The sdapslayout package*

Benjamin Berg
benjamin@sipsolutions.net

February 15, 2020

1 Documentation

Please refer to https://sdaps.org/class-doc for documentation.

2 Implementation

This package uses the LaTeX3 language internally, so we need to enable it.

```
% We need at least 2011-08-23 for \keys_set_known:nnN
\RequirePackage{expl3}[2011/08/23]
%\RequirePackage{xparse}
\ExplSyntaxOn
```

And we need a number of other packages.

```
\ExplSyntaxOff

\RequirePackage{sdapsbase}
\RequirePackage{sdapsarray}
\RequirePackage{xparse}


\ExplSyntaxOn

```

2.1 Choice Question Layout

2.1.1 Choice Question Matrix Layout

The following macros provide the funcitonality to layout choice questions in a matrix like fashion.

```

\tl_new:N \l_sdaps_choicearray_qobject_type_tl
\bool_new:N \l_sdaps_choicearray_horizontal_bool
\tl_new:N \l_sdaps_choicearray_layouter_tl
\tl_new:N \l_sdaps_choicearray_align_tl
\tl_new:N \l_sdaps_choicearray_extra_tl
\tl_new:N \l_sdaps_choicearray_type_tl
```

*This document corresponds to sdapslayout v0.1, dated 2015/04/10.

```
\tl_new:N \l_sdaps_choice_var_tl
\tl_new:N \l_sdaps_choice_val_tl
\tl_new:N \l_sdaps_choice_text_tl
\tl_new:N \l_sdaps_question_var_tl
\tl_new:N \l_sdaps_question_text_tl
\clist_new:N \l_sdaps_question_range_clist
\int_new:N \g_sdaps_choices_box_int
\seq_new:N \g_sdaps_choices_filter_seq
\seq_new:N \g_sdaps_choices_cell_seq
\seq_new:N \g_sdaps_choices_text_seq

\keys_define:nn { sdaps / choicearray }
{
  horizontal .bool_set:N = \l_sdaps_choicearray_horizontal_bool,
  horizontal .default:n  = true,
  horizontal .initial:n  = true,
  vertical   .bool_set_inverse:N = \l_sdaps_choicearray_horizontal_bool,
  vertical   .default:n  = true,
  layouter   .tl_set:N   = \l_sdaps_choicearray_layouter_tl,
  layouter   .initial:n  = default,
  align      .tl_set:N   = \l_sdaps_choicearray_align_tl,
  align      .initial:n  = { },

  type       .choices:nn  = { multichoice, singlechoice } { \tl_set:Nx \l_sdaps_choicearray_t
  type       .initial:n   = { multichoice },

  singlechoice .meta:n   = { type=singlechoice },
  multichoice  .meta:n   = { type=multichoice },

  noalign    .meta:n  = { align= },
}

\keys_define:nn { sdaps / choicearray / choice }
{
  var        .tl_set:N   = \l_sdaps_choice_var_tl,
  val        .tl_set:N   = \l_sdaps_choice_val_tl,
  text       .tl_set:N   = \l_sdaps_choice_text_tl,
}

\keys_define:nn { sdaps / choicearray / question }
{
  var        .tl_set:N   = \l_sdaps_question_var_tl,
  text       .tl_set:N   = \l_sdaps_question_text_tl,

  range      .clist_set:N   = \l_sdaps_question_range_clist,
  range      .initial:n = {...}
}

\cs_new_protected_nopar:Npn \_sdaps_choicearray_preprocess:n #1
{
  \keys_set_known:nnN { sdaps / choicearray } { #1 } \l_sdaps_choicearray_extra_tl

  \sdaps_checkbox_set_type:V \l_sdaps_choicearray_type_tl
  \tl_if_eq:VnTF \l_sdaps_choicearray_type_tl { multichoice } {
```

```
    \tl_set:Nn \l_sdaps_choicearray_qobject_type_tl { Choice }
  } {
    \tl_set:Nn \l_sdaps_choicearray_qobject_type_tl { Option }
  }
}
\cs_generate_variant:Nn \_sdaps_choicearray_preprocess:n { V }

\cs_new_protected_nopar:Npn \_sdaps_choicearray_process_choice_insert_tail_after:w {
  \bgroup
    \group_insert_after:N \_sdaps_choicearray_process_choice_tail:
    \sdaps_array_nested_alignenv:
    \tex_let:D\next=
}

\cs_new_protected_nopar:Nn \_sdaps_choicearray_process_choice_tail: {
  \ignorespaces
}

\cs_new_nopar:Nn \_sdaps_choicearray_grab_choice:n {
  \seq_gput_right:Nn \g_sdaps_choices_text_seq { #1 }
  \group_begin:
    \sdaps_array_nested_alignenv:
    #1
  \group_end:
  \_sdaps_choicearray_process_choice_tail:
}

\cs_new_nopar:Npn \_sdaps_choicearray_process_choice:nw #1
{
  % This modifies grouping so it has to be at the start
  \sdaps_array_alignment:
  \leavevmode

  \tl_clear:N \l_sdaps_choice_var_tl
  \tl_clear:N \l_sdaps_choice_val_tl
  \tl_clear:N \l_sdaps_choice_text_tl

  \keys_set:nn { sdaps / choicearray / choice } { #1 }

  \int_gincr:N \g_sdaps_choices_box_int

  \tl_if_empty:NT \l_sdaps_choice_var_tl {
    % Prefix with _ to force prefixing with generated variable
    \tl_set:Nx \l_sdaps_choice_var_tl { \int_use:N \g_sdaps_choices_box_int }
  }
  \tl_if_empty:NT \l_sdaps_choice_val_tl {
    \tl_set:Nx \l_sdaps_choice_val_tl { \int_use:N \g_sdaps_choices_box_int }
  }

  \tl_if_eq:VnTF \l_sdaps_choicearray_type_tl { multichoice } {
    \seq_gput_right:NV \g_sdaps_choices_filter_seq \l_sdaps_choice_var_tl
  } {
    \seq_gput_right:NV \g_sdaps_choices_filter_seq \l_sdaps_choice_val_tl
  }
```

```
129
130   \seq_gput_right:Nx \g_sdaps_choices_cell_seq { \exp_not:n { \sdaps_checkbox:nn } { _ \l_sda
131
132   \tl_if_empty:NTF \l_sdaps_choice_text_tl {
133     % We need to leave a command in the stream that grabs the next parameter
134     % and outputs it immediately
135     \cs_set_eq:NN \l_tmpa_token \_sdaps_choicearray_grab_choice:n
136   } {
137     % Nothing else to do
138     \seq_gput_right:NV \g_sdaps_choices_text_seq { \l_sdaps_choice_text_tl }
139     \cs_set_eq:NN \l_tmpa_token \_sdaps_choicearray_process_choice_insert_tail_after:w
140   }
141   \l_tmpa_token
142 }
143 \cs_generate_variant:Nn \_sdaps_choicearray_process_choice:nw { Vw }
144
145 \cs_new_protected_nopar:Nn \_sdaps_choicearray_process_question_grab:n {
146   \tl_set:Nn \l_sdaps_question_text_tl { #1 }
147
148   \_sdaps_choicearray_process_question_head:
149
150   \group_begin:
151     \sdaps_array_nested_alignenv:
152     #1
153   \group_end:
154
155   \_sdaps_choicearray_process_question_tail:
156 }
157
158 \cs_new_protected_nopar:Npn \_sdaps_choicearray_process_question_inline:w {
159   \_sdaps_choicearray_process_question_head:
160   \bgroup
161     \group_insert_after:N \_sdaps_choicearray_process_question_tail:
162     \sdaps_array_nested_alignenv:
163     \tex_let:D\next=
164 }
165
166 \cs_new_protected_nopar:Nn \_sdaps_choicearray_process_question_head: {
167   \sdaps_qobject_begin:nnV { choicearray_question } \l_sdaps_choicearray_qobject_type_tl \l_s
168
169   \tl_if_empty:NF \l_sdaps_question_var_tl {
170     \sdaps_qobject_append_var:V \l_sdaps_question_var_tl
171   }
172 }
173
174 \cs_new_protected_nopar:Nn \_sdaps_choicearray_process_question_tail: {
175   \seq_gclear:N \g_tmpa_seq
176   % l_tmpa_bool is whether we are in a run (i.e. ...)
177   \bool_set_false:N \l_tmpa_bool
178
179   \clist_gset_eq:NN \g_tmpa_clist \l_sdaps_question_range_clist
180   \clist_gpop:NN \g_tmpa_clist \l_tmpa_tl
181
182   \seq_map_inline:Nn \g_sdaps_choices_filter_seq {
```

```
    \tl_if_eq:VnT \l_tmpa_tl { ... } {
      \bool_set_true:N \l_tmpa_bool
      \clist_gpop:NN \g_tmpa_clist \l_tmpa_tl
    }

    \tl_if_eq:VnTF \l_tmpa_tl { ##1 } {
      \seq_gput_right:Nn \g_tmpa_seq { \c_true_bool }
      \bool_set_false:N \l_tmpa_bool
      \clist_gpop:NN \g_tmpa_clist \l_tmpa_tl
    } {
      % Append if we are handling a run of items and the current item is not
      % the last one.
      \bool_if:NTF \l_tmpa_bool {
        \seq_gput_right:Nn \g_tmpa_seq { \c_true_bool }
      } {
        % Otherwise, ignore this item
        \seq_gput_right:Nn \g_tmpa_seq { \c_false_bool }
      }
    }
  }

  \seq_gset_eq:NN \g_tmpb_seq \g_tmpa_seq

  \seq_map_inline:Nn \g_sdaps_choices_text_seq {
    \seq_gpop_left:NN \g_tmpa_seq \l_tmpa_tl
    \tl_if_eq:VnT \l_tmpa_tl { \c_true_bool } {
      \sdaps_answer:f { ##1 }
    }
  }

  \seq_map_inline:Nn \g_sdaps_choices_cell_seq {
    \sdaps_array_alignment:

    \seq_gpop_left:NN \g_tmpb_seq \l_tmpa_tl
    \tl_if_eq:VnT \l_tmpa_tl { \c_true_bool } {
      ##1
    }
  }

  \sdaps_qobject_end:n { choicearray_question }
  \ignorespaces
}

\cs_new_nopar:Npn \_sdaps_choicearray_process_question:nw #1
{
  \sdaps_array_newline:
  \leavevmode

  \keys_set:nn { sdaps / choicearray / question } { #1 }

  \tl_if_empty:NTF \l_sdaps_question_text_tl {
    % We need to leave a command in the stream that grabs the next parameter,
    % outputs it again, and finishes the question.
    \cs_set_eq:NN \l_tmpa_token \_sdaps_choicearray_process_question_grab:n
```

```
  } {
    % Insert the question around the next argument
    \cs_set_eq:NN \l_tmpa_token \_sdaps_choicearray_process_question_inline:w
  }
  \l_tmpa_token
}
\cs_generate_variant:Nn \_sdaps_choicearray_process_question:nw { Vw }



%
```

2.2 Range Question Layout

2.2.1 Range Question Matrix Layout

The following macros provide the functionality to layout range/option questions in a matrix like fashion.

```


\tl_new:N \l_sdaps_rangearray_align_tl
\tl_new:N \l_sdaps_rangearray_extra_tl
\int_new:N \l_sdaps_rangearray_rangecount_int
\bool_new:N \l_sdaps_rangearray_other_bool
\tl_new:N \g_sdaps_question_var_tl
\tl_new:N \g_sdaps_question_text_tl
\tl_new:N \g_sdaps_question_lowertext_tl
\tl_new:N \g_sdaps_question_uppertext_tl
\tl_new:N \g_sdaps_question_othertext_tl

\msg_new:nnn { sdapslayout } { option_not_supported } { The~#1~option~is~not~supported~by~thi

\keys_define:nn { sdaps / rangearray }
{
  count      .int_set:N  = \l_sdaps_rangearray_rangecount_int,
  count      .initial:n  = 5,
  align      .tl_set:N   = \l_sdaps_rangearray_align_tl,
  align      .initial:n  = { },
  % Override and disallow flipping; it does not work currently
  flip       .code:n     = { \msg_error:nnn { sdapslayout } { option_not_supported } { flip }
  other      .bool_set:N = \l_sdaps_rangearray_other_bool,
  other      .default:n  = true,
  other      .initial:n  = false,
}

\keys_define:nn { sdaps / rangearray / question }
{
  var        .tl_gset:N   = \g_sdaps_question_var_tl,
  text       .tl_gset:N   = \g_sdaps_question_text_tl,
  upper      .tl_gset:N   = \g_sdaps_question_uppertext_tl,
  lower      .tl_gset:N   = \g_sdaps_question_lowertext_tl,
  other      .tl_gset:N   = \g_sdaps_question_othertext_tl,
}

```

```
\cs_new_protected_nopar:Npn \_sdaps_rangearray_preprocess:n #1
{
  \keys_set_known:nnN { sdaps / rangearray } { #1 } \l_sdaps_rangearray_extra_tl

  \sdaps_checkbox_set_type:n { singlechoice }
}
\cs_generate_variant:Nn \_sdaps_rangearray_preprocess:n { V }

% Before/After the different parts

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_question: {
  \sdaps_array_newline:
  % Note: This needs to be after sdaps_array_newline as the command may be
  %       discarded otherwise (i.e. it does not make it into the output stream)
  %       We also need to leave vmode here
  \leavevmode
  \sdaps_qobject_begin:nnV { rangearray_question } { Range } \g_sdaps_question_text_tl

  \tl_if_empty:NF \g_sdaps_question_var_tl {
    \sdaps_qobject_append_var:V \g_sdaps_question_var_tl
  }

  \ignorespaces
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_lower: {
  % right align
  \sdaps_array_alignment:
  \leavevmode
  \sdaps_range:nnV { lower } { 0 } \g_sdaps_question_lowertext_tl
  \sdaps_if_rtl:F {
    \hfill
  }
  \ignorespaces
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_upper: {
  \sdaps_array_alignment:
  \leavevmode
  \sdaps_range:nnV { upper } { \l_sdaps_rangearray_rangecount_int - 1 } \g_sdaps_question_upp
  \sdaps_if_rtl:T {
    \hfill
  }
  \ignorespaces
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_before_other: {
  % Insert an extra empty column for further spacing
  \sdaps_array_alignment:
  \sdaps_array_alignment:
  \leavevmode
  \sdaps_answer:V \g_sdaps_question_othertext_tl
  \sdaps_if_rtl:TF {
    \hfill
```

```
  } {
    \sdaps_checkbox:nn { } { 0 } {} ~ {}
  }
  \ignorespaces
}



\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_after_question: {
  % Insert an extra empty column for further spacing
  \sdaps_array_alignment:
  \tl_if_empty:NTF \g_sdaps_question_lowertext_tl {
    \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_grab_lower:n
  } {
    \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_inline_lower:w
  }
  \l_tmpa_token
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_after_lower: {
  % Insert the option checkbox column
  \sdaps_if_rtl:T {
    \hfill\kern 0pt
  }
  \sdaps_array_alignment:
  \leavevmode
  % Seems like right to left writing mode is not started without a paragraph
  \sdaps_if_rtl:T { \beginR }
  % Assume we have at least one checkbox
  \sdaps_checkbox:nn { } { 1 }
  \int_step_inline:nnnn { 2 } { 1 } { \l_sdaps_rangearray_rangecount_int } {
    \hspace{1em} \sdaps_checkbox:nn { } { ##1 }
  }
  \sdaps_if_rtl:T { \endR }

  \tl_if_empty:NTF \g_sdaps_question_uppertext_tl {
    \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_grab_upper:n
  } {
    \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_inline_upper:w
  }
  \l_tmpa_token
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_after_upper: {
  \sdaps_if_rtl:F {
    \hfill\kern 0pt
  }

  \bool_if:NTF \l_sdaps_rangearray_other_bool {
    \tl_if_empty:NTF \g_sdaps_question_othertext_tl {
      \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_grab_other:n
    } {
      \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_inline_other:w
    }
```

```
  } {
    \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_finish:
  }
  \l_tmpa_token
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_after_other: {
  \sdaps_if_rtl:TF {
    {} ~ \sdaps_checkbox:nn { } { 0 } {}
  } {
    \hfill\kern 0pt
  }
  \ignorespaces
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_finish: {
  \sdaps_qobject_end:n { rangearray_question }
  \ignorespaces
}

% Processors for inline processing/grabbing the argument

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_question:n {
  \tl_gset:Nn \g_sdaps_question_text_tl { #1 }

  \_sdaps_rangearray_process_question_before_question:
  \group_begin:
    \sdaps_array_nested_alignenv:
    #1
  \group_end:
  \_sdaps_rangearray_process_question_after_question:
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_lower:n {
  \tl_gset:Nn \g_sdaps_question_lowertext_tl { #1 }

  \_sdaps_rangearray_process_question_before_lower:
  \group_begin:
    \sdaps_array_nested_alignenv:
    #1
  \group_end:
  \_sdaps_rangearray_process_question_after_lower:
}

\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_upper:n {
  \tl_gset:Nn \g_sdaps_question_uppertext_tl { #1 }

  \_sdaps_rangearray_process_question_before_upper:
  \group_begin:
    \sdaps_array_nested_alignenv:
    #1
  \group_end:
  \_sdaps_rangearray_process_question_after_upper:
}
```

```
\cs_new_protected_nopar:Nn \_sdaps_rangearray_process_question_grab_other:n {
  \tl_gset:Nn \g_sdaps_question_othertext_tl { #1 }

  % If the text is empty, assume that this particular question does not have
  % an alternative choice. Note that this column might not exist and this
  % macro will not even be called in that case.
  % If we skip the optional "other" option then we still need to insert the
  % alignment to create the column.
  \tl_if_empty:NTF \g_sdaps_question_othertext_tl {
    \sdaps_array_alignment:
    \leavevmode
    \ignorespaces
  } {
    \_sdaps_rangearray_process_question_before_other:
    \group_begin:
      \sdaps_array_nested_alignenv:
      #1
    \group_end:
    \_sdaps_rangearray_process_question_after_other:
  }
  \_sdaps_rangearray_process_question_finish:
}



\cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_question:w {
  \_sdaps_rangearray_process_question_before_question:
  \bgroup
    \group_insert_after:N \_sdaps_rangearray_process_question_after_question:
    \sdaps_array_nested_alignenv:
    \tex_let:D\next=
}

\cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_lower:w {
  \_sdaps_rangearray_process_question_before_lower:
  \bgroup
    \group_insert_after:N \_sdaps_rangearray_process_question_after_lower:
    \tex_let:D\next=
}

\cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_upper:w {
  \_sdaps_rangearray_process_question_before_upper:
  \bgroup
    \group_insert_after:N \_sdaps_rangearray_process_question_after_upper:
    \tex_let:D\next=
}

\cs_new_protected_nopar:Npn \_sdaps_rangearray_process_question_inline_other:w {
  \_sdaps_rangearray_process_question_before_other:
  % If we reach this macro then a text has been set for the other item. This
  % means we never need to ignore the "other" parameter at this point.
  \bgroup
    \group_insert_after:N \_sdaps_rangearray_process_question_after_other:
```

```
    \group_insert_after:N \_sdaps_rangearray_process_question_finish:
    \tex_let:D\next=
}
\cs_new_nopar:Npn \_sdaps_rangearray_process_question:nw #1
{
  % Is there a better way other than clearing these before parsing?
  \tl_gclear:N \g_sdaps_question_var_tl
  \tl_gclear:N \g_sdaps_question_text_tl
  \tl_gclear:N \g_sdaps_question_uppertext_tl
  \tl_gclear:N \g_sdaps_question_lowertext_tl
  \tl_gclear:N \g_sdaps_question_othertext_tl

  \keys_set:nn { sdaps / rangearray / question } { #1 }

  \tl_if_empty:NTF \g_sdaps_question_text_tl {
    % We need to leave a command in the stream that grabs the next parameter,
    % outputs it again, and finishes the question.
    \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_grab_question:n
  } {
    % We need to generate the question after the next group stops
    \cs_set_eq:NN \l_tmpa_token \_sdaps_rangearray_process_question_inline_question:w
  }
  \l_tmpa_token
}
\cs_generate_variant:Nn \_sdaps_rangearray_process_question:nw { Vw }
%
```

2.3 Export user facing environments

```
%
\newenvironment { choicearray } [ 1 ] []
{
  \group_begin:

    \sdaps_context_get:nN { choicearray } \l_tmpa_tl
    \tl_if_eq:NNT \l_tmpa_tl \q_no_value {
      \tl_set:Nn \l_tmpa_tl {}
    }

    \tl_if_empty:nF { #1 } {
      \tl_if_empty:NTF \l_tmpa_tl {
        \tl_set:Nn \l_tmpa_tl { #1 }
      } {
        \tl_set:Nf \l_tmpa_tl { \l_tmpa_tl, #1 }
      }
```

```
    }

    \_sdaps_choicearray_preprocess:V \l_tmpa_tl
    % Clear the variables
    \seq_gclear:N \g_sdaps_choices_filter_seq
    \seq_gclear:N \g_sdaps_choices_cell_seq
    \seq_gclear:N \g_sdaps_choices_text_seq

    \int_gzero:N \g_sdaps_choices_box_int

    % Define new commands
    \newcommand \choice [ 1 ] [] {
      \_sdaps_choicearray_process_choice:nw { ##1 }
    }
    \newcommand \question [ 1 ] [] {
      \_sdaps_choicearray_process_question:nw { ##1 }
    }

    \group_begin:

      \tl_set:Nx \l_tmpb_tl {keepenv,layouter=\tl_use:N\l_sdaps_choicearray_layouter_tl,align
      \expandafter\sdapsarray\expandafter[\l_tmpb_tl]
}
{
      \endsdapsarray
    \group_end:

  \group_end:
}

\newenvironment { optionarray } [ 1 ] []
{
  \choicearray[singlechoice,#1]
}
{
  \endchoicearray
}

\newenvironment { rangearray } [ 1 ] []
{
  \group_begin:

    \sdaps_context_get:nN { rangearray } \l_tmpa_tl
    \tl_if_eq:NNT \l_tmpa_tl \q_no_value {
      \tl_set:Nn \l_tmpa_tl {}
    }

    \tl_if_empty:nF { #1 } {
      \tl_if_empty:NTF \l_tmpa_tl {
        \tl_set:Nn \l_tmpa_tl { #1 }
      } {
        \tl_set:Nf \l_tmpa_tl { \l_tmpa_tl, #1 }
      }
    }
```

```
    \_sdaps_rangearray_preprocess:V \l_tmpa_tl

    \newcommand \range [ 1 ] [] {
      \_sdaps_rangearray_process_question:nw { ##1 }
    }

    \group_begin:

      \tl_set:Nx \l_tmpb_tl {keepenv,align=\l_sdaps_rangearray_align_tl,no_header,\l_sdaps_ra
      \expandafter\sdapsarray\expandafter[\l_tmpb_tl]
}
{
      \endsdapsarray
    \group_end:

  \group_end:
}


\ExplSyntaxOff

%
```