

Module 6 Pre-Read - config and GNU Make

Sunday, February 25, 2024 12:44

To create a variable in git bash (environment variable):

```
export SOME_VARIABLE="some value"
echo $SOME_VARIABLE
```

Then, to pass it into an R script, type  
sys.getenv("SOME\_VARIABLE")

The config package

touch config.yml  
Creates a yamli file, inside we will specify our parameters  
e.g. inside the file, write

```
1 default:
2   parameter1: 10
3   parameter2: 20
4   parameter3: "some value"
```

To get the config parameters in an rmd, here is an example:

```
1 ---
2 title: Test out the config package
3 output: html_document
4 ---
5
6 ```{r, read-config}
7 config_list <- config::get()
8
9
10 The value of parameter1 from our config file is 'r config_list$parameter1'.
11 The value of parameter2 from our config file is 'r config_list$parameter2'.
```

To get the config parameters in an r script, here is an example:

```
18 WHICH_CONFIG <- Sys.getenv("WHICH_CONFIG")
19 config_list <- config::get()
20 config = WHICH_CONFIG
21
22
23 binary_and <- glef
24 1/lab_resistance ~ config_list$outpoint) ~ shield_glycam ~ region ~ env_length,
25 data = data,
26 family = binomial())
27 )
```

To update the names of the save files to reflect config parameters:

```
38 # e.g., active config is default
39 # saved file will be called both_models_config_default.rds
40 both_models_filename <- paste0(
41   both_models_config, "-"
42   , "rds"
43 )
44
45 saveRDS(
46   both_models,
47   file = here::here("output", both_models_filename)
48 )
```

To update the name of the rendered html file:

```
1 here:::("code/04_render_report.R")
2
3 WHICH_CONFIG <- Sys.getenv("WHICH_CONFIG")
4 config_list <- config::get()
5 config = WHICH_CONFIG
6
7
8 library(markdown)
9 report_filename <- paste0(
10   "hiv_report_config_",
11   WHICH_CONFIG,
12   ".html"
13 )
14 # rendering a report in production mode
15 render(
16   "hiv_report.Rmd",
17   output_file = report_filename
18 )
```

GNU Make

Make an empty file called Makefile  
touch Makefile

In Makefile, to write a rule, here is an example:  
output/table\_clean.rds: code/00\_clean\_data.R raw\_data/vrc01\_data.csv  
[press enter and then the tab key, NOT A SPACE] Rscript code/00\_clean\_data.R

Then in git bash write  
make output/data\_clean.rds

A full example:  
output/table\_one.rds: code/01\_make\_table1.R output/data\_clean.rds  
Rscript code 01/make\_table1.R

Here, our table\_one object inherits the stuff from data\_clean object, so if something changes in code/00\_clean\_data.R, then when we try to make table\_one, everything will be rebuilt accordingly, including data\_clean dependencies

To remove all the output:  
.PHONY: clean  
clean:  
rm -f output/\*.rds && rm -f output/\*.png  
Then in bash: make clean

We need the phony because otherwise GNU make thinks ur looking for a file called clean.

Grouped targets:

```
13 output/both_models_config_default.rds output/both_regression_tables_config_default.rds&: \
14   03_models.R output/data_clean.rds
15   Rscript code/03_models.R
16
17 .PHONY: clean
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
```

```
1 ## hiv_report_config_default.html: This is the final report.
2 hiv_report_config_${WHICH_CONFIG}.html: code/04_render_report.R \
3   hiv_report.Rmd descriptive_analysis regression_analysis
4   Rscript code/04_render_report.R
```

Commands executed in sequence:

```
10 ## output/table_one.rds: This is the table 1 summary created using gtsummary
11 output/table_one.rds: code/01_make_table1.R
12   Rscript code/00_clean_data.R && \
13   Rscript code/01_make_table1.R
```