单了。跳转到项目所在目录并双击保存的镜像文件即可。这样做可以启动R，载入保存的工作空间，并设置当前工作目录到这个文件夹中。

1.3.4　输入和输出

启动R后将默认开始一个交互式的会话，从键盘接受输入并从屏幕进行输出。不过你也可以处理写在一个脚本文件（一个包含了R语句的文件）中的命令集并直接将结果输出到多类目标中。

1. 输入

函数source("*filename*")可在当前会话中执行一个脚本。如果文件名中不包含路径，R将假设此脚本在当前工作目录中。举例来说，source("myscript.R")将执行包含在文件myscript.R中的R语句集合。依照惯例，脚本文件以.R作为扩展名，不过这并不是必需的。

2. 文本输出

函数sink("*filename*")将输出重定向到文件*filename*中。默认情况下，如果文件已经存在，则它的内容将被覆盖。使用参数append=TRUE可以将文本追加到文件后，而不是覆盖它。参数split=TRUE可将输出同时发送到屏幕和输出文件中。不加参数调用命令sink()将仅向屏幕返回输出结果。

3. 图形输出

虽然sink()可以重定向文本输出，但它对图形输出没有影响。要重定向图形输出，使用表1-4中列出的函数即可。最后使用dev.off()将输出返回到终端。

表1-4　用于保存图形输出的函数

函　数	输　出
pdf("filename.pdf")	PDF文件
win.metafile("filename.wmf")	Windows图元文件
png("filename.png")	PBG文件
jpeg("filename.jpg")	JPEG文件
bmp("filename.bmp")	BMP文件
postscript("filename.ps")	PostScript文件

让我们通过一个示例来了解整个流程。假设我们有包含R代码的三个脚本文件script1.R、script2.R和script3.R。执行语句：

```
source("script1.R")
```

将会在当前会话中执行script1.R中的R代码，结果将出现在屏幕上。

如果执行语句：

```
sink("myoutput", append=TRUE, split=TRUE)
pdf("mygraphs.pdf")
source("script2.R")
```

文件script2.R中的R代码将执行，结果也将显示在屏幕上。除此之外，文本输出将被追加到文件

myoutput中，图形输出将保存到文件mygraphs.pdf中。

最后，如果我们执行语句：

```
sink()
dev.off()
source("script3.R")
```

文件script3.R中的R代码将执行，结果将显示在屏幕上。这一次，没有文本或图形输出保存到文件中。整个流程大致如图1-6所示。

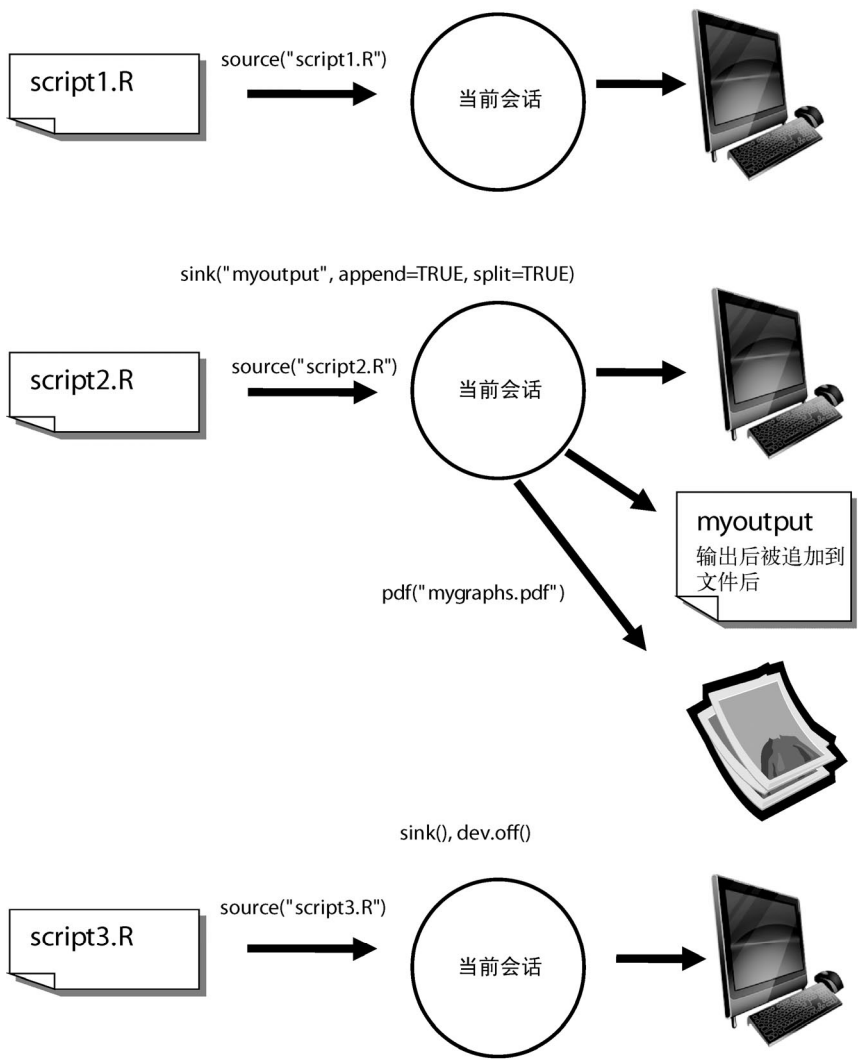

图1-6 使用函数source()进行输入并使用函数sink()进行输出

R对输入来源和输出走向的处理相当灵活，可控性很强。在1.5节中，我们将学习如何在批处