

In-Text Tables Revisited: A Flexible System Pleasing All Company Departments

Mark Janssens, Tibotec (Johnson & Johnson), Mechelen, Belgium

ABSTRACT

In-text tables and graphs are used by many in our company. Unfortunately, the respective requirements differ (at least) slightly.

These differing expectations break down in the classic challenges:

- (1) How to split content from style?
- (2) How to add graphs? Vector or bitmap? Linked or embedded?
- (3) How to change the style, once an in-text file has been produced? How to merge in-text files into one document?

We present a case based answer to each of these questions, and confine to the default solution, RTF and PNG, for this abstract:

- (1) The system generates tailored RTF, as some layout and style features would not (or not easily) be adjustable using ODS.
- (2) The system produces embedded high-resolution PNG graphs. The graphs are no separate PNG files, so each in-text file stands alone when it comes to content and style.
- (3) The system allows to assemble several in-text files into one document. At that point, a renewed decision on tables, graphs, and style can still be made.

INTRODUCTION

When looking for a solution, one can start at the technical solutions available, or at the business problem itself. Both approaches have specific merits and disadvantages. The advantage of an off-the-shelf solution is its readiness. The gain is not just in development, but also in testing, roll-out, documentation, and training. On the other end of the continuum, systems which are developed in-house are attractive because they fulfill specific business needs in an efficient and elegant way. However, reality is never black and white. ODS is a technical solution allowing a lot of customization. The solution that we present is not black or white either, but compared to ODS, it is positioned much more towards the business problem it originated from.

Let us quickly outline the business specifications.

- The idea is the creation of **RTF files** that contain tables, listings, and graphs. Each file reflects a single business question, and all files are organized by domain. A given RTF table in a file may contain several sections linked to different data sets.
- It should be possible to **assemble all RTF files into one consistent document** (referred to as the master document). That document serves the Statistical, the Medical, and/or the Writing department respectively. The fact that different customers are being served has some consequences regarding content and style.
- Regarding content and depending on the department, **some RTF files are excluded from the master document**. Or perhaps, some files must always be included, but the mode of the graphs (color versus monochrome) varies with the customer we serve.
- Regarding style and depending on the department, the **style of the master document may differ**. What we say here is that, after the generation of all RTF files, the style must still be adjustable.

These specifications apply to RTF mode, but the system should be able to operate in HTML mode as well. The style specifications should take the same form in either case. Hence, when we mention 'RTF', we mean RTF and HTML. Finally, when we speak of 'tables', it is implied that this includes listings also.

Sounds like ODS? We have considered ODS to tackle this challenge. It goes without saying that ODS holds a valuable place in report generation, and we use it regularly to that purpose. But this time, we have taken another direction. The ODS drawbacks, given our setting, were the following:

- Regarding content, we mention two downsides. (1) A SAS output is basically linked to one data set. The flexibility we desired was one RTF table potentially consisting of several sections, linked to different data sets. (2) Next, tables and graphs are not always well integrated. A graph must follow directly from the produced summary table. Elegant by-processing of table sections and graphs was a concern too.
- Regarding style, the limitations are twofold. (1) A first issue is that ODS styles are not uncomplicated to modify. (2) A second problem is the permanence of the style, once a program has run and an output has been generated.

EXAMPLES

1. SIMPLE EXAMPLE

The code below produces the RTF file, pasted next to this code.

```
%init_table(
  outdir=C:\Output,
  outfile=T01 Ethnic origin (a),
  colclass_ds=data.demo,
  colclass=randgrp,
  fmt=randgrp.,
  topleft_label="n (%)",
  title="Number of subjects by ethnic origin"
);
%add_section(
  ds=data.demo,
  rowclass=race,
  param=count (percent),
  fmt=3.0-5.1,
  percent_ds=data.demo,
  percent_distinct=usubjid
);
%assemble_sections;
```

Number of subjects by ethnic origin

n (%)	400 mg bid	800 mg bid	Active Control
BLACK	17 (21.3)	15 (19.0)	7 (17.5)
CAUCASIAN / WHITE	45 (56.3)	47 (59.5)	29 (72.5)
HISPANIC	16 (20.0)	17 (21.5)	3 (7.5)
NOT ALLOWED TO ASK PER LOCAL REGULATIONS	1 (1.3)		
OTHER	1 (1.3)		1 (2.5)

2. SECOND EXAMPLE

This example stretches a bit further: another style and several other options are used; a chart is added. The introduced options are highlighted (left) and explained (right).

```
%init_table(
  outdir=C:\Output,
  outfile=T02 Ethnic origin (b),
  colclass=randgrp,
  colorder=1|2|all1|*|all2,
  colorder_label="400 mg bid"|"800 mg bid"|"
  "All TMC"|"Active Control"|"All Subjects",
  width_left=144pt,
  topleft_label="n (%)",
  title="Number of subjects by ethnic origin",
  css=C:\Styles\Presentation.css,
  outds=T02
);
%add_section(
  section_label="TOTAL",
  distinct=usubjid,
  ds=data.demo,
  rowclass=race,
  param=count (percent),
  fmt=3.0-5.1,
  out_by=-count | colorder_label="All Subjects",
  percent_ds=data.demo,
  percent_distinct=usubjid
);
%add_chart(
  hsize=3in, vsize=3in,
  title="'All TMC' versus 'Active Control'",
  title_embedded="byval",
  font_size=10pt,
  subset=colorder in (3 4),
  by=colorder_label,
  statements=%str(format race $18. percent 5.1;),
  chart=pie race / sumvar=percent value=arrow
  noheading other=0,
  colors=black white red gray silver,
  legend=label=("Ethnic origin") value=(j=left)
);
%assemble_sections;
```

colclass denotes the variable that is used for the columns. In the data set at hand, the variable 'randgrp' has values 1 (for 400 mg bid), 2 (for 800 mg bid) and 4 (for active control).

colorder specifies/limits the values for the columns of the table, and their order. The key values 'all1' and 'all2' summarize all the preceding columns (the summary columns themselves excluded).

colorder-label holds the corresponding labels.

width-left sets the width of the left most column.

css reads the Cascading Style Sheet (CSS) which will layout the RTF (or HTML) file that is generated.

outds writes the data behind the RTF table to a SAS data set.

section_label adds a label above the section.

distinct makes sure that only distinct subject IDs enter the table. The ingoing data set, data.demog, contains one row per subject already. Another effect of the option is the summary line, generated at the level of the section label (see table output below).

out_by sorts the section, in this example by descending counts in the 'All Subjects' column. As '-count' is specified, the parameter 'count' is used to sort the result in descending order (notice the minus sign). The condition at the end clarifies the column that is to be used for sorting the result (see table output below).

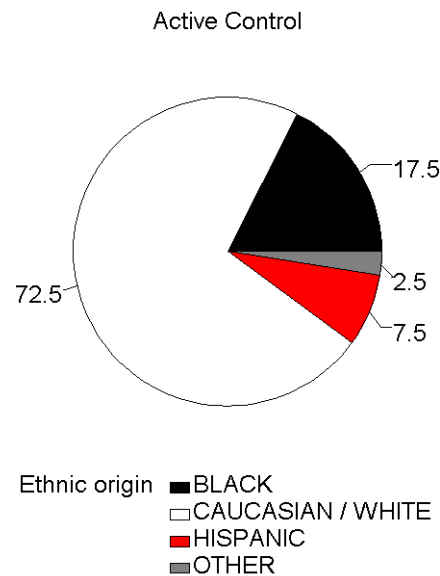
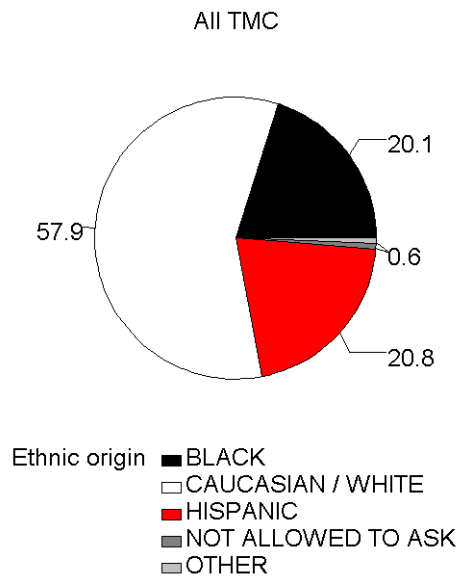
add_chart chiefly passes SAS/GRAPH statements to PROC GCHART. A high-resolution PNG is created and embedded in the RTF file (technical details are in the third section).

by causes a chart for all values of **colorder_label**, which are, given the subset, 'All TMC' and 'Active Control'. Also note that the by-value is used in the embedded title (see the parameter **title_embedded**, and the graphs below).

Number of subjects by ethnic origin

n (%)	400 mg bid	800 mg bid	All TMC	Active Control	All Subjects
TOTAL	80 (100.0)	79 (100.0)	159 (100.0)	40 (100.0)	199 (100.0)
CAUCASIAN / WHITE	45 (56.3)	47 (59.5)	92 (57.9)	29 (72.5)	121 (60.8)
BLACK	17 (21.3)	15 (19.0)	32 (20.1)	7 (17.5)	39 (19.6)
HISPANIC	16 (20.0)	17 (21.5)	33 (20.8)	3 (7.5)	36 (18.1)
OTHER	1 (1.3)		1 (0.6)	1 (2.5)	2 (1.0)
NOT ALLOWED TO ASK PER LOCAL REGULATIONS	1 (1.3)		1 (0.6)		1 (0.5)

Ethnic origin: 'All TMC' versus 'Active Control'



3. FINAL EXAMPLE

This final example shows some more features of the in-text system at hand.

```
%init table(
outdir=C:\Output,
outfile=T03 Viral load (a),
colclass ds=data.vl,
colclass subset=randgrp in (1 2),
colclass=randgrp,
fmt=randgrp.,
topleft label="Viral load",
title="Viral load",
css=C:\Styles\Presentation.css,
colspan1=x-x,
colspan1 label="TMC",
orientation=landscape
);
%add section(
section label="Change from baseline",
ds=data.vl,
subset=use="YES" and week in (0 12 24 48),
rowclass=week,
fmt rowclass=week.,
param=count/denom (percent)@mean (stddev),
fmt=3.0-3.0-5.1-5.1-6.2,
placement=column,
var=logv1chg,
percent ds=data.vl,
percent_distinct=usubjid
);
%add section(
section label="Response as copies <50",
ds=data.vl,
subset=use="YES" and week in (0 12 24 48) and
response=1,
rowclass=week,
fmt rowclass=week.,
param=count/denom (percent),
fmt=3.0-3.0-5.1,
var=response,
percent_ds=data.vl,
```

colspan produces spanning above the table header. Only one layer is produced in this example. The expression 'x-x' means that the two columns (represented by a symbolic x) are merged. In case of, for instance, five columns, the expression could take the form 'x-x|x-x|x'. In that case, column 1 and 2 are merged, 3 and 4 as well, and column 5 remains separated.

orientation is portrait or landscape, and refers to the page setup of the generated RTF file.

percent_by specifies the denominator that needs to be used for percentages. This is especially useful in case of by-processing (this option is not shown in this paper) or in case of a denominator varying with the values of *rowclass* (shown here). To provide an example, the denominators are added to each section of the table. In the first section, the denominator is the number of subjects at baseline, and so the percentage indicates the survival rate. In the second section, the denominator is the number of subjects at each time point, hence the percentage indicates the response rate.

add_plot chiefly passes SAS/GRAPH statements to PROC Gplot. A high-resolution PNG is created and embedded in the RTF file (technical details are in the third section).

BW produces a black and white plot, in addition to the color graph. Only the color version is shown in the output below; the black and white one will be used further down.

```

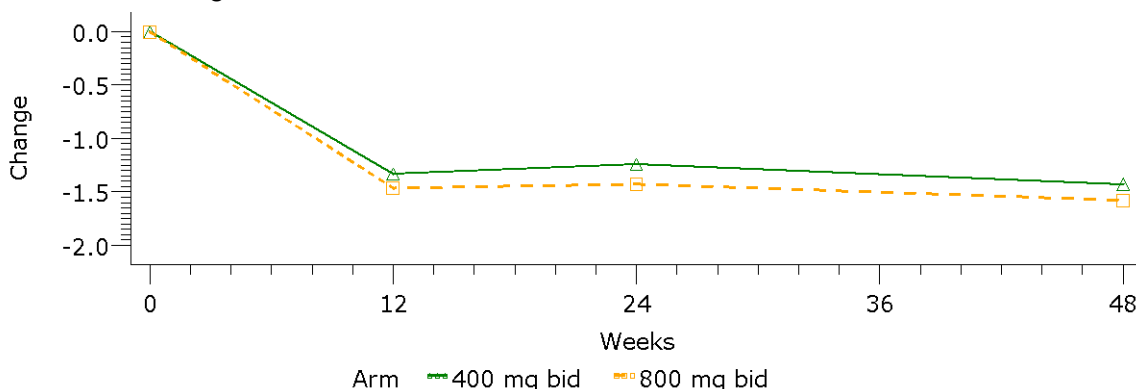
percent_by=week,
percent_distinct=usubjid
);
%add plot(
hsize=6in, vsize=2in,
title="Viral load: Change from baseline",
subset=section=1,
plot=plot mean*week=randgrp,
colors=green orange gray,
symbols h=3, symbols w=2,
haxis=order=(0 to 48 by 12) label=("Weeks"),
vaxis=order=(-2 to 0 by 0.5) label=(a=90 r=0
"Change"),
statements=%str(format randgrp randgrp.);,
legend=label=("Arm") value=(j=left),
BW=Y
);
%assemble_sections;

```

Viral load

Viral load	TMC	
	400 mg bid	800 mg bid
	n/N (%) Mean (SD)	n/N (%) Mean (SD)
Change from baseline		
Baseline	80/80 (100.0) 0.0 (0.00)	79/79 (100.0) 0.0 (0.00)
Week 12	69/80 (86.3) -1.3 (1.28)	70/79 (88.6) -1.5 (1.25)
Week 24	61/80 (76.3) -1.2 (1.37)	65/79 (82.3) -1.4 (1.27)
Week 48	48/80 (60.0) -1.4 (1.58)	48/79 (60.8) -1.6 (1.37)
Response as copies <50		
Week 12	35/69 (50.7)	39/70 (55.7)
Week 24	28/61 (45.9)	34/65 (52.3)
Week 48	24/48 (50.0)	26/48 (54.2)

Viral load: Change from baseline



Note: only the color version is shown; the monochrome version comes back in the section on *styling and assembling*.

STYLING AND ASSEMBLING

In line with the creation of an individual output (previous section), we do not seek to present all the possibilities of styling and assembling (current section). Rather, and again with examples, we will attempt to convey the main principles. Afterwards, we redirect the focus to the underlying technicalities (next section).

1. STYLING

In the example section, three RTF files have been generated:

(in 1. Simple Example)	T01 Ethnic origin (a).rtf	with style = report style (built-in default)
(in 2. Second Example)	T02 Ethnic origin (b).rtf	with style = presentation style as defined by Presentation.css
(in 3. Final Example)	T03 Viral load (a).rtf	with style = presentation style as defined by Presentation.css

Suppose we would like to dispose of the viral load output, but in report style. Reprocessing the code of example 3, by leaving out the parameter `css=C:\Styles\Presentation.css`, is one way of doing that. A simpler way is to start from the viral load file which has already been generated (T03 Viral load (a).rtf). The RTF files are built up such that styles can be read and rewritten quickly and easily. The following code (left or right) does the style transformation:

```

%mod_style(
file=C:\Output\T03 Viral load (a).rtf,
css=,
outfile=C:\Output\T03 Viral load (b).rtf
);

```

```

%mod_style(
file=C:\Output\T03 Viral load (a).rtf,
css=,
replace=Y
);

```

The report style, which is the built-in default, is selected as CSS is left blank. Another user-defined style, Presentation.css is just one style definition, could have been picked at this point.

2. ASSEMBLING

Another possibility is to assemble several RTF files into one master document. At this point too, the document can be restyled. In addition, a choice of color and/or monochrome graphs can be made. The following code creates a master report:

```
%assemble_document(  
indir=C:\Output,  
outdir=C:\Output,  
outfile=Assembled document,  
orientation=portrait,  
css=,  
subset=file in ("T01 Ethnic origin (a).rtf" "T03 Viral load (a).rtf"),  
graphs_bw=Y,  
graphs_color=N  
);
```

The file of example 1 (T01 Ethnic origin (a).rtf) contains no graphs. The report style was used for this file.

The file of example 3 (T03 Viral load (a).rtf) contains both color and monochrome graphs, but only monochrome graphs are selected in our example. The presentation style was used for this file.

Only monochrome graphs will be selected into the master document. All styles will be replaced by the report style as CSS is left blank.

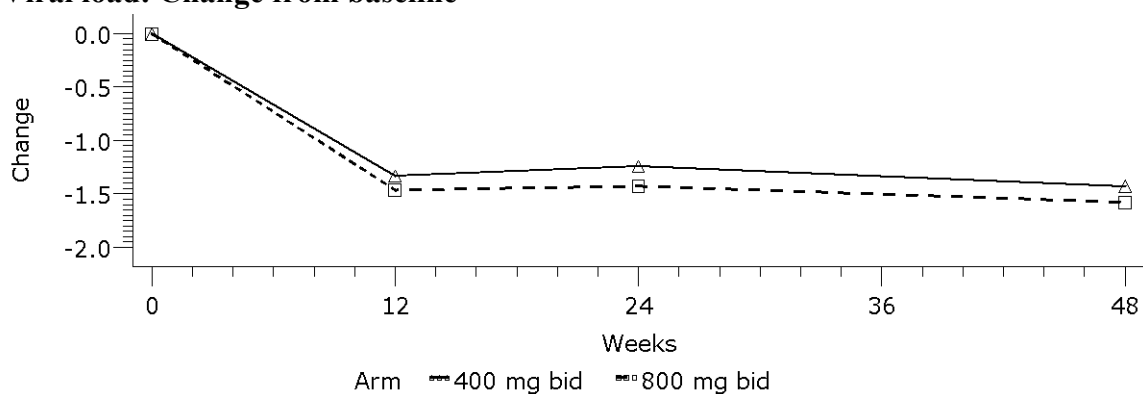
The master document (Assembled document.rtf) is displayed on the following page:

Number of subjects by ethnic origin

n (%)	400 mg bid	800 mg bid	Active Control
BLACK	17 (21.3)	15 (19.0)	7 (17.5)
CAUCASIAN / WHITE	45 (56.3)	47 (59.5)	29 (72.5)
HISPANIC	16 (20.0)	17 (21.5)	3 (7.5)
NOT ALLOWED TO ASK PER LOCAL REGULATIONS	1 (1.3)		
OTHER	1 (1.3)		1 (2.5)

Viral load

	TMC	
Viral load	400 mg bid	800 mg bid
	n/N (%) Mean (SD)	n/N (%) Mean (SD)
Change from baseline		
Baseline	80/80 (100.0) 0.0 (0.00)	79/79 (100.0) 0.0 (0.00)
Week 12	69/80 (86.3) -1.3 (1.28)	70/79 (88.6) -1.5 (1.25)
Week 24	61/80 (76.3) -1.2 (1.37)	65/79 (82.3) -1.4 (1.27)
Week 48	48/80 (60.0) -1.4 (1.58)	48/79 (60.8) -1.6 (1.37)
Response as copies <50		
Week 12	35/69 (50.7)	39/70 (55.7)
Week 24	28/61 (45.9)	34/65 (52.3)
Week 48	24/48 (50.0)	26/48 (54.2)

Viral load: Change from baseline

TECHNICAL DETAILS

1. PROCESS FLOW

The production of tables and listings is similar. The overall process consists of three steps. The first step, i.e. the macro call `%init_table`, contains all settings that need to be the same for the entire table and output file. The second block consists of one (or several) sections, i.e. the calls `%add_section`, `%add_chart`, `%add_plot`. Per section, SAS creates a small temporary SAS data set, containing RTF code (also for graphics, see further below). We will name it a 'section data set'. The final step is the macro call `%assemble_sections`, having no parameters. At that point, SAS starts to compile all sections and writes one single file to the file system.

The second step, i.e. the block of sections and graphics, is not entirely the same for tables compared to listings. For tables, one extra substep is required: the construction of a summary data set.

To summarize, the overall process is:

(1) Initialization of the table (title, column widths, etc.) and the output file (name, style, etc.).

(2) Creation of section data sets. The calls to `%add_section`, `%add_chart`, `%add_plot` are processed sequentially, and each call leads to:

- Where processing, if parameter `subset` is specified (see example 3), restricting the data right from the start (performance).
- By-processing, leading to several sections/graphs (see example 2), as if per by-value a section/graph had been specified.
- In case of a graph, the creation of an image, involving PROC GCHART, PROC GPLOT, and also ODS RTF (see further below).
- In case of a table (*not* in case of a listing), the calculation of summary statistics, involving PROC FREQ for counts and percentages, and PROC MEANS for other statistics.
- In case of table or listing, the ordering of the result set, if parameter `out_by` is specified (see example 2).
- In case of table or listing, a subset of the result set, if parameter `out_subset` is specified.
- The creation of an empty section data set (an empty container) which will be filled by the actions below.
- In case of table or listing, the addition of the table header to the section data set, when it concerns the first section.
- The addition of the actual section output to the section data set.

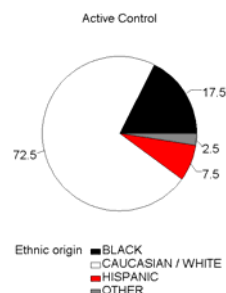
(3) Compilation of all section data sets. The only new elements at this point are the style declaration, and the opening and closing tags of the RTF file.

2. GRAPHS

When it comes to graphs, the system does make a difference between HTML versus RTF. In HTML, our graphs are not embedded, they are separate PNG, GIF, or EMF files, *linked* to the HTML output file at hand. We discard ActiveX here, because only basic and not all graphs can be created with ActiveX (e.g. annotation difficulties). In RTF, we use PNG files, *embedded* in the output file at hand. The graph is described in hexadecimal coding, and RTF processors visualize the graph appropriately. This does not mean that it is a vector graph (it is a bitmap), the visualization of our PNGs has no common ground with the reproduction of a vector graph by, for instance, a word processor.

In the RTF output file, our high-resolution PNG graphs looks as follows:

```
{\comment START OF COLOR GRAPH}
\pard\plain{\(*\shppict{\pict\pngblip
89504E470D0A1A0A000000D494844520000025800...
000000FFC0C0C0808080000000BD180F8000000009...
9B552BE8717955E7CCC492910C976AD01667590000...
75F057A8036621D6FC34122BB409B1A0A15867B644...
2A1CA3D7BDFA9E02949C8DD79C3B5B8081F9F46BB8...
...
}}}\pard\plain \s0 \par
{\comment END OF COLOR GRAPH}
```

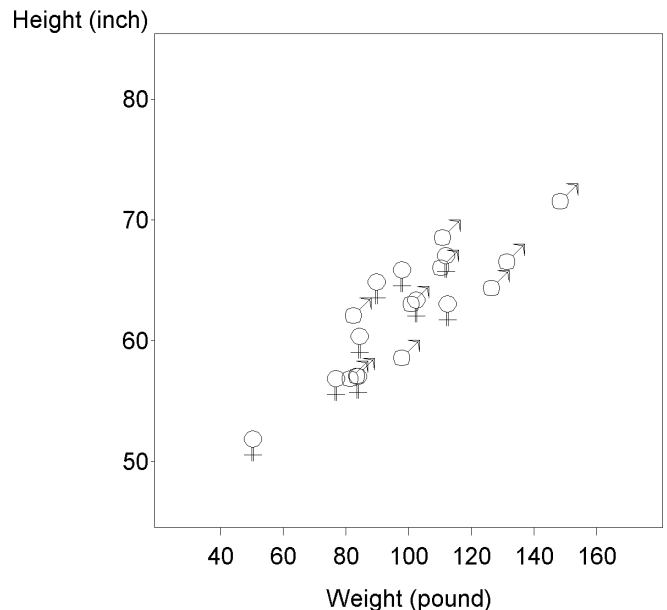


We briefly outline how the embedded PNG graphs can be produced with the SAS System. The trick is to combine ODS RTF with graphics driver PNG. For our in-house system, we then read in the resulting RTF file, and filter out the hexadecimal code that describes the aspired graph. The code below (left) is a simple example showing how to produce such a graph (right) with SAS.

```

goptions
  gsfname=gsf
  device=png target=png
  xmax=4in ymax=4in
  xpixels=1200 ypixels=1200 /* 300dpi */
  ftext="Arial" htext=12pt
  colors=(black)
;
filename gsf "C:\Output\Example.png";
ods rtf file="C:\Output\Example.rtf";
proc gplot data=sashelp.class;
  plot Height*Weight=Sex / nolegend
        haxis=axis1
        vaxis=axis1;
  symbol1 v=* h=3; /* value for symbol Female */
  symbol2 v=> h=3; /* value for symbol Male */
  axis1 offset=(10pct, 10pct) minor=none;
  label Height="Height (inch)"
        Weight="Weight (pound)";
run;
quit;
ods rtf close;

```



We prefer an embedded graph because all information is contained by one physical file. With information, we mean text, associated style attributes, and graphs. This does not prevent us to alter the style of a file, or to compile several files into a new document (with perhaps another style), as shown in the examples and as will be explained further below. Because we prefer embedded graphs, we are 'convicted' to RTF. Another advantage of RTF, speaking of clinical development, is that medical writers are used to work with word processors, and that is also the ultimate format of all clinical research reports.

3. STYLING

The style is already fixed in the first step (see the process flow above), because, and that is an RTF specific, we need it twice. We need it in the declaration part, analogous to a CSS style definition in HTML. But we also need it in the content part itself, and that is different from HTML. In an HTML file, using CSS, one would *refer* to the style name, and the style attributes would then be picked up from the declaration part. In RTF, the style name is mentioned, but *also* the style attributes themselves are repeated. The simple RTF example below illustrates this point. The style attribute `\i` is repeated at the time that it is actually used, i.e. in the second paragraph. This is not a matter of choice as it is the only valid RTF that works. The reason is that, historically, not all RTF readers supported styles; if styles were not supported, then the RTF content would still be visualized correctly.

```

{\rtf
  {\stylesheet
    {\s0 \sbasedon222\snext0 Normal;}
    {\s1 \i \sbasedon0\snext0 Styled;}
  }

  \pard This is a default paragraph.\par
  \pard \s1 \i This is a "Styled" paragraph.\par
}

```

This is a default paragraph.

This is a "Styled" paragraph.

Applied to our in-text system, it helps to understand why the style is fixed in the first step, i.e. by the macro call `%init_table`. The *declaration* of the styles is only needed when all sections are assembled (third and final step). However, styles have to be embedded in the *actual output* of the sections as well (second step). Let us revisit the first example of this paper (see the example section). The style for the table title 'Number of subjects by ethnic origin', appears twice. In the stylesheet, as:

```
{\s1 \f1\fs24\b\sa40\li40\ri40\sb200\ql \sbasedon0\snext0 tablettitle;};
```

and when it is used:

```
\pard\plain \s1 \f1\fs24\b\sa40\li40\ri40\sb200\ql Number of subjects by ethnic origin\par.
```

The first example made use of the default style. No call to another, external style is found in the code for example 1. This was not the case for the second example. In the second example, the parameter `css=C:\Styles\Presentation.css` calls in an external style. Regardless of the mode (HTML or RTF), the external style should be specified in the form of a Cascading Style Sheet (CSS). The reason is that a CSS is much more user-friendly than styles specified in RTF. A list of all styles in the system (left) and a fragment of the CSS file used for example 2 (right) is taken up below.

List of styles:

- default (shown right)

```
.default{
/* cell properties */
border: 1pt solid gray;
```

PhUSE 2006

- `tabletitle` (shown below and right)
- `tablefootnote`
- `graphtitle`
- `graphfootnote`
- `toleft` (shown below and right)
- `colclass`
- `colspan`
- `colorder` (shown below and right)
- `param` (shown below and right)
- `section` (shown below and right)
- `rowclass`
- `row` (shown below and right)
- `rowleft` (shown below and right)
- `firstrow`
- `firstrowleft`
- `midrow`
- `midrowleft`
- `lastrow`
- `lastrowleft`

Viral load		
TMC		
Viral load	400 mg bid	800 mg bid
	n/N (%)	n/N (%)
	Mean (SD)	Mean (SD)
Change from baseline		
Baseline	80/80 (100.0) 0.0 (0.00)	79/79 (100.0) 0.0 (0.00)
Week 12	69/80 (86.3) -1.3 (1.28)	70/79 (88.6) -1.5 (1.25)
Week 24	61/80 (76.3) -1.2 (1.37)	65/79 (82.3) -1.4 (1.27)
Week 48	48/80 (60.0) -1.4 (1.58)	48/79 (60.8) -1.6 (1.37)
Response as copies <50		
Week 12	35/69 (50.7)	39/70 (55.7)
Week 24	28/61 (45.9)	34/65 (52.3)
Week 48	24/48 (50.0)	26/48 (54.2)

```

vertical-align: middle;
background-color: silver;
/* text properties */
margin-left: 2pt;
margin-right: 2pt;
margin-top: 0pt;
margin-bottom: 0pt;
text-align: center;
color: black;
font-family: arial;
font-size: 10pt;
font-weight: normal;
font-style: normal;
}
.tabletitle{
margin-top: 8pt;
margin-bottom: 4pt;
text-align: left;
font-size: 11pt;
}
.colspan{
font-weight: bold;
font-style: italic;
}
.toleft{
background-color: black;
text-align: left;
color: yellow;
font-size: 11pt;
font-weight: bold;
}
.colorder{
color: navy;
font-weight: bold;
}
.param{
}
.section{
text-align: left;
color: navy;
font-weight: bold;
}
.row{
background-color: yellow;
}
.rowleft{
text-align: left;
}
...

```

The default style is the only style that pops up in the CSS file but that will not appear in the style declaration of the eventual output file. Any style attribute that is not specified, will *inherit* its value from the default style. And if not all attributes are specified in the default style, then the default style will *inherit* from the built-in style (the report style is the built-in default). The double inheritance makes the development or modification of styles easier and faster. One chooses a new standard (e.g. for the font, the border, etc.); next, only the deviations from the defaults need to be stated.

4. ASSEMBLING

We could have a style issue when several output files are assembled to one, new document. For instance, a first output file making use of the report style may conflict with a second output file in presentation style. This was the case for the created master document at the end of the previous section. However, when we create such a master document, one consistent style can (and has to) be specified. In absence of a specified style, this will be the built-in style. What happens, technically, is the following. The style declarations of all output files involved, are discarded. The new style definition is pasted at the top of the master document, followed by the content (tables, graphs) of the output files at hand. In RTF mode, very important, the embedded styles are also replaced by the active style (see the previous topic for this RTF specific).

If one wants to change the style of a single file, then there are two options. In line with the previous paragraph, one could call in `%assemble_document`, selecting one single input file and writing to that same file. The easier and preferred way is to use `%mod_style`, specifying the target style. In that case, the output file can be the same file, or still another file if the original is to be preserved.

CONCLUSION

For in-text tables and graphs, requirements do differ (slightly) from one department to another. These differing expectations came down to the classic challenges: How to split content from style? How to take up graphs? And how permanent is a style, once an in-text file has been produced?

We presented a case based answer, via the in-text system developed at Tibotec. We focused primarily on the production of RTF files containing tables, listings, and embedded PNG graphs. Three special features impact the flexibility of the in-text system. The first feature is the use of styles that are each stored in a Cascading Style Sheet (CSS) file. The second is the possibility to alter the style of an output file. The third is the possibility to compile several output files into a new output file, referred to as a master document; at that point, a renewed decision on tables, graphs, and style could still be made.

The developed system is tailored and specific, but it produces almost any table we use. However, it is by no means a replacement for ODS. On the contrary, ODS RTF is used to generate the embedded PNG files. And, as a matter of fact, we have presented our in-text system to a preferred Contract Research Organization (CRO), with the question to come up with something similar, as we would like to source out (part of) the creation of in-text tables. The intermediate results that we have seen so far look very promising, and the technical approach is a combination of ODS RTF, PROC TEMPLATE, and PROC REPORT. Finally, also ODS Document may lead to interesting developments in this area.

CONTACT INFORMATION

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

Mark Janssens
Tibotec (Johnson & Johnson)
L 11B 3, Generaal De Wittelaan
2800 – Mechelen
Belgium
0032 15 44 42 11 – 0032 472 59 60 48
mjansse2@tibbe.jnj.com