

Using Bundles for R Package Management

Magnus Mengelbier
Managing Director

Topics

- Introduction
- Package management basics
- Project package managers
- Bundles
- Conclusion

The R Language

- R is an abstract programming language for statistical computing and graphics ...
- Interpreted scripting language
- Object oriented
- **Package** – Distributable collection of re-usable objects, methods and functions
- **Library** – Collection of packages
- Packages can depend on other packages (*dependency chain* and *reverse dependencies*)
- Native support for compute logic written in languages other than R (*tool chain*)
 - Packages logic
 - Shared libraries
- Generic methods and functions
- Redefine anything

Example Package Library

- Basics data manipulation
- Import/export Microsoft Excel
- Survival analysis
- Mixed Models
- Indexing and Scoring
- General output format/file types
- Data models



348 packages

A Simple Package Request

- 2 new packages
- ... that has 8 dependencies ... so now 10
- ... 4 dependencies already exist ... so now 6
- ... 2 dependencies need to updated to a later version ... so now 8
- ... 1 dependency is a minor update with 2 new dependencies ... 10
- ... 18 packages share dependencies on the 2 existing ... now 28
- ... of those 18 ... 9 have new versions with new dependencies ...



34 packages to validate

Package Management

- ✓ Adding features and capability
- ✓ Conflict management
- ✓ Maintaining validated state

Standard R Libraries

Library

Minimum set of packages required for R to function.

`.Library`

Site library

Additional packages to support analysis and reporting, which may include packages from CRAN, Bioconductor, internal sources, etc.

`.Library.site`

User library

Directory accessible from within R, most commonly in the user home directory.

`$R_LIBS_USER`

- Any other location on the condition
 - Accessible from within your R session
 - Can be added to the library tree, i.e. `.libPaths()`

Central and Shared versus Personal and Private

Library

Minimum set of packages required for R to function.

`.Library`

Site library

Additional packages to support analysis and reporting, which may include packages from CRAN, Bioconductor, internal sources, etc.

`.Library.site`

User library

Directory accessible from within R, most commonly in the user home directory.

`$R_LIBS_USER`

- Any other location on the condition
 - Accessible from within your R session
 - Can be added to the library tree, i.e. `.libPaths()`

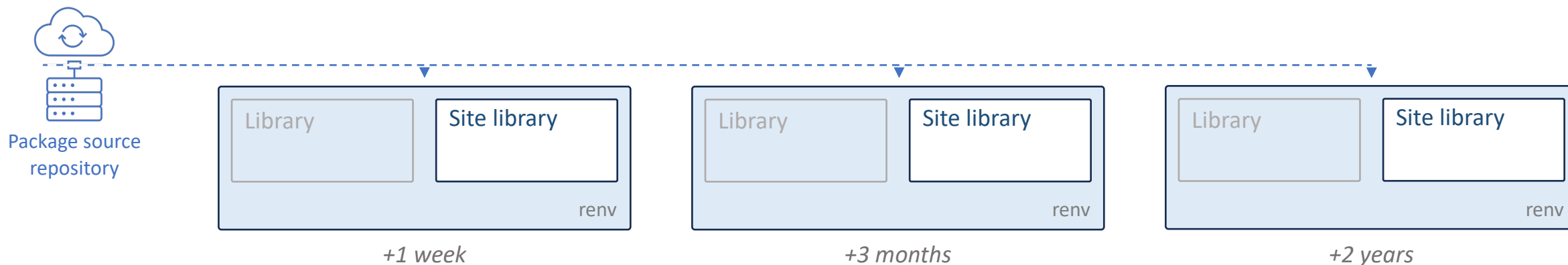
Central & Shared

Disruption

Personal & Private

Reproducibility

Project Package Managers



- Each analysis team selects and installs required packages for project analysis
- Users permitted to install and manage packages
 - Common is a source repository of *validated* packages
 - Configuration can be imported and/or exported (example `renv.lock` file)
 - Resolves disruption and reproducibility issues
 - Makes R modular (*GAMP v5 Category 4 Configurable Software*)
- Complex processes to maintain that a package is validated across time
- Sensitive to IT updates and security patches (tool chain dependencies)

A User's Perspective of Bundles

```
cxlib_bundle_load( "outputs/1.1.1", user.library = FALSE )
```

```
cxlib_bundle_load( "outputs", user.library = TRUE )
```

- A bundle is “loaded”
- *Named*, as in *outputs*
 - Multiple bundles
 - Designed for a particular purpose
 - Aligned with organization, function, process, analysis, etc.
 - Give it a *license for use*, is it GxP, exploratory, a pilot, for decisions
 - Narrower scope of use simplifies maintenance and validation
- Versioned, *that would be 1.1.1*
 - *Concept of a default version (second statement)*
 - *Minimizes disruption and improves reproducibility*

Loading a Bundle

```
> .libPaths()
[1] "/home/user/R/x86_64-pc-linux-gnu-library/4.0" "/opt/R/4.0.3/lib64/R/site-library"
[3] "/opt/R/4.0.3/lib64/R/library"
>
> cxlib_bundle_load( "outputs/1.1.1" )
>
> .libPaths()
[1] "/home/user/R/x86_64-pc-linux-gnu-library/4.0" "/opt/bundles/outputs/1.1.1/libraries/R/4.0.3"
[3] "/opt/R/4.0.3/lib64/R/site-library"           "/opt/R/4.0.3/lib64/R/library"
>
```

- A bundle is a dynamically loadable R package library
- Standard R mechanisms for finding and accessing packages



Just manipulating the library tree, e.g. `.libPaths()`

Bundle Convention

```
cxlib_bundle_load( "outputs/1.1.1", user.library = FALSE )
```

/opt/bundles/**outputs/1.1.1**/libraries/R/4.0.3

Install Directory

Bundle R Package Library

Multi-lingual
(e.g. not just R)

- Install directories added to bundle search tree
- First occurrence of bundle is used
- R bundle identified by `libraries/R` (case in-sensitive)
- Support for multiple versions of R
 - Compliance requirements
 - Tool chain dependencies

Creating a Bundle

```
# create the bundle package library directory
dir.create( "/opt/bundles/outputs/2.3.4/libraries/R/4.0.3", recursive = TRUE )

# -- note: create any property files here if your bundle has dependencies

# load the emmpty bundle
# note: user.library is disabled so that any user library packages are ignored
cxlib_bundle_load( "outputs/2.3.4", user.library = FALSE )

# install packages
install.packages( ... )
```

- R package library within a bundle is just another directory
- Standard R package install program methods
- Maintenance process is install - validate - use
- Bundle loading is accessing a pre-installed R package library
- R is *static* application (GAMP v5 Category 3 Non-configurable software)
- Simplified maintenance and validation process (e.g. Capture-n-Replay)

Bundle Deployment

```
# deploy outputs  
# note: assuming that the deploy sources have been defined in options  
mydeploy <- cxlib_bundle_deploy( "outputs/2.3.4" )
```

- Central controlled source or file-based repository
- Independent of environment architectures and structures
- Cross-environment deployments
- Controlled deployments
 - Validated transfers
 - Controlled package installation
- Consistency
 - Packages
 - Package versions

Bundle Specifications

```
# create initial bundle reference
# note: ... or read in an existing specification
mybundle <- cxlib_bundle( name = "outputs")
mybundle$version( "2.3.4" )

# disable user library
mybundle$option( "r/user.library" = FALSE )

# add packages
mybundle$add.packages( ... )

# create the specification file
mybundle$save()
```

- Specifications and install sources
- Self-documenting
- Reproducible
- Shareable

In Summary

- Package Management is
 - Adding features and capabilities
 - Managing conflicts
 - Maintaining validated state
 - Making R packages accessible to your users
- Avoiding disruption
- Improving reproducibility
- Consistency
- Simplified maintenance and validation
- Sharing

Magnus Mengelbier

magnus.mengelbier@limellogic.com

Limellogic AB

Jungmansgatan 12

SE-211 11 Malmö

Sweden

www.limellogic.com

github.com/limellogic