RSP Markup Language - Reference Card

An RSP document consists of text with RSP-embedded markup. When compiled, independently of programming language, (i) comments are dropped, (ii) preprocessing directives are processed, and (iii) text and code expressions are translated into a code script. The translated code script can then be (iv) evaluated, which generates the output document, which in turn may be (v) postprocessed. [The R.rsp package knows how to postprocess output such as TeX, Markdown, Sweave, knitr etc.] Examples (in R): (1) main.tex.rsp \rightarrow (main.tex.rsp) \rightarrow main.tex \rightarrow main.pdf. (2) main.md.rsp \rightarrow (main.md.R) \rightarrow main.md \rightarrow main.html. (3) main.Rnw.rsp \rightarrow (main.Rnw.R) \rightarrow main.Rnw \rightarrow main.tex \rightarrow main.pdf.

Comments, Trimming & Escapes

Comments can be used to exclude text, code expressions and preprocessing directives.

Markup	Description
<%{anything}%>	Drops \langle anything \rangle . Number (\geq 2) of hyphens must match. Comments can be nested, if different number of hyphens.
<%-%>, <%%>,	"Empty" comments. Like above comments, these ones force following white space and line break to be dropped.
<% +%>	A hyphen (plus) attached to the end tag, forces following white space (including the line break) to be dropped (kept).
<%% and %%>	Inserts <% and %>.

Preprocessing directives

Preprocessing directives are independent of programming language used. They are applied after dropping comments and before translating text and code expressions to a code script. It is not possible to tell from the translated code script whether preprocessing directives have been used or not, nor are their variables accessible (except metadata).

Markup	Description
<pre><%@include file="\file URL\"%></pre>	Inserts the content of file $\langle \text{file} \text{URL} \rangle$ into the document before RSP-to-script translation.
<pre><%@meta \(\text{name} \) = "\(\text{content} \) "%></pre>	Assigns (content) to metadata variable (name). Metadata may be used by preprocessors, e.g. including HTML title.
<pre><%@meta name="$\langle name \rangle$"%></pre>	Inserts the content of metadata variable (name).
$<$ %@ \langle type \rangle \langle name \rangle =" \langle content \rangle "%>	Assigns (content) to preprocessing variable (name) of type (type). Supported types are 'string', 'numeric', 'integer' and 'logical'.
$\c \c \$	Inserts the content of preprocessing variable $\langle name \rangle$.
<pre><%@ifeq \(\angle\)"="\(\content\)"%></pre>	If preprocessing variable (name) equals (content), then (incl) is inserted otherwise (excl). <%@else%> is optional.
$\langle \mathrm{incl} \rangle$ <%@else%> $\langle \mathrm{excl} \rangle$ <%@endif%>	<pre><%@ifneq%> negates the test.</pre>

Code expressions

Code expressions are evaluated after translation. They may be of any programming language as long as there is a code translator for it. Code expressions have no access to preprocessing variables (except metadata). Output written to standard output is inserted into the final document.

Markup	Description
<%(code)%>	Inserts (code) (may be an incomplete expression) into the translated code script without including content in the output document.
<%=(code chunk)%>	Inserts (code chunk) (must be a complete expression) into the translated code script and includes the returned value in the output document.

Example of text file with RSP-embedded R code

2. Without comments and preprocessed: <%@meta title="Example"%> Title: Example Title: <%@meta name="title"%> Counting: <% for (i in 1:3) { %> <%=i-%> Counting:<% for (i in 1:3) { %><%-%> <% } %> <%=i-%> <% } %>

3. Translated code script:

```
cat("Title: Example\nCounting:")
                                       Title: Example
for (i in 1:3) {
                                       Counting: 1 2 3
cat(" ")
cat(i)
}
```

R.rsp commands

1. RSP document:

```
rcat('Today is <%=Sys.Date()%>')
                                               s <- rstring('Today is <%=Sys.Date()%>')
                                                                                                        output <- rfile('\langle file | URL \rangle')
                                                                                                                                                                           rsource('\langle file | URL \rangle')
rcat(file='\langle file|URL\rangle')
                                               s <- rstring(file='\( file | URL \)')</pre>
                                                                                                        output <- rfile('\(\file|\URL\)', postprocess=FALSE)
```

4. Output document: