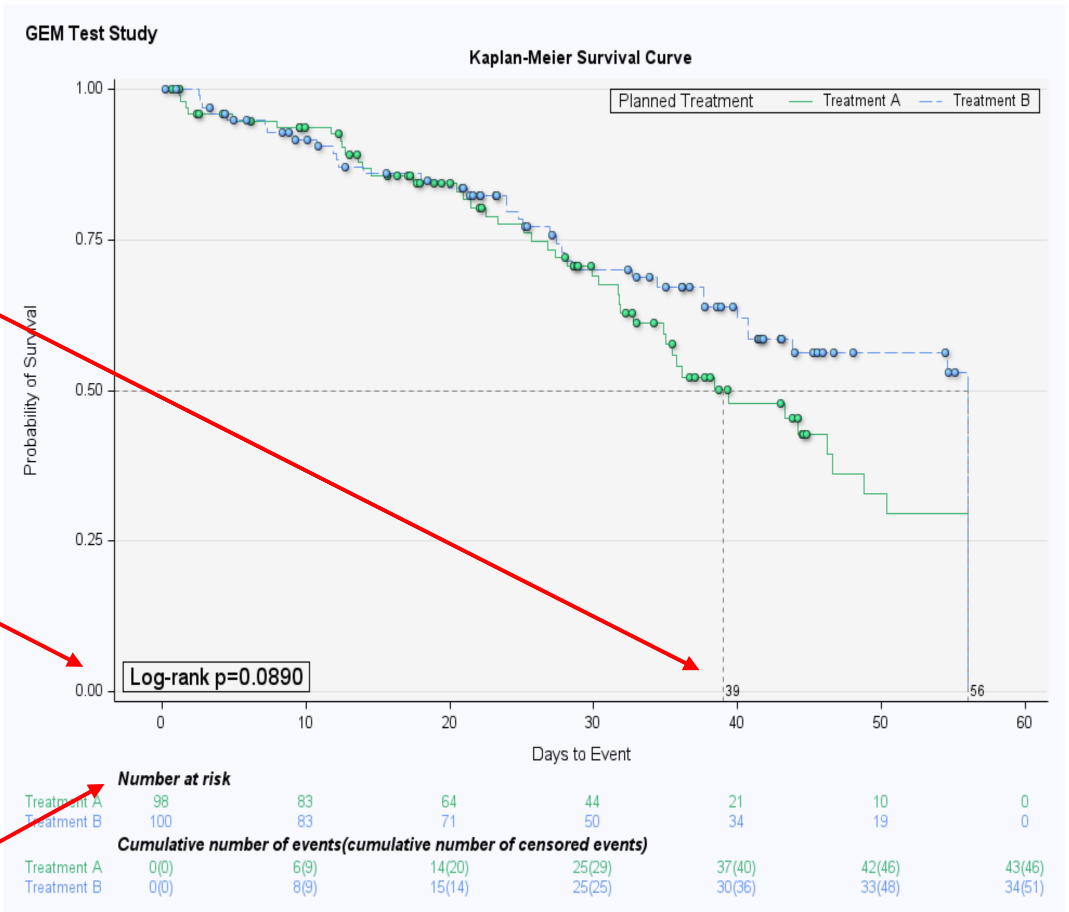
```
animate(animated_plot, nframes=6,
fps=0.5, renderer =
gifski_renderer("anim_bxplt_poster.gif"),
height=7.5,  width=10, units="in", res=200)
```

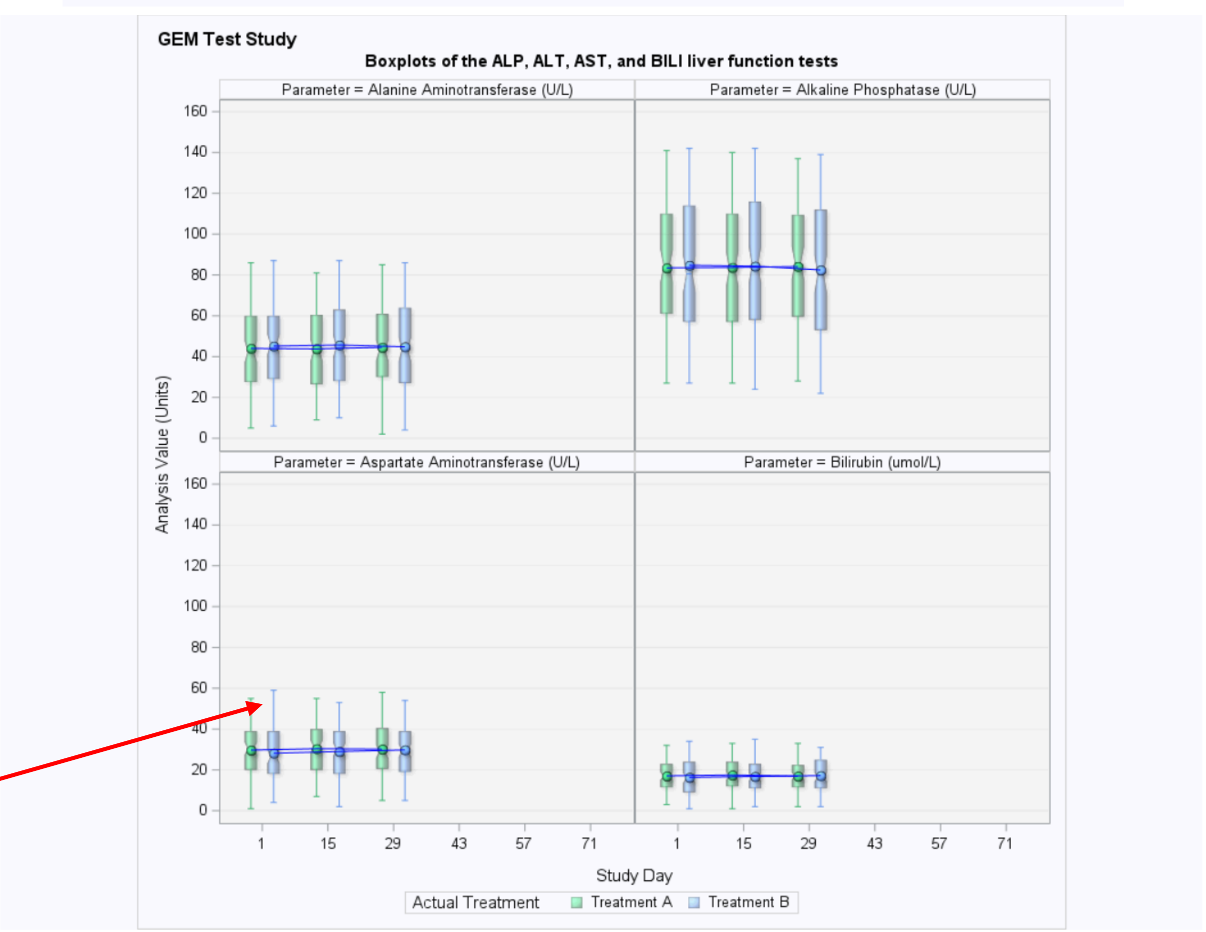
Use the following options:

```
printerpath=gif /*GIF universal printer*/
animduration=1 /*1 seconds between frames*/
animloop=yes /*Play continuously*/
Noanimoverlay /*Display graphs
sequentially*/
```

With an ODS PRINTER statement, the animation is started and stopped with an `animation=start/stop` option.

SAS





EASE OF USE:

- **SAS**: The two procedures, `proc sgplot` and `proc sgpanel` enable users to create a variety of plots simply by changing the statements within. For example, `vbox` and `panelby` with `proc sgpanel` is used to obtain the boxplots and `scatter`, `step`, and `xaxistable` along with `proc sgplot` is used to obtain the KM plot.
- **R**: Using the *ggplot2* package, creating any plot is a case of building 'layers' on top of the 'base' defined using the `ggplot` function. For example, to obtain the stationary boxplots, five additional layers were required: `geom_boxplot` (for the basic boxes), `stat_boxplot` (for the whiskers with caps), `geom_point` (highlighting the mean for each box), `geom_line` (joining up the mean points), and `facet_wrap` (a separate subplot per lab test). The KM plot could be created using a similar method however the package *survminer* gives the capability to produce the curves, certain tables, p-value, and median lines just by setting different arguments in the function `ggsurvplot`.

ANIMATION (Boxplots Only):

- **SAS**: The animation of these boxplots in SAS required a couple of extra steps on top of a normal program. Firstly, a variable was created based on the values of the x-axis tick marks called 'build_day'. To retain the previous boxes at later frames the data needed to be duplicated for each previous value of the build_day variable. For example, the second frame consisted of both Day 1 and Day 15 data. Next, system animation options needed to be applied and an `ods printer` statement was used to generate a .gif file. This means it isn't a straightforward job to adapt an existing figure's program to produce an animated GIF compared to R.
- **R**: The package *gganimate*, which is an extension to *ggplot*, makes the animation of plots easy to do with the addition of only a couple of functions to the static plot. The first is an additional 'layer' to the plot which identifies the variable to animate across and the second is the function `animate`. This renders the resulting plot to whichever format is required with many options available. The manual 'coding' of the animated variable needed in SAS is achieved 'behind the scenes' as part of the *gganimate* package.

Discussion

ANNOTATION (KM Plots Only):

- **SAS**: A standard KM plot in SAS can be produced relatively easily but to further customise can be a challenge. To get the median values for reference lines, another ODS statement was needed - `ods output quartiles=<dataset name>;`. These median values were then used as coordinates to get the label correctly positioned near the reference line on the x-axis using a `text` statement. These median values were also used in `dropline` statements to get the reference line in the required location. The p-value was obtained using the ODS statement `ods output homtests=<dataset name>;`. This was then placed on the plot using an `inset` statement within the SGPLOT procedure. There are multiple options for positioning and formatting of text within graphs which fall outside the scope of this poster.
- **R**: Annotations such as the p-value, median droplines, and risk tables can be easily added to a KM plot when using the *survminer* package. It is just a case of changing the values of certain arguments within the `ggsurvplot` function. However, in order to add in additional information, such as the median values and the bottommost table, further work is required. For example, in order to obtain the data for the second x-axis table in the correct format, the output produced from the `ggsurvplot` function had to be manipulated. This then meant that the final plot and two tables had to be aligned back together using the *gridExtra* package.

APPEARANCE:

- **SAS**: For both of the plots in SAS, the visual attributes were mapped to treatment groups using a discrete attributes map, like the boxes in the box plots. For the KM plot, the attributes map controls the colour and style of the lines and markers, as well as the x-axis table values. The option, `dattrmap=<dataset name>` is used on the procedure statement. Also, on each statement within the plotting procedure the `attrid=<variable name>` option is used to point SAS to the correct variable used to map the attributes.
- **R**: Within the *ggplot* package, there is a `theme` function which can be used to specify background colours, text families, legend size, etc. If this is saved as an R object (as done here), then it can be added to the end of each graph definition in order to obtain consistency across all plots in a study. Furthermore, functions such as `scale_fill_manual` and `scale_y_continuous` can be added to the plot definition in order to customise the defaults with company colours, titles, legends etc.

Conclusions

- Overall, both R and SAS offer the capability to readily produce complex, graphical visualisations of data. In some cases, R may provide a more apparent method of customisation, e.g. the KM plot droplines.
- Each of these programming languages has the ability to produce high quality customised plots, with consistent themes and animation capabilities.
- Historically, a potential drawback for R in the pharmaceutical industry, is the validation effort required for each package used (in this case, around seven packages). This topic is under much discussion across the industry, in particular with groups such as the *R Validation Hub* (https://www.pharmar.org/about/).
- A direction worth investigating is the integrated use of SAS and R, for instance via SAS/IML®, where each language is being used to its strengths. GEM will investigate this and present our findings at a future date.

Scan the QR code for high-res plots and program code.