

Overview

Declararing Parameters

YAML Params Field

Parameter Types

Using Parameters

Accessing from R

Passing Parameters

Parameter User Interfaces

Parameterized Reports

Overview

R Markdown documents can optionally include one or more parameters. Parameters are useful when you want to re-render the same report with distinct values for various key inputs, for example:

1. Running a report specific to a department or geographic region.
2. Running a report that covers a specific period in time.
3. Running multiple versions of a report for distinct sets of core assumptions.

R Markdown parameter names, types, and default values are declared in the YAML section at the top of the document. To change these values for a given rendering you use the `params` argument to the `rmarkdown::render` function.

Note that parameterized reports are a new feature of R Markdown and therefore require very recent versions of the **knitr** (v1.11) and **rmarkdown** (v0.8) packages. You can install the most up to date versions with the following command:

```
install.packages(c("knitr", "rmarkdown"))
```

If you are using RStudio you should also download the preview release (<https://www.rstudio.com/products/rstudio/download/preview/>) of RStudio (as it has some additional changes required to support previewing reports with parameters).

Declararing Parameters

YAML Params Field

Parameters are declared using the `params` field within the YAML section at the top of the document, for example:

```
---
title: My Document
output: html_document
params:
  region: east
---
```

Parameter values can be provided inline as illustrated above or can be included in a value sub-key. For example:

```
---
title: My Document
output: html_document
params:
  region:
    value: east
---
```

This second form is useful when you need to provide additional details about the parameter (e.g. information about how parameters should be presented to end-users).

Parameter Types

All of the standard R types that can be parsed by the `yaml::yaml.load` function are supported including `character`, `integer`, `numeric`, and `logical`. In addition, you can use arbitrary other R object types by specifying the value using an R expression. For example, to specify a date or date-time you could use this code:

```
---
title: My Document
output: html_document
params:
  start: !r as.Date("2015-01-01")
  snapshot: !r as.POSIXct("2015-01-01 12:30:00")
---
```

Note that the date and data-time values are prefaced with `!r`, which indicates that the value is an R expression rather than a literal value.

Using Parameters

Accessing from R

The declared parameters are automatically made available within the knit environment as components of a read-only list named `params`. For example, the values of the above two parameters can be accessed in a chunk with the following R code:

```
```${r}
params$region
params$start
```
```

When you preview the document in RStudio (or otherwise call the `rmarkdown::render` function with no arguments) the default parameter values listed in the YAML will be used.

Passing Parameters

To vary the parameters of an R Markdown document from the defaults you use the `params` argument of the `rmarkdown::render` function. For example:

```
rmarkdown::render("MyDocument.Rmd", params = list(
  region = "west",
  start = as.Date("2015-02-01")
))
```

You can of course specify only a subset of the available parameters in your call to render. For example:

```
rmarkdown::render("MyDocument.Rmd", params = list(
  region = "west"
))
```

In this case the default values are used for any parameters not explicitly provided.

You might also find it convenient to wrap the call to render in an R function, for example:

```
renderMyDocument <- function(region, start) {  
  rmarkdown::render("MyDocument.Rmd", params = list(  
    region = region,  
    start = start  
  ))  
}
```

Parameter User Interfaces

In the Passing Parameters section above we described wrapping the invocation of a report in an R function to allow customization of its parameters. It's also possible to provide a user-interface for specifying parameter values.

If your document contains parameters and you specify the `params = "ask"` argument to the `render` function then a user-interface is provided to specify the parameter values. For example, consider the following parameter declarations:

```
---  
title: "My Document"  
output: html_document  
params:  
  minimum: 100  
  region: east  
  data: results.csv  
---
```

If you call the `render` function as follows:

```
rmarkdown::render("MyDocument.Rmd", params = "ask")
```

Then you'll see the following user-interface for parameter entry:

127.0.0.1:4383

127.0.0.1:4383

minimum

100

region

east

data

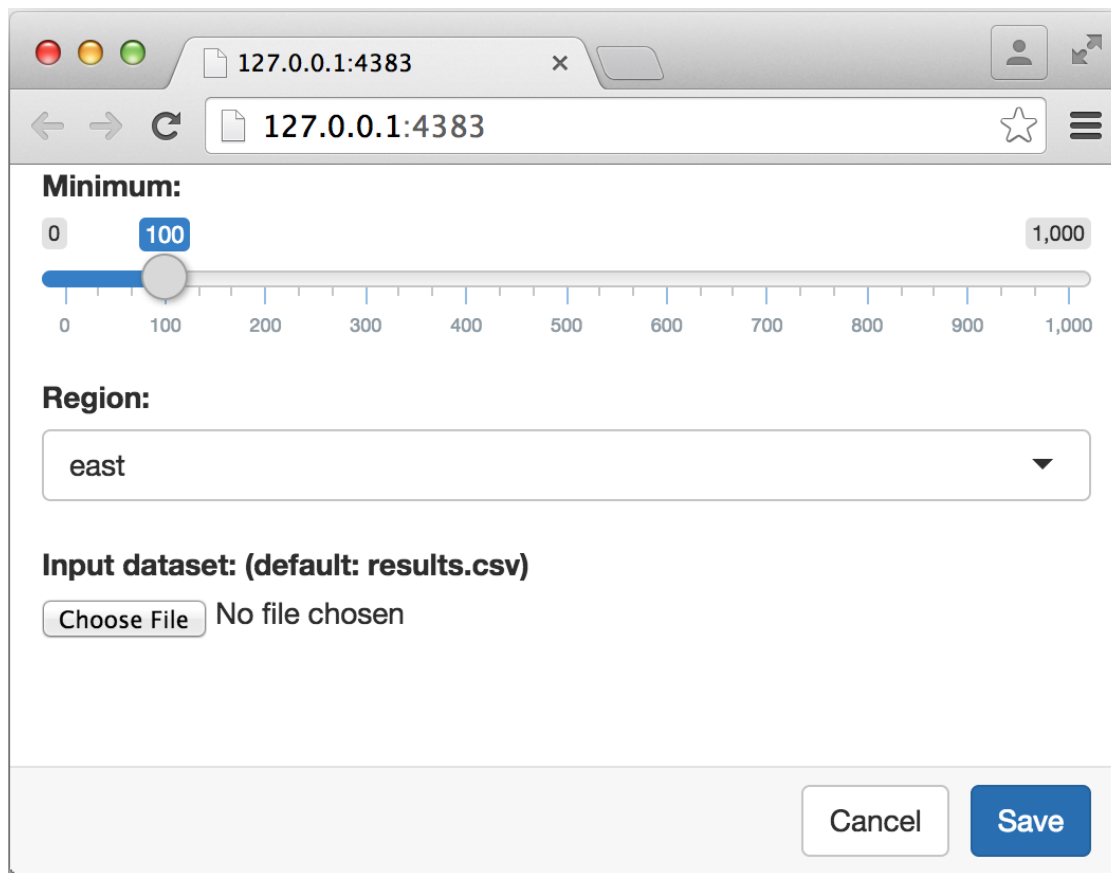
results.csv

Cancel Save

The user-interface is generated using Shiny (<http://shiny.rstudio.com>) and as a result can take advantage of standard Shiny input controls. You can customize inputs by adding the appropriate values to the parameter YAML. For example, the following parameters are customized to use the Shiny `sliderInput`, `selectInput`, and `fileInput` controls:

```
---
title: "My Document"
output: html_document
params:
  minimum:
    label: "Minimum:"
    value: 100
    input: slider
    min: 0
    max: 1000
  region:
    label: "Region:"
    value: east
    input: select
    choices: [east, west, north, south]
  data:
    label: "Input dataset:"
    value: results.csv
    input: file
---
```

This results in the following user-interface for parameter editing:



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:4383'. The page content includes a slider control labeled 'Minimum:' with a range from 0 to 1,000 and a current value of 100. Below the slider is a dropdown menu labeled 'Region:' with 'east' selected. Further down is a file input section labeled 'Input dataset: (default: results.csv)' with a 'Choose File' button and the text 'No file chosen'. At the bottom right are 'Cancel' and 'Save' buttons.

The type of Shiny control used is controlled by the `input` field. The following input types are currently supported (see the help for the associated Shiny function for additional attributes which can be specified to customize the input):

| Input Type | Shiny Function |
|------------|---|
| checkbox | <code>checkboxInput</code>
(http://shiny.rstudio.com/reference/shiny/latest/checkboxInput.html) |
| numeric | <code>numericInput</code>
(http://shiny.rstudio.com/reference/shiny/latest/numericInput.html) |
| slider | <code>sliderInput</code>
(http://shiny.rstudio.com/reference/shiny/latest/sliderInput.html) |
| date | <code>dateInput</code>
(http://shiny.rstudio.com/reference/shiny/latest/dateInput.html) |
| text | <code>textInput</code>
(http://shiny.rstudio.com/reference/shiny/latest/textInput.html) |
| file | <code>fileInput</code>
(http://shiny.rstudio.com/reference/shiny/latest/fileInput.html) |

| | |
|--------|---|
| radio | radioButtons
(http://shiny.rstudio.com/reference/shiny/latest/radioButtons.html) |
| select | selectInput
(http://shiny.rstudio.com/reference/shiny/latest/selectInput.html) |

Note that attributes provided for parameters are automatically passed as arguments to the respective Shiny input functions listed above.

RStudio Preview

When previewing an R Markdown document within RStudio you can use the **Knit with Parameters** command (available from the standard Knit toolbar menu) to specify parameters prior to previewing.

24 Comments

R Markdown

 Login ▾

 Recommend 3

 Share

Sort by Best ▾



Join the discussion...



Sri · 5 months ago

Useful information. Question : Can I define and pass an object (data table or list) as parameter, if yes - what is the syntax?

1 ^ | v · Reply · Share ›



Thomas Neitmann · 6 months ago

Is there a possibility to select multiple files via shiny and then get the file names as strings?

1 ^ | v · Reply · Share ›



Philipp · a month ago

is it possible to define a subset of the parameters via the render function and ask for all others?

^ | v · Reply · Share ›



jjallaire Mod → Philipp · a month ago

No that isn't possible right now.

^ | v · Reply · Share ›



Philipp → jjallaire · a month ago

too bad, but thx! :(is that something that will be added in the

future? is it a common request?

^ | v · Reply · Share ›



Gabi Huiber · a month ago

I got into the habit that now I ``rmarkdown::render()`` straight-up R scripts. I only need to write my comments in Roxygen2 style and put some yaml on top. But this parameterized `render()` doesn't seem to work with R scripts as described here. I mean that it only works if I declare a params list by hand in the global environment, with properly named arguments. The `params = 'ask'` recipe doesn't work at all. Should it? Am I missing something?

^ | v · Reply · Share ›



Philipp · 2 months ago

when using `params = "ask"`, can the shiny user interface for parameter input be customized?

^ | v · Reply · Share ›



W. Duncan Wadsworth · 2 months ago

Funny story. I'd like to load some simulations and process them with an `.Rmd` script but within each `.RData` file I load there is an object named ``params``. So I get the error: "Error: cannot change value of locked binding for 'params' Execution halted". Is there some way around this?

^ | v · Reply · Share ›



Christine Herlihy · 2 months ago

Can you put in a vector instead of a single string for a parameter? (For example, instead of: `region: east`, can you put in `region: c("east", "west", "north", "south")` and then auto-generate a report for each of these regions, filling in the correct/corresponding data?

^ | v · Reply · Share ›



David Holstius · 3 months ago

Are params inherited by child documents? If a child also defines the same param as its parent, which takes precedence?

^ | v · Reply · Share ›



Yihui Xie Mod → **David Holstius** · 3 months ago

The params should be inherited by child documents. Params defined in child documents will be ignored if the parent document is being rendered.

^ | v · Reply · Share ›



MarcelMan · 4 months ago

Re the last example, how do you then access the input file?



ne the last example, how do you then access the input file :

read.csv(params\$data) gives me the following: Error in file(file, "rt") : cannot open the connection

^ | v · Reply · Share ›



sunewlsu → MarcelMan · 2 months ago

My script loads my data and runs if I use the default value for my data parameter.

params:

label: "Count dataset:"

value: counts.txt

input: file

But, if I use the choose file button to select the same file, the script fails with the error you noted.

Error in file(file, "rt") : cannot open the connection

Has anyone found a solution to this problem?

^ | v · Reply · Share ›



MarcelMan → sunewlsu · 2 months ago

I haven't found a solution, but from what I think I understand, the temporary file created when loading the file of your choosing gets erased before the script can access it. Loading the default works because there is no temporary file needed.

^ | v · Reply · Share ›



MarcelMan → sunewlsu · 2 months ago

It seems to me that the temporary file created when loading gets erased before the script can read it, hence the error message. I have asked the question at stackoverflow (<http://stackoverflow.com/quest...> but failed to get attention on this issue. Still stuck here :-(

^ | v · Reply · Share ›



alain content · 7 months ago

Hi,

very nice and useful addition !

Why "!r" rather than `r.` as in inline rmarkdown ? isn't that potentially confusing ?

^ | v · Reply · Share ›



jjallaire Mod → alain content · 7 months ago

We used the !r syntax because that's the standard for the YAML package. That said, I agree that most users won't appreciate that

package. That said, I agree that most users won't appreciate that subtlety and we may create some confusion. Perhaps we can just support both.

^ | v · Reply · Share ›



alain content → jjallaire · 7 months ago

Hmm. Thinking a bit more, I wonder. Actually `r` and !r are doing two different things, right ? If I understand correctly !r causes assignment of the result of the following expression to the param name, whereas `r` indicates that the enclosed expression is to be interpreted.

^ | v · Reply · Share ›



jjallaire Mod → alain content · 7 months ago

Both would have a similar result (assign the R expression to the parameter). Now that I've thought about it some more I realize that one reason we don't support the embedded R with back ticks is that the YAML can read and interpreted without calling knitr so we'd have to add an extra processing step on the YAML to make this work.

^ | v · Reply · Share ›



Louis Springer · a year ago

Is there a complete example of parameter usage somewhere?

^ | v · Reply · Share ›



jjallaire Mod → Louis Springer · a year ago

What beyond the examples provided above are you looking for?

^ | v · Reply · Share ›



Antonio Piccolboni → jjallaire · 7 months ago

I suspect that the Accessing from R section is a little too terse. You could show a small R markdown doc that uses parameters. I suspect params is just in scope for every code chunk. But there isn't a single code chunk in this whole web page. This would be my only problem with this page, and I suspect it may be related to what Louis is reporting

^ | v · Reply · Share ›



jjallaire Mod → Antonio Piccolboni · 7 months ago

Agreed that the examples are light. We have a complete revision of this document coming to reflect some additional capabilities soon and we'll be sure to have

