Implemention of an R Environments

Liam O'Suilleabhain

YHAT

February 04, 2019

Outline

git clone https://github.com/losuilleabhain/AMORE.git

Introduction

Environment

Application

Implementation

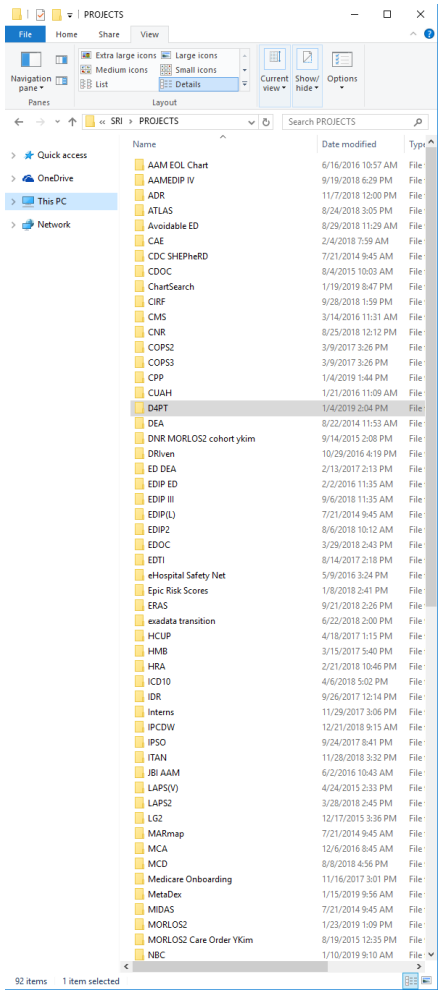
Motivation

Reproducibility necessitates a
Standard Environment

A Project has 3 sets of information

1. Resources
2. Operations

   A Functional Program maps data to output
3. Results



Many Shared Projects
Many Project Structures

?Startup

Description:

In R, the startup mechanism is as follows.

Unless '–no-environ' was given on the command line, R searches for site and user files to process for setting environment variables. The name of the site file is the one pointed to by the environment variable 'R_ENVIRON'; if this is unset, 'R_HOME/etc/Renviron.site' is used (if it exists, which it does not in a 'factory-fresh' installation). The name of the user file can be specified by the 'R_ENVIRON_USER' environment variable; if this is unset, the files searched for are '.Renviron' in the current or in the user's home directory (in that order). See 'Detailsr" for how the files are read.

Then R searches for the site-wide startup profile file of R code unless the command line option '–no-site-file' was given. The path of this file is taken from the value of the 'R_PROFILE' environment variable (after tilde expansion). If this variable is unset, the default is 'R_HOME/etc/Rprofile.site', which is used if it exists (which it does not in a 'factory-fresh' installation). This code is sourced into the 'base'package. Users need to be careful not to unintentionally overwrite objects in 'base', and it is normally advisable to use 'local' if code needs to be executed: see the examples.

Then, unless '–no-init-file' was given, R searches for a user profile, a file of R code. The path of this file can be specified by the 'R_PROFILE_USER' environment variable (and tilde expansion will be performed). If this is unset, a file called '.Rprofile' is searched for in the current directory or in the user's home directory (in that order). The user profile file is sourced into the workspace.

An R Environment

1. R_ENVIRON — R_HOME/etc/Renviron.site
2. R_ENVIRON_USER — ~/.Renviron
3. R_PROFILE — GROUP_HOME/Rprofile.site
4. R_PROFILE_USER — ~/.Rprofile

Files in blue are available on github

One R Environment

Platform	Development	Notebooks	Datasets
Machine	DOR Desktop	KPIT Server	DOR Server
Processor	2.5GHz 4 Core	2.3GHz 8 Core	3.0GHz 32 Core
Memory	16GB	64GB	256GB
Storage	Unlimited	1TB	1TB
Support	DOR IT	KPIT	DOR IT

Example: Set Environment Variable

Example ∼/.Renviron on Unix

R_LIBS="∼ /R/library"

Example .Renviron on Windows

R_LIBS="C:/R/library"

R_ENVIRON - Global Environment (Optional)

$(R RHOME)/etc/Renviron.site

Create a .Renviron file - ~/.Renviron

Set R_ENVIRON_USER

R_ENVIRON_USER - User Environment

~/.Renviron

Create a group profile - GROUP_HOME/Rprofile.site

Set R_PROFILE

Create a shared library directory - GROUP_HOME/R_LIBS

Set R_LIBS_SITE

Create a personal library directory - ~/R_LIBS

Set R_LIBS_USER

R_PROFILE - Group Profile

GROUP_HOME/Rprofile.site

Create .Rprofile - ~/.Rprofile

Set R_PROFILE_USER

Create Project .Rprofiles e.g.
GROUP_HOME/RPROFILE/.RPROFILE

Create Function to source Project Environments

```
> Renv <- function(x){
+  switch(x,
+         RPROFILE = source(paste0(GROUP_HOME,"RPROFIL
+ }
> Renv('RPROFILE')
>
```

R_PROFILE_USER - User Profile (Freedom!)

~/.Rprofile

Create Development Directory e.g.
~/Development/
Set Development Directory

Create Password Vault
~/pwv.txt
Load Passwords

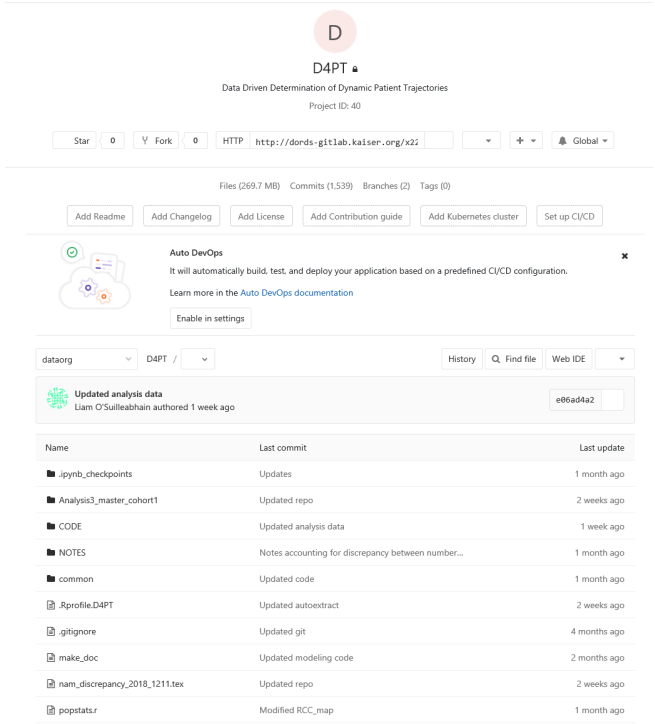
A Project Profile

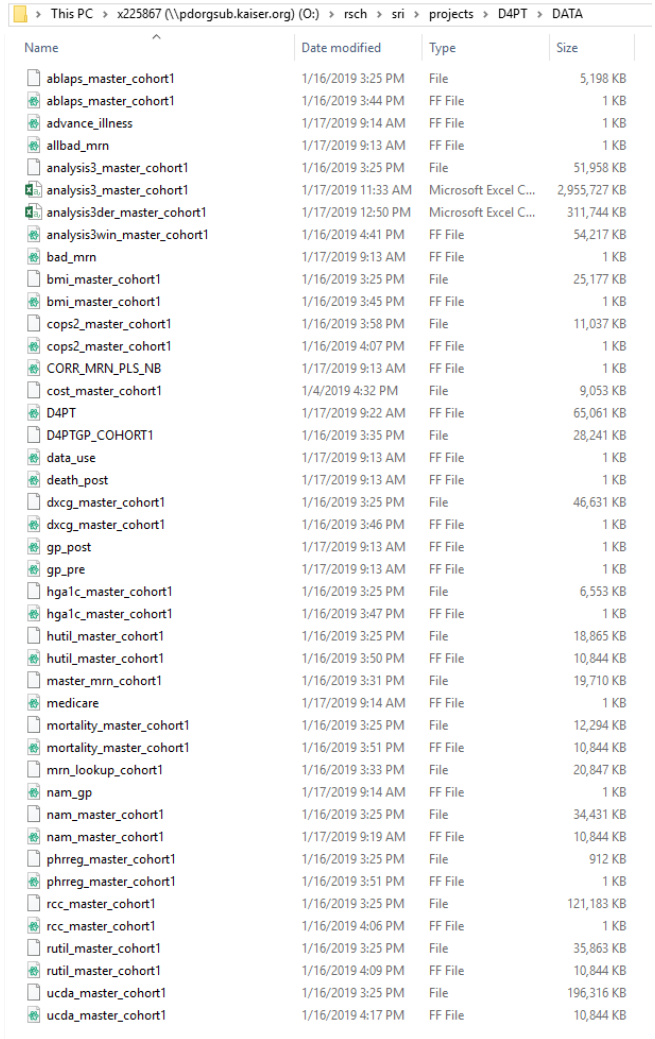
GROUP_HOME/RPROFILE/Rprofile.RPROFILE

Create Resource Directory
Create Code Directory
Create Analysis Directory

Load Libraries
Set Project Options
Load Project Resources
Set Database Connections

Content Management





Project Workflow

Option 1:

Reference Files and Operations Relative to project path e.g. source("./CODE/analysis/models.r")

Option 2:

Separate git controlled code from the project e.g. ource(paste0(DATA,"analysis/models_output"))

Suggestions

Only manage code that creates data or analysis

Management

- Content
- Structure

Maintain same format for Code and Analysis directories

Maintenance

- Operations - Code
- Output - Analysis

Future

Option 1 vs. Option 2

- Option 1 is less verbose
- Option 2 provides extra flexibility
- Option 2 isolates code maintenance

Develop Packages for Healthcare Analytics

Maintain git repositories with open-source code

Acknowledgements

Thanks:

SRI for supporting this framework.

Chris Paciorek for ideas and advice.

Raj, Alejandro, Gina and Brian for helpful conversations.