VALIDATION OF GRAPHICAL OUTPUT – SUPERIMPOSED WAY USING SAS OR R

Niraj Pandya

Sumptuous Data Sciences

Introduction / steps

SUMPTUOUS
DATA SCIENCES

- Production side generates graphical output (i.e. PNG file) as per specs
- Validation side also generates a version using complementary colors to production side
- Validation side adjusts the size and transparency of production output
- Validation side overlays transparent production output on top of validation side output
- Validation side inspects overlaid figures for matches and mismatches
- Matching data points will appear in grayscale

Complementary Color Theory

- Two colors that cancel each other's hues when mixed
- Produces grayscale color
- Black – complete absence of hue White – complete presence of all hues
- SAS has 8 color schemes – Widely used RGB scheme - CX*rrggbb*
- R, G and B components are presented by hexadecimal numbers
- Ranges from 00-FF (0% to 100%) of the respective component hue
- For any RGB color, complementary color can be obtained by subtracting respective R, G and B component's hex number from hex number FF

Complementary Color Theory

- Example:

  Red = CXFF0000 

  Complementary color = CXFFFFFF (White) – CXFF0000 (Red) = CX00FFFF (CYAN) 

  Yellow = CXFFFF00 

  Complementary color = CXFFFFFF (White) – CXFFFF00 (Yellow) = CX0000FF (Blue) 

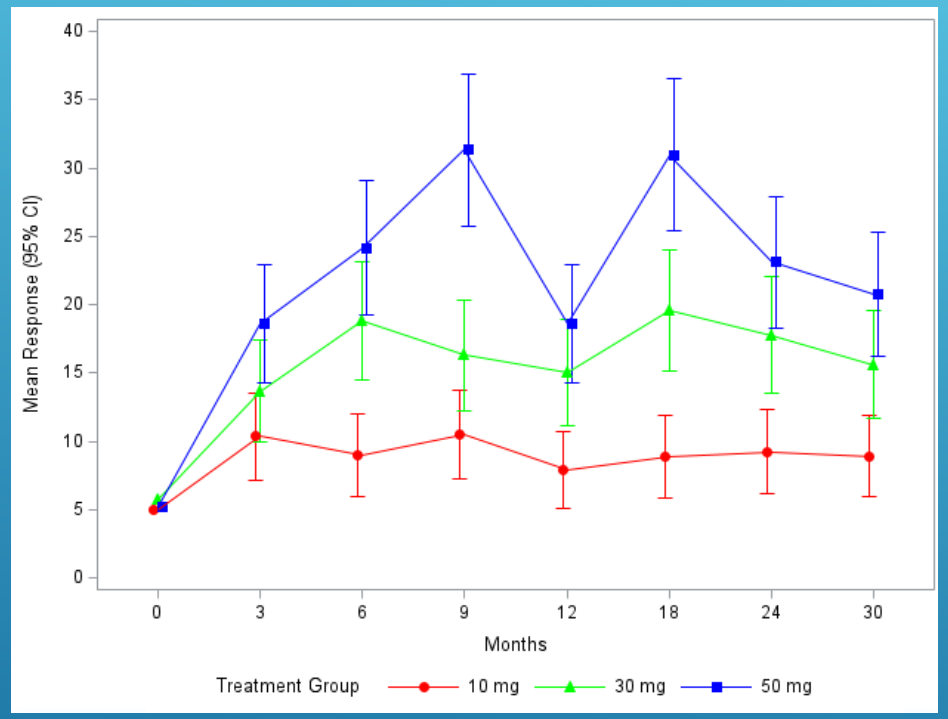
Color Scheme from SAS/GRAPH

SUMPTUOUS
DATA SCIENCES

Basic Hues

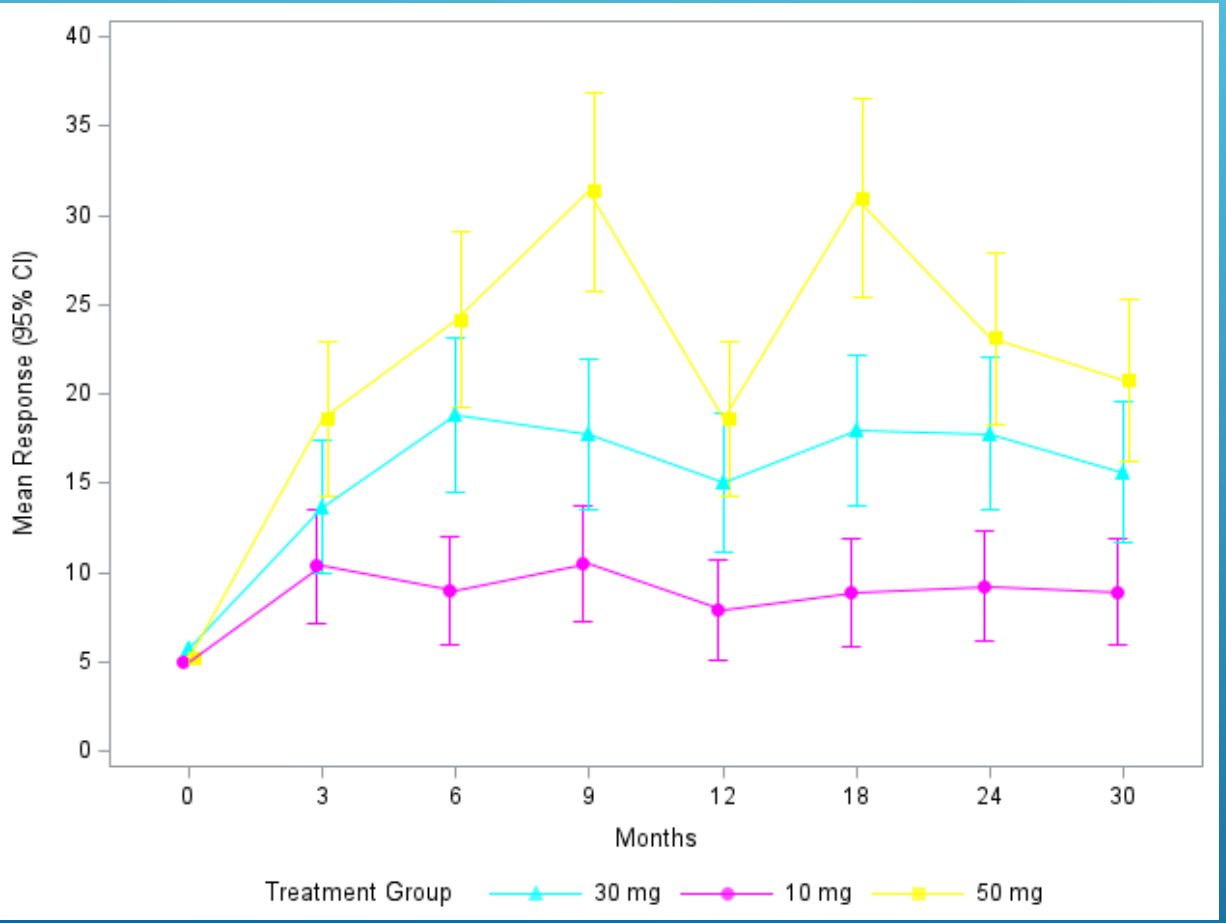
Name	Description	RGB	Sample
BLACK | BL	black	#000000	
BLUE | B	blue	#0000FF	
BROWN | BR	brown	#A05000	
CHARCOAL	charcoal	#4F4F4F	
CREAM	cream	#E8D898	
CYAN	cyan	#00FFFF	
GOLD	gold	#FFAA00	
GRAY | GREY | GR	gray	#808080	
GREEN | G	green	#00FF00	
LILAC	lilac	#E06090	
LIME	lime	#C0FF81	
MAGENTA	magenta	#FF00FF	
MAROON	maroon	#700000	
OLIVE	olive	#2A8307	
ORANGE | O	orange	#FF8000	
PINK	pink	#FF0080	
PURPLE | P	purple	#703070	
RED | R	red	#FF0000	
ROSE	rose	#FF6060	
SALMON	salmon	#FF0055	
STEEL	steel	#3883A8	
TAN	tan	#E0A860	
VIOLET	violet	#B090D0	
WHITE | WH	white	#FFFFFF	
YELLOW | Y	yellow	#FFFF00	

Production Output – Line plot

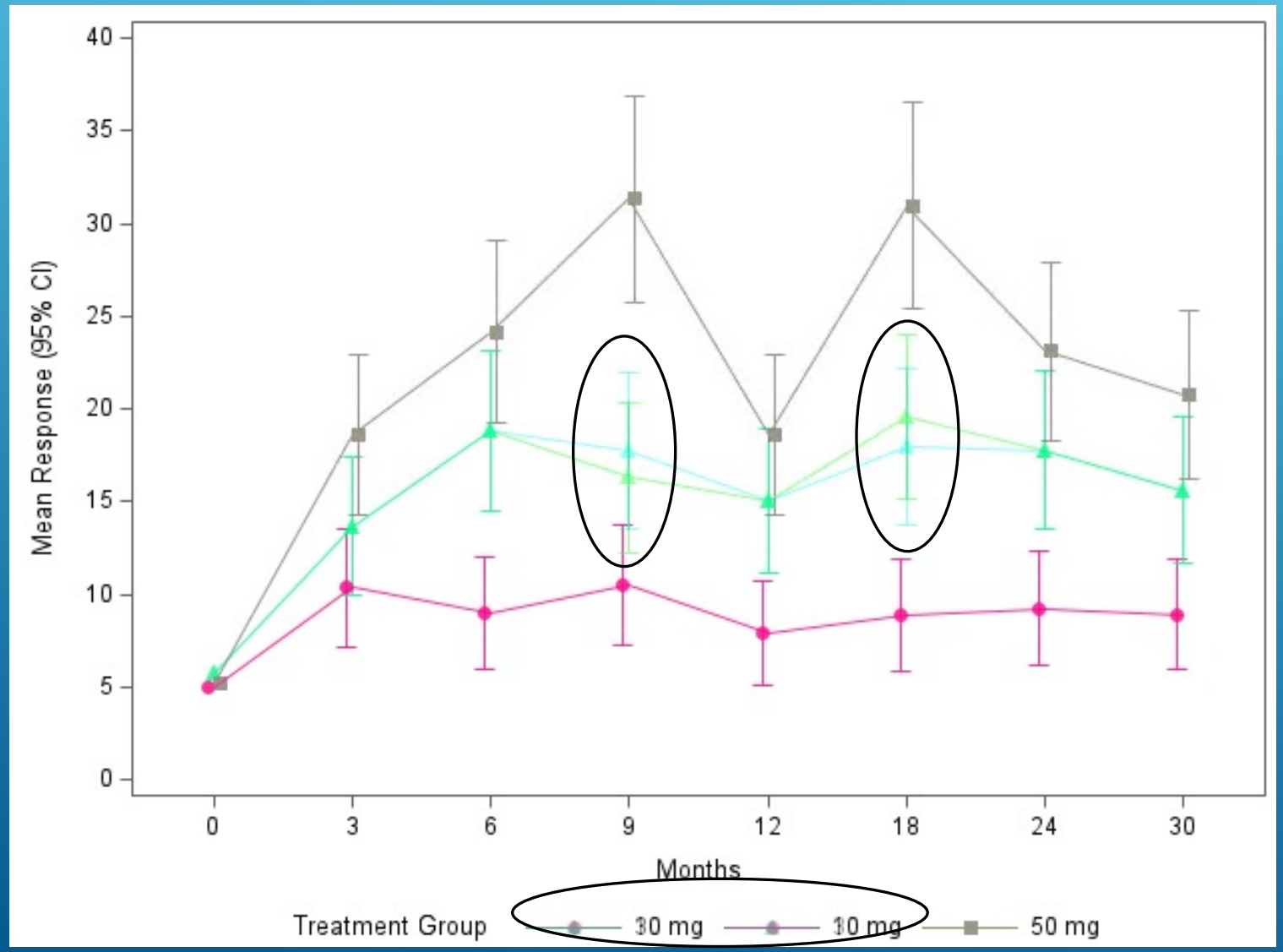



```
proc template;
      define statgraph temp_name;
        begingraph;
          layout ......;

          endlayout;
        endgraph;
      end;
run;

ods graphics / imagefmt=png imagename="gtlimg" noborder;
ods listing gpath="location";

proc sgrender data=data_name template=temp_name;
run;
```

Validation Output (with complementary colors) – Line plot

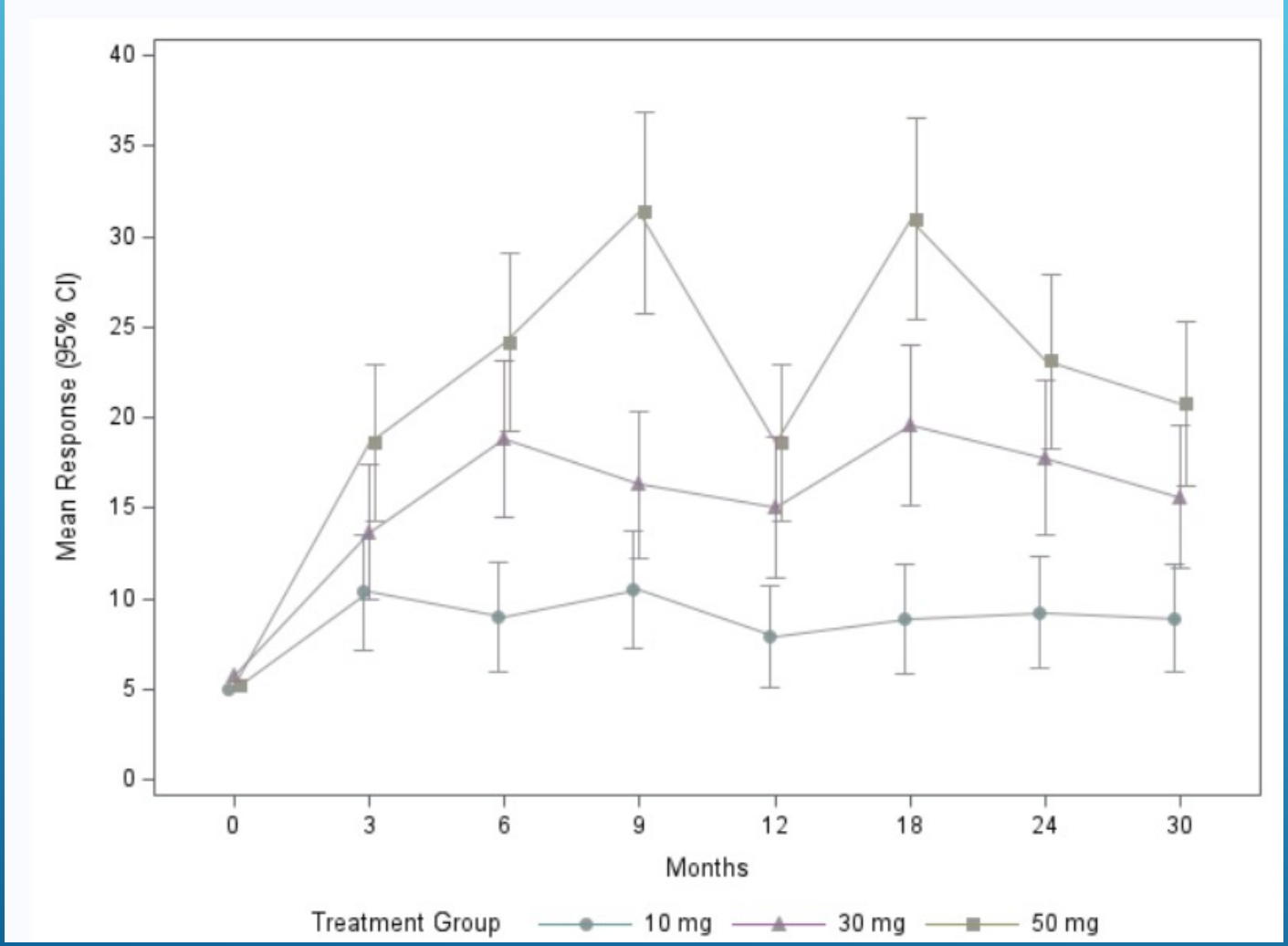


Validation Output with overlay from production output – Line plot




```
proc template;
      define statgraph vtemp;
        begingraph;
          layout ......;
            seriesplot x=xx y=yy
datatransparency=0.3;
            drawimage "location\filename" / x=0
y=100 anchor=topleft drawspace=graphpercent
transparency=0.6;
          endlayout;
        endgraph;
      end;
run;

proc sgrender data=data_name template=vtemp;
run;
```

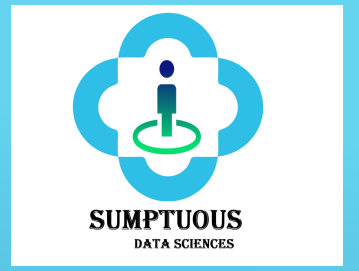
Validation Output with complete match



Notes:

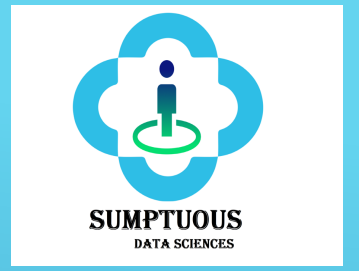
- Use of GTL
- Use of SGPLOT and SGPANEL
- In addition to complementary colors, both figures need to be transparent
- DATATRANSPARENCY = 0.3 needed
- Can be used for grouped data as well
- Complementary colors hex code can be stored in macro variables so make process more dynamic
- Can be used for grayscale production outputs too
- Shadowing effect with grayscale production output – use any color from non grayscale spectrum

R for Figure Comparison



- Many R packages that can read in generated output
- Extract the figure from output .rtf file
- Compare the pixel values within the figures
- imager package – based on Cimg, a C++ llibrary
- It provides API (Application Programming Interface) for image processing
- imager replicates Cimg functionalities and interface
- It is recommended to use image file formats of .png or .jpg

Extract image from .rtf file

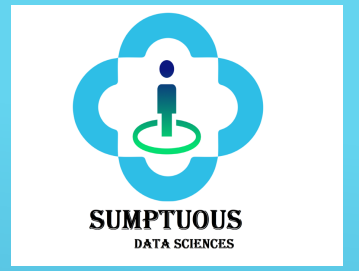


- Figures or plots in .rtf file are stored as binary code
- Read in .rtf file in R using readLines function
- Isolate binary code related to figure and convert to byte list
- Once byte list is extracted, writeBin function is used to convert byte list in to .png/.jpg file.
- This .png/.jpg file is used as input to imager package

Extract image from .rtf file

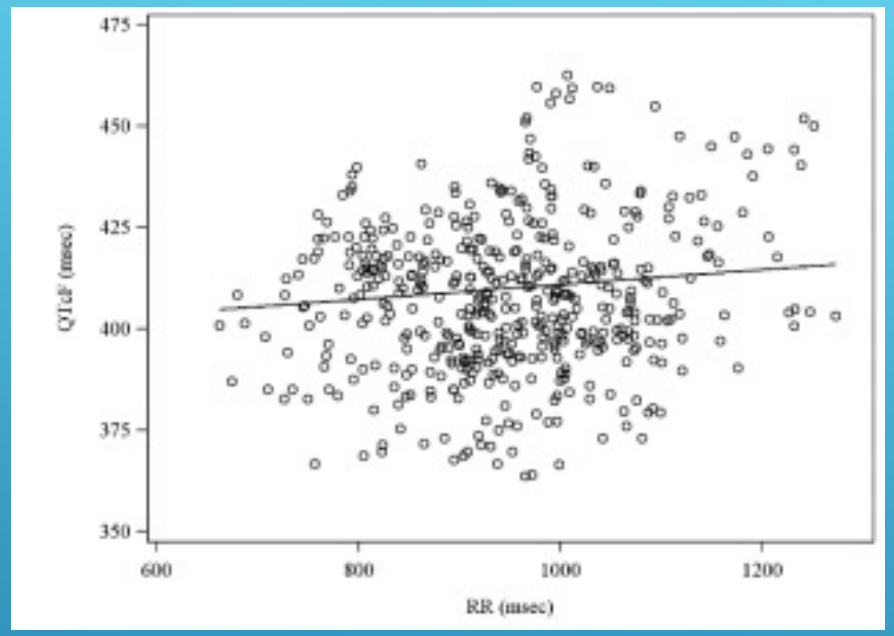
```
extract_rtf_png <- function(rtfFile){
 #read in RTF
 myrtf <- readLines(rtfFile)

 blist <- c()
 ind <- 0
 for (str in myrtf){
       if (ind){
          if (length(grep("\\}\\}\\}",str))){
             break
           }
          #split string 2 character
          sst <- strsplit(str, "")[[1]]
          list <- paste0(sst[c(TRUE, FALSE)], sst[c(FALSE, TRUE)])
          blist <- c(blist,list)
          }

         if (length(grep("pict",str))){
            ind <- 1
       }
  }
 bytelist <- as.raw(as.hexmode(blist))
 return(bytelist)
}
```

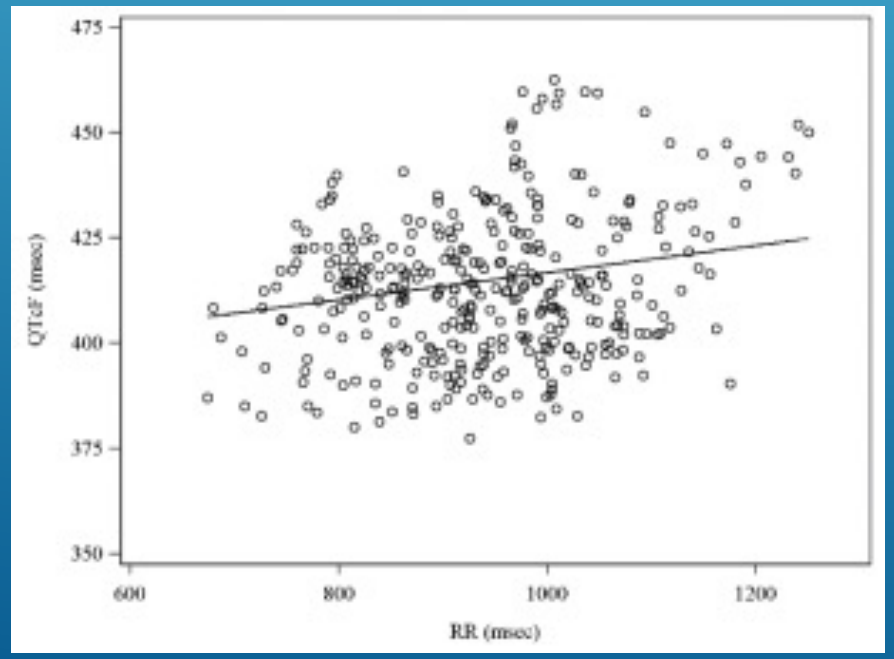
Read in Image and compare

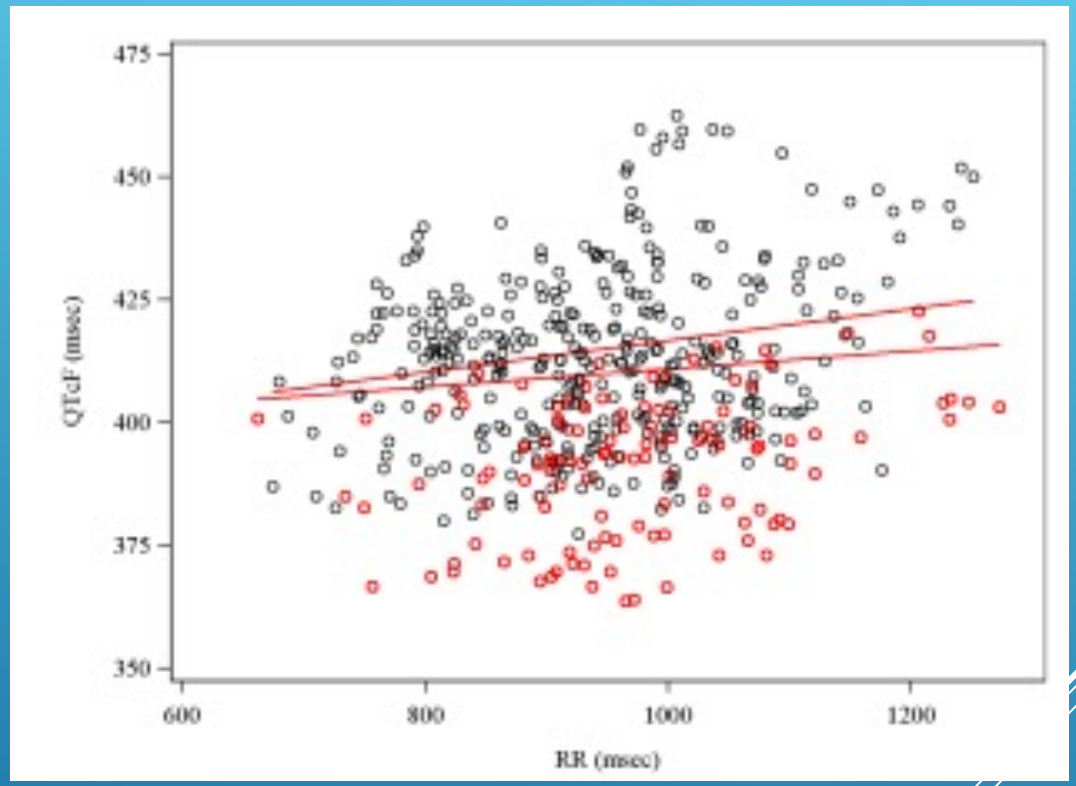


- Powerful image processing function available in imager package
- load.image( ) function can be used to read in .png or .jpg file
- Data read in will be compared against reference image data
- as.pixset(plotA – plotB) function will return the differences between two pixset matrices
- Differences will be highlighted by superimposing the differences

```
diff <- as.pixset(plotA - plotB)

comp <- colorise(plotA, diff, "red", alpha=1)
```







SUMPTUOUS
DATA SCIENCES

Conclusion

SUMPTUOUS
DATA SCIENCES

- Traditional methods of graphical output validation are error prone
- Needs manual comparison beyond dataset comparison
- New methods reduce likelihood of human error
- Potential to increase quality of final product
- Method with R further increases the efficiency and confidence in the validation result

References

- https://support.sas.com/content/dam/SAS/support/en/books/pro-template-made-easy-a-guide-for-sas-users/62007_Appendix.pdf

- https://www.pharmasug.org/proceedings/2022/AD/PharmaSUG-2022-AD-076.pdf

- https://www.pharmasug.org/proceedings/2023/DV/PharmaSUG-2023-DV-256.pdf

SUMPTUOUS
DATA SCIENCES