**Yihui Xie** es actualmente un ingeniero de software en RStudio (http://www.rstudio.com). Obtuvo su doctorado en el Departamento de Estadística, Universidad del Estado de Iowa. Se interesa por los gráficos estadísticos interactivos y computación estadística. Como un usuario activo R, es autor de varios paquetes de R, como la animación, knitr, el formato, la diversión, la mímica, más alto, servir y Rd2roxygen, entre los cuales el paquete de animación ganó el Premio Software John M. Salas de Estadística 2009 (ASA) y el paquete knitr fue galardonado con la "Mención de Honor" premio en las "Aplicaciones de R en competencia del negocio 2012", gracias a la revolución de Analytics.

En 2006, fundó la "Capital de la estadística" (http://cos.name), que ha crecido hasta convertirse en una gran comunidad online sobre las estadísticas en China. Se inició la primera conferencia R China en 2008, y ha sido la organización de conferencias R en China desde entonces. Durante su formación de doctorado en la Universidad Estatal de Iowa, ganó el Premio Vince Sposito Estadística Informática (2011) y el Premio de Snedecor (2012) en el Departamento de Estadísticas.

**Introducción**

La idea básica detrás de documentos dinámicos se deriva de programación literaria, un paradigma de programación concebida por Donald Knuth (Knuth, 1984). La idea original era principalmente para la escritura de software: mezclar el código fuente y la documentación juntos; Tenemos la posibilidad de extraer el código fuente a cabo (llamado maraña) o ejecutar el código para obtener los resultados compilados (llamada de la armadura). Un documento dinámico no es del todo diferente de un programa de ordenador: un documento dinámico, tenemos que ejecutar los paquetes de software para compilar nuestras ideas (a menudo implementado como código fuente) en la salida numérica o gráfica, e insertar el resultado en nuestros escritos literales (como documentación).

Si conoces un lenguaje de programación web como PHP (que puede incrustar código de programa en documentos HTML), esta idea debería resultar familiar. El ejemplo anterior muestra la salida de código en línea, lo que significa que el código fuente está en línea mezclado con una oración. El otro tipo de salida es la salida trozo, que da los resultados de todo un bloque de código. La salida fragmento tiene mucha más flexibilidad; por ejemplo, podemos producir gráficos y tablas a partir de un trozo de código.

Si tuviéramos que hacerlo a mano, tendríamos que abrir R, pegue el código en la consola R para elaborar la trama, guardarlo como un archivo PDF, y la inserta en un documento LATEX con includegraphics \ {}. Esto es tanto tedioso para el autor y difícil de mantener - suponiendo que queremos cambiar la semilla aleatoria en set.seed (), aumentar el número de pasos, o usar un diagrama de dispersión en lugar de un gráfico de líneas, vamos a tener que actualizar tanto el código fuente y la salida. En la práctica, el cálculo y el análisis pueden ser mucho más complicado que el ejemplo de juguete en la Figura 1.1, y se requerirá más trabajo manual en consecuencia.

Es por eso que necesitamos el paradigma de programación literaria. En este paradigma, un autor tiene dos tareas:

1. escribir el código de programa para hacer la computación, y

2. escribir narrativas para explicar lo que está haciendo el código del programa

Técnicamente, la programación literaria consta de tres pasos:

1. analizar el código del documento y separar el código de las narrativas

2. ejecutar código fuente y retornar los resultados

3. Los resultados de mezcla a partir del código fuente con las narrativas originales

Estos pasos se pueden implementar en los paquetes de software, por lo que los autores no tienen que hacerse cargo de estos detalles técnicos. En su lugar, sólo controlamos lo que la salida debe ser similar. Hay muchos detalles que podemos sintonizar para un informe (sobre todo para los informes relacionados con el análisis de datos), aunque la idea de programación literaria parece ser simple. Por ejemplo, los informes de datos a menudo incluyen tablas, y en la Tabla 1.1 se muestra una tabla generada a partir del código R a continuación, utilizando la función de tabla () en knitr

**characteristics document of Reproducible research**

Los resultados de la investigación científica tienen que ser reproducibles para ser digno de confianza. No queremos un hallazgo que es simplemente debido a un suceso aislado, por ejemplo, sólo un investigador de laboratorio específico puede producir los resultados en una fecha específica, y nadie más puede producir los mismos resultados en las mismas condiciones.

la investigación reproducible (RR) es un posible subproducto de documentos dinámicos, pero los documentos dinámicos no garantizan absolutamente RR. Debido a que por lo general hay ninguna intervención humana cuando se genera un informe de forma dinámica, es probable que sea reproducible, ya que es relativamente fácil de preparar el mismo entorno de software y hardware, que es todo lo que necesitamos para reproducir los resultados. Sin embargo, el significado de reproducibilidad puede estar más allá de la reproducción de un resultado específico o un informe en particular. Como un ejemplo trivial, uno podría haber hecho una simulación de Monte Carlo con una cierta semilla aleatoria y conseguimos una buena estimación de un parámetro, pero el resultado fue en realidad debido a una semilla aleatoria "suerte". Aunque podemos reproducir estrictamente la estimación, en realidad no es reproducible en el sentido general. Existen problemas similares en los algoritmos de optimización, por ejemplo, diferentes valores iniciales pueden dar lugar a diferentes raíces de la misma ecuación.

**Literature**

La termino de investigación reproducible fue propuesto por primera vez por Jon Claerbout de la Universidad de Stanford (Fomel and Claerbout, 2009). La idea es que el producto final de la investigación no es sólo el papel en sí, sino también el entorno computacional completo utilizado para producir los resultados en el papel como el código y los datos necesarios para la reproducción de los resultados y sobre la base de la investigación.

D. Donoho

Un artículo sobre la ciencia computacional en una publicación científica no es la beca en sí, que no es más que la publicidad de la beca. La beca real es el entorno de desarrollo de software completo y el conjunto completo de instrucciones que generaron las figuras.

Esto fue bien dicho! Afortunadamente, las revistas se han estado moviendo en esa dirección también. Por ejemplo, Peng (2009) proporcionan instrucciones detalladas a los autores de los criterios de reproducibilidad y la forma de presentar los materiales para reproducir el artículo en la revista Bioestadística

A nivel técnico, RR suele estar relacionada con la programación ilustrada (Knuth, 1984), un paradigma concebido por Donald Knuth para integrar código de computadora con la documentación del software en un solo documento. Sin embargo, las primeras implementaciones como WEB (Knuth, 1983) y noweb (Ramsey, 1994) no eran directamente adecuado para el análisis de datos y generación de informes. Hay otras herramientas en este camino de la generación de documentación, como roxygen2 (Wickham et al., 2015), que es una implementación de R Doxygen (van Heesch, 2008). Sweave (Leisch, 2002) fue una de las primeras implementaciones para tratar los documentos dinámicos en R (Ihaka y Caballero, 1996; R Equipo Central, 2015). Todavía hay una serie de problemas que no fueron resueltos por las herramientas existentes; por ejemplo, Sweave está estrechamente ligada al látex y difícil de extender. El paquete knitr (Xie, 2015b) se basa en las ideas de las herramientas anteriores con un rediseño marco, que permite un control fácil y multa de muchos aspectos de un informe. Vamos a introducir otros instrumentos en el capítulo 16.

Una visión general de la programación ilustrada aplicada a análisis estadístico se puede encontrar en Rossini (2002). Caballero y Temple Lang (2004) introdujo los conceptos generales de los documentos de programación literaria para el análisis estadístico, con una discusión de la arquitectura de software. Gentleman (2005) es un ejemplo práctico basado en caballero y Temple Lang (2004), usando un paquete de R GolubRR para distribuir el análisis reproducibles. Baggerly et al. (2004) puso de manifiesto varios problemas que pueden surgir con la práctica habitual de la publicación de resultados de análisis de datos, lo que puede dar lugar a falsos descubrimientos debido a la falta de detalles sobre la reproducibilidad (Incluso con conjuntos de datos suministrado). En lugar de separar los resultados de la computación, podemos poner todo en un solo documento (llamado un compendio de Caballero y Temple Lang (2004)), incluyendo el código de ordenador y narrativas. Cuando compilamos este documento, el código de computadora se ejecutará, dándonos los resultados directamente.

**Good and Bad Practices**

La clave a tener en cuenta para el RR es que otras personas deben ser capaces de reproducir nuestros resultados, por lo tanto, debemos hacer todo lo posible para hacer que nuestra computación portátil. Se discuten algunas buenas prácticas para el RR a continuación y explicar por qué no puede ser malo para seguirlos.

• Gestionar todos los archivos de origen en el mismo directorio y utilizar rutas relativas siempre que sea posible: rutas absolutas pueden romper reproducibilidad, por ejemplo, un archivo de datos como C: /Users/john/foo.csv o /home/joe/foo.csv solamente puede existir en una computadora, y otras personas pueden no ser capaces de leerlo ya que es probable que sea diferente en su disco duro la ruta absoluta. Si mantenemos todo bajo el mismo directorio, se puede leer un archivo de datos con read.csv ('foo.csv') (si está bajo el directorio actual de trabajo) o read.csv ('../ datos / foo.csv ') (subir un nivel y encontrar el archivo con los datos / directorio); cuando la difusión de los resultados, podemos hacer un archivo de todo el directorio (por ejemplo, como un paquete postal).

::::No cambie el directorio de trabajo después de la computación ha comenzado: setwd () es la función en R para establecer el directorio de trabajo, y no es raro ver a setwd (C:/path/to/some/dir') en el usuario de código, que es malo porque no sólo es una ruta absoluta, sino que también tiene un efecto global sobre el resto del documento de origen. En ese caso, hay que tener en cuenta que todas las rutas relativas pueden necesitar ajustes desde el directorio raíz ha cambiado, y el software pueden escribir el resultado en un lugar inesperado (por ejemplo, se espera que las cifras que se genere en los ./figures / directorio, pero son realmente escritos en ./data/figures/ lugar si setwd ( './ datos /')). Si tenemos que establecer el directorio de trabajo en absoluto, hacerlo en el comienzo de una sesión de R; la mayor parte de los editores para ser introducido en el capítulo 4 sigue esta regla, y el directorio de trabajo se establece en el directorio del documento de origen antes de knitr se llama compilar documentos. Si es inevitable, o hace que sea mucho más conveniente para usted para escribir código después de establecer un directorio de trabajo diferente, se debería restaurar el directorio más adelante; por ejemplo.,

f <- **function**(...) {

*# stores current dir to a variable owd*

owd <- **setwd**("a/different/dir/")

*# restore working dir when the function exits*

**on.exit**(**setwd**(owd), add = TRUE)

*# now you can work under a/different/dir*

...

}

:::: Compilar los documentos en una sesión de R limpia: objetos R existentes en la sesión actual de R pueden "contaminar" los resultados en la salida. Está bien si escribimos un informe mediante la acumulación de fragmentos de código uno por uno y se ejecuta de forma interactiva para comprobar los resultados, pero al final nos debería compilar un informe en el modo por lotes con una nueva sesión de R por lo que todos los resultados son recién generados a partir de el código.

:::: Evitar los comandos que requieren la interacción humana: la intervención humana puede ser muy impredecible; por ejemplo, no se sabe con certeza en qué archivo que el usuario elegirá si nos aparecerá un cuadro de diálogo que le pedirá al usuario que elija un archivo de datos. En lugar de utilizar funciones como file.choose () para introducir un archivo a read.table (), debemos escribir el nombre del archivo de forma explícita; por ejemplo.,

read.table ('a-específica-archivo.txt').

:::: Evitar las variables de entorno para el análisis de datos: mientras que las variables de entorno a menudo se utilizan en gran medida en la programación para fines de configuración, es poco aconsejable utilizarlos en el análisis de datos, ya que requieren instrucciones adicionales para los usuarios configurar, y los seres humanos puede simplemente se olvidan de hacer esto . Si hay ninguna opción para configurar, hacerlo dentro del documento fuente.

:::: Adjuntar información de sesión () (instrumentos dev r :: información de la sesión ()) y las instrucciones de cómo compilar este documento: la información de la sesión hace un lector consciente del entorno de software, tales como la versión de R, el sistema operativo, y añadir -on paquetes utilizados. A veces no es tan simple como llamar a una sola función para compilar un documento, y tenemos que dejar claro cómo compilar si se requieren pasos adicionales; pero es mejor para proporcionar las instrucciones en la forma de una secuencia de comandos del ordenador; por ejemplo, un script de shell, un Makefile, o un archivo por lotes.

:::: Estas prácticas no están necesariamente restringidos a la lengua R, aunque utilizamos R para los ejemplos. Las mismas reglas se aplican también a otros entornos de computación.

Tenga en cuenta que las herramientas de programación literaria a menudo requieren que los usuarios recopilar los documentos en el modo por lotes, y es bueno para la investigación reproducible, pero el modo por lotes puede ser engorroso para el análisis exploratorio de datos. Cuando no hemos decidido qué poner en el documento final, es posible que tengamos que interactuar con los datos y el código de frecuencia, y no vale la pena la compilación de todo el documento cada vez que se actualice el código. Este problema puede ser resuelto por un editor capaz como RStudio y Emacs / ESS, que se introducen en el Capítulo 4. En estos editores, podemos interactuar con el código y explorar los datos libremente (por ejemplo, enviar o escribir código R en un asociado R sesión), y una vez que terminemos el trabajo de codificación, podemos compilar todo el documento en el modo por lotes para asegurarse de que todo el código funciona en una sesión de R limpia.

**Barriers**

A pesar de todas las ventajas de RR, existen algunas barreras prácticas, y aquí está una lista no exhaustiva:

::::los datos pueden ser enormes: por ejemplo, es común que la física de alta energía y datos de secuenciación de próxima generación en la biología pueden producir decenas de terabytes de datos, y no es trivial para archivar los datos con los informes y distribuirlos

::: confidencialidad de los datos: se puede estar prohibido para liberar los datos en bruto con el informe, sobre todo cuando se está involucrado con sujetos humanos debido a los problemas de confidencialidad

:::versión del software y de configuración: un informe puede ser generado con una versión antigua de un paquete de software que ya no está disponible, o con un paquete de software que recopila de manera diferente en diferentes sistemas operativos

::: la competencia: uno puede optar por no liberar el código o datos con el informe debido al hecho de que los competidores potenciales pueden conseguir fácilmente todo de forma gratuita, mientras que los autores originales han invertido una gran cantidad de dinero y esfuerzo

Desde luego, no debemos esperar que todos los informes en el mundo para estar a disposición del público y estrictamente reproducibles, pero es mejor compartir código o problemáticos conjuntos de datos, incluso mediocres o defectuosos que no compartir nada en absoluto. En vez de persuadir a la gente a RR por las políticas, es posible que tratamos de crear herramientas que hacen más fácil que RR cortar y pegar, y knitr es tal intento. El éxito de RPubs (http://rpubs.com) es evidencia de que una herramienta fácil puede promover rápidamente RR, ya que los usuarios disfrutar de su uso. Los lectores pueden encontrar cientos de informes aportados por los usuarios en el sitio web RPubs.

Es bastante común ver las tareas y ejercicios estudiante allí, y una vez que los estudiantes son entrenados de esta manera, podemos esperar que la investigación científica más reproducible en el futuro.

A First Look

El paquete knitr es un motor de programación literaria de propósito general que soporta los formatos de documentos incluido el látex, HTML y de rebajas (véase el capítulo 5), y los lenguajes de programación tales como R, Python, awk, C ++, y scripts de shell (Capítulo 11). Antes de empezar, necesitamos instalar knitr en R. A continuación, vamos a introducir los conceptos básicos con ejemplos mínimos. Por último, vamos a mostrar cómo generar informes de forma rápida a partir de secuencias de comandos R puros, lo cual puede ser útil para los principiantes que no saben nada de documentos dinámicos.

Desde knitr es un paquete de R, que puede ser instalado de CRAN en el usual

forma en R:

Tenga en cuenta aquí que las dependencias = VERDADERO es opcional, y se instalará todos los paquetes que no sean absolutamente necesarios, pero puede mejorar este paquete con algunas características útiles. La versión de desarrollo se encuentra alojado en Github: https://github.com/yihui/knitr, y siempre se puede revisar la versión más reciente desarrollo, que puede no ser estable, pero contiene las últimas correcciones de errores y nuevas características. Si tiene algún problema con knitr, la primera cosa a comprobar es su versión:

Una vez que hemos instalado knitr, podemos compilar documentos de origen mediante la función knit(), por ejemplo,

**library**(knitr)

**knit**("your-file.Rnw")

Un archivo \* .Rnw es por lo general un documento de látex con R código incrustado en él, como veremos en el siguiente apartado y en el capítulo 5, en el que se introducirán más tipos de documentos.