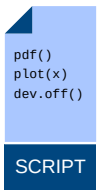
The conduit Package

Ashley Noel Hinton & Paul Murrell

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

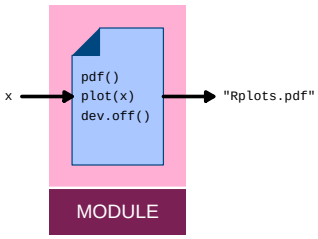
> sessionInfo()
[1] "June 30 - July 3, 2015"
[2] "Aalborg, Denmark"

SCRIPTS



MODULES

```
module("m", language = "R",
       inputs = list(
           moduleInput("x", internalVessel("x")))),
       sources = list(
           moduleSource(fileVessel("script.R"))),
       outputs = list(
           moduleOutput("Rplot",
                        fileVessel("Rplots.pdf"))))
```



CONDUIT

The **conduit** package for R provides a system for combining R **SCRIPTS**.

A script is first embedded within a **MODULE** wrapper, which defines the inputs required by the script and the outputs that the script produces.

Modules are then combined into **PIPELINES** by specifying connections ("pipes") from the outputs of one module to the inputs of another module.

The **conduit GLUE SYSTEM** runs a pipeline by executing the scripts within the modules and passing results from one module to the next.

The architecture of the package is designed to maximise the independence of scripts, modules, and pipelines so that, for example, the author of a script can be distinct from the author of a module, who can in turn be distinct from the author of a pipeline.

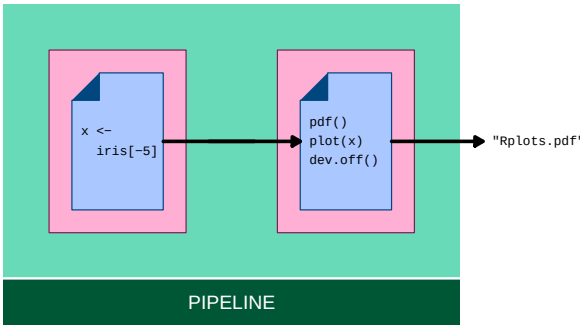
Importantly, the author of a script can have no knowledge of **conduit**, so that scripts written originally for one purpose can be repurposed and reused via the **conduit** package.

Furthermore, modules can be created to wrap scripts in other languages, such as Python.

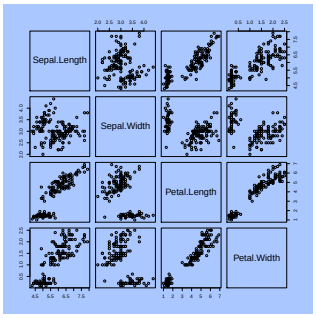
The motivation for the package is to support greater use of public data sets by facilitating and encouraging the creation, reuse, and recombination of small scripts that perform simple tasks.

PIPELINES

```
pipeline("p",
         modules = list(m1, m2),
         pipes = list(pipe("m1", "x", "m2", "x")))
```



OUTPUTS



GLUE SYSTEM

```
> p1 <- loadPipeline("p1", "pipeline.xml")
> runPipeline(p1)

       $m2
       $m2$Rplot
       $m2$Rplot$name
       [1] "Rplot"

       $m2$Rplot$type
       [1] "fileVessel"

$m2$Rplot$object
[1] "/tmp/RtmpVHD4wt/pipelines/p1/modules/m2/Rplots.pdf"
```

ACKNOWLEDGEMENTS AND FURTHER READING

This work was partially funded by a University of Auckland, Faculty of Science, Research Development Fund and an Ockham Foundation Postgraduate Scholarship.

conduit github page
https://github.com/anhinton/conduit

The **conduit** package is an implementation of the OpenAPI architecture. The following is an extract from the document "Introducing OpenAPI".

What is OpenAPI

The problem: connecting people with data

OpenAPI's main aim is to make it easier for people to connect with data. Connecting with data is not just a simple case of making raw data available to people (though this is one component of it). When people are fully connected with data they can examine, manipulate, display, compare, interpret and share it. We believe the following things are required to connect people with data:

- Access to data
- Domain knowledge
- Data Science skills
- Statistical Graphics skills
- Graphical Design skills

While many people possess one or several of these attributes, it is very rare to find someone who possesses all of them. Access to data is getting easier thanks to initiatives to make data widely and freely available, like Open Data, Open Government and Open Access. Domain knowledge, data science skills, statistical graphics skills and graphical design skills can all be acquired through education and experience. OpenAPI intends to help connect with data those people who do not have the luxury of acquiring special skills and knowledge. OpenAPI will do this by allowing anyone to contribute using those skills that they already possess.

Possible solutions

One solution to the problem of connecting people with data is providing software or resources containing all of the required attributes on the user's behalf. Some examples of this are:

Public visualisation services
: There are a variety of services which present data in visual form for non-experts to use. Some examples of this are Wiki New Zealand (`http://wikinewzealand.org/`) and Gapminder (`http://www.gapminder.org/`).

Visual programming
: Two significant obstacles to connecting people with data are data science skills and statistical graphics skills. Visual programming software attempts to overcome these obstacles by providing graphical interfaces to common data analysis jobs. This way a user can do an analysis by choosing different options from the interface without having to work with any scripts or code.

  Visual programming makes massive demands on its authors to ensure it covers all the uses that may be required, and to make sure that the entire program works well. If some piece of analysis or some aspect of an analysis is not coded into the visual programming software it is no small task for a user to have it included.

Comprehensive solutions for advanced users
: We recognise that there appears to be some similarities between the OpenAPI system and some of the advanced comprehensive solutions that already exist. One such example is the Galaxy platform for bio-medical research. A discussion of how these systems differ from OpenAPI in both their intentions and their execution can be found in the full version of this document (see the link below).

The OpenAPI solution: everyone contributes a small amount

Rather than attempt to capture all of the necessary attributes in one program or service, OpenAPI attempts to capture small contributions which can be combined to suit the user. This means that contributors only require some of the attributes that we have listed. This allows for small contributions like:

- A contributor with raw data can make this data available to other OpenAPI users.
- A domain expert can clean up data and annotate it.
- Someone with data analysis skills can produce meaningful statistics about the data.
- A statistical graphics guru can provide a script to make suitable charts for a data type.
- The graphical design team at a news blog can take graphical output and manipulate it to produce well designed images for their articles.

In OpenAPI each of these contributions is meaningful. Further, they allow for someone who has none of the required attributes to combine other users' contributions in a meaningful way. Even if a contributor has no particular facility in any of the areas mentioned so far, she can still contribute my making data and scripts available to other users of OpenAPI.

At its simplest OpenAPI is about modules. Each module can ask for inputs, describe some work to be done, and produce some outputs. Modules can be combined in pipelines which plug one module's outputs into another module's inputs. The small contributions listed above could each be captured in a module, and a further contribution would combine them as a pipeline.

The OpeAPI architecture defines an XML format for describing modules and pipelines and it also defines the requirements for a glue system that can read and execute modules and pipelines.

Further reading

The full version of this document can be found at `http://stattech.wordpress.fos.auckland.ac.nz/2015-01-introducing-openapi/`.