# Rocker 简介：R 的 Docker 容器

*作者：Carl Boettiger、Dirk Eddelbuettel*

**抽象的**我们描述了 Rocker 项目，该项目提供了一套广泛使用的 Docker 镜像，其中包含针对特定任务的自定义 R 环境。我们讨论了该套件的组织方式，以及这些工具如何提高 R 用户和开发人员的可移植性、可扩展性、可重复性和便利性。



## 介绍

Rocker 项目于 2014 年 10 月启动，作为 CB 和 DE 之间的合作，提供包含 R 环境的高质量 Docker 镜像（博蒂格和埃德尔布特尔,2014年）。从那时起，该项目在社区中得到了广泛的关注，并得到了实质性的发展和演变。在这里，我们寻求记录该项目的目标和用途。

### 什么是 Docker?

Docker 是一种流行的开源工具，用于创建、分发、部署和运行软件应用程序*容器*。容器提供了一个虚拟环境（参见克拉克等人（2014年）了解常见虚拟环境的概述）需要应用程序运行所需的所有操作系统组件：代码、运行时、系统工具、系统运行时。 Docker 容器是轻量级的，因为它们共享操作系统内核，使用分层文件系统立即启动，最大限度地减少磁盘占用空间和下载时间，基于在所有主要平台（Linux、Mac、Windows）上运行的开放标准构建，并提供附加层通过在隔离环境中运行应用程序来保证安全性（Docker, 2015年）。熟悉一些关键术语有助于理解本文。术语“容器”是指计算机上的隔离软件环境。 R 用户可以将运行容器视为类似于加载 R 包；它提供了一组软件功能。 Docker“镜像”是该软件的二进制存档，类似于 R 二进制包：给定版本仅下载一次，然后可以在需要时“运行”以创建容器。 “Dockerfile”是创建 Docker 镜像的配方、源代码。 Rocker 项目的开发和贡献重点在于这些 Dockerfile 的构建、组织和维护。

## 设计原则和用例

Docker 使用户可以非常方便地访问预配置和预构建的内容*二进制*“正常工作”的图像。这使 R 用户可以访问比 R 项目本身或其发行版（通常只关注一个（当前）版本）提供的更广泛的现成环境。例如，Windows 上的 R 用户只需启动一个命令（一旦安装了 Docker 本身）即可在本地运行 RStudio Server 或 Shiny Server。另一个常见用例是在不影响本地系统的情况下访问 R-devel。在这里，我们详细介绍了这些容器化 R 环境版本的一些主要用例，以及帮助它们发挥作用的设计原则。

### 便携性：从笔记本电脑到云

Rocker 容器的一个常见用例是提供一种快速可靠的机制，将自定义 R 环境部署到远程服务器，例如 Amazon Web Services Elastic Compute (AWS EC2)、DigitalOcean、NSF 的 Jetstream 服务器（斯图尔特等人,2015年) 或私人或机构服务器硬件。 Rocker 容器也很容易在大多数现代笔记本电脑上本地运行，只需首先为 Windows、MacOS 和基于 Linux 的操作系统安装适当的 Docker 命令 (或发行版)。 通过与本地主机共享卷，用户仍然可以使用熟悉的本机工具操作文件，同时通过可重现的容器化环境执行计算 (博蒂格，2015年）。能够在本地计算机上的可预测、预配置的 R 环境中测试代码，然后在远程服务器上的相同环境中运行相同的代码（*例如*，为了访问更大的 RAM、更多的处理器，或者只是为了将本地计算机从长时间运行的计算中解放出来）对于低摩擦扩展分析至关重要。如果没有这种容器化，让代码在远程环境中正确运行可能是一项艰巨的任务，需要时间和许多潜在用户可能不具备的知识。

例如，以下 bash 命令在几乎所有基于 Linux 的服务器上安装 Docker，然后启动 Rocker 容器，通过 Web 界面提供 RStudio 服务器环境。

```
wget -qO- https://get.docker.com/ |嘘
sudo docker run -p 8787:8787 -e PASSWORD=<PICK-A-PASSWORD> rocker/rstudio
```

这docker 运行选项 -p设置 RStudio 的端口，8787 是 RStudio 的默认端口（将你的用户添加到 docker 组可以避免需要须藤呼叫命令docker：sudo usermod -g docker $USER）。许多学术和商业云提供商使得在启动容器时执行此类代码片段成为可能，而无需远程控制进入机器。用户只需粘贴服务器的 IP 地址或 DNS 名称（后跟所选端口，*例如*, :8787)进入浏览器并输入适当的密码。这为用户提供了一个在远程计算机上运行的熟悉的交互式环境，同时需要最少的专业知识。

以前，在集中管理的多用户计算机（例如部门或大学集群）上部署 Docker 容器比较困难，因为系统管理员不希望允许 Docker 运行时环境所需的提升的用户权限。为了解决这个问题，劳伦斯伯克利国家实验室（LBNL）提出了“奇点”（劳伦斯伯克利国家实验室,2017年）：一个容器运行时环境，用户可以安装并使用它来运行大多数 Docker 容器，而无需 root 权限，从而可以更轻松地在此环境中部署 Rocker 容器。

这种可移植性在教学环境中也很有价值。要求学生在个人笔记本电脑上安装所有必要的软件对于短期研讨会来说尤其具有挑战性，因为下载和安装时间以及跨异构计算机的故障排除对于学生和教师来说都很耗时且令人沮丧。通过部署 Rocker 映像或 Rockerderived 映像（参见*可扩展性*)，讲师只需使用笔记本电脑上的浏览器，即可轻松为所有学生提供对预配置软件环境的访问权限。根据我们在研讨会和学期课程中的经验，这种策略已被证明是有效的。类似的基于 Docker 的云部署已扩展到数百名学生的课程，*例如*，在杜克大学（切廷卡亚-龙德尔和龙德尔,2017年）和加州大学伯克利分校（加州大学伯克利分校,2017年）。

## 接口

Rocker 项目设计的一个重要方面是用户能够通过交互式 shell 会话（例如 R shell 或 bash shell）或通过 Web 浏览器访问 RStudio 与容器上的软件进行交互®服务器集成开发环境（IDE）。 R 用户的传统远程和高性能计算工作流程通常需要使用远程控制以及仅限终端的界面，对交互式图形提出了挑战，并对不熟悉这些工具和环境的用户造成了障碍。访问 RStudio®容器通过浏览器消除了这些障碍。 Rocker 镜像包含预先安装并经过 RStudio 明确许可配置的 RStudio 服务器软件®公司

用户可以访问root巴什shell 在运行的 Rocker 容器中使用

```
docker exec -ti <容器id> bash
```

这对于安装系统依赖项等管理任务非常有用。所有 Rocker 镜像都可以作为交互式 R、RScript 或 bash shell 运行，而无需运行 RStudio，这对于批处理作业或任何喜欢该环境的人都很有用。

与任何交互式 Docker 容器一样，用户应该指定交互式（-我）和终端 (-t,这里结合互动作为 -顺流ags，并指定所需的可执行环境（*例如*,R，尽管其他常见选项可能是脚本或者重击）：

```
docker run --rm -ti rocker/tidyverse R
```

此示例显示了 --rm FLag 表示在交互式会话结束时应删除容器。有关共享卷、管理用户权限等详细信息，请访问 Rocker 网站。https://rocker-project.org。

## 沙盒化

Rocker 容器的另一个功能是能够提供沙盒环境，与软件以及可能与机器上的其他数据隔离。许多用户不愿意升级

一套已安装的软件包，如果安装不顺利，可能会破坏现有代码甚至 R 环境。然而，为了对同事进行分析或访问更新的方法，通常需要升级软件包和/或 R 环境。 Rocker 提供了一个简单的解决方案。例如，用户可以在 Rocker 容器内运行需要最新版本的 R 和相关软件包的 R 代码，而无需先升级其本地安装。相反，人们可以使用 Rocker 在带有早期版本的 R 软件包的较旧 R 版本上运行代码，同样无需对本地 R 安装进行任何更改。另一个常见的用例是访问支持特定选项的容器，例如使用 gcc 或 clang 编译器清理程序（埃德尔比特尔,2014年）这些要求 R 本身具有专门的设置，这些设置可能对许多 R 用户在其本机系统上不可用或不熟悉，但可以通过提取 Rocker 镜像轻松部署 摇杆/r-devel-san或者摇杆/r-devel-ubsan-clang。

此沙箱功能在远程计算环境中也很有价值，它允许系统管理员授予用户在容器内安装需要 root 权限的软件的自由，同时不授予他们在主机上的 root 访问权限。启动 Docker 容器需要 root 访问权限，但不需要访问已经运行并提供 RStudio 等某些服务的容器。用户通过 RStudio 登录容器。默认情况下，interface 没有 root 权限，但可以安装 R 包。在容器中授予这些用户 root 权限仍然会使他们处于主机容器的沙箱中。此外，可以通过备用容器运行时环境 Singularity 从没有 root 权限的计算机部署 Docker 映像（劳伦斯伯克利国家实验室,2017年)。与传统虚拟机不同，这些容器不会占用大量的预留资源，因为典型的主机可以轻松支持数百个容器（Docker,2015年）。

## 透明的

用户可以通过检查相关的 Dockerfile 配方轻松确定任何 Rocker 映像上安装的软件堆栈，该配方提供了简洁、人类可读的安装记录。所有 Rocker 镜像都通过 Docker Hub 进行自动构建，Docker Hub 也充当分发镜像的中央默认存储库。使用自动构建而不是将预构建的映像二进制文件上传到 Docker Hub 可以避免构建与配方不匹配的可能性。相应的 Dockerfile 在 Docker Hub 和 Rocker 项目的链接 GitHub 存储库中都可见（https://github.com/rocker-org)，它提供了对这些配方所做的所有更改的透明版本历史记录，以及文档、社区 Wiki 和问题跟踪器，用于讨论对 Dockerfile 提出的更改、错误、改进，并解决用户可能遇到的任何问题。让受信任的提供商 (Docker Hub) 自动构建这些公共源文件，而不是在本地构建并仅以二进制形式上传，从安全角度来看，这也有助于避免恶意软件。

## 社区优化

拥有由公开托管的可复制配方创建的共享、透明的计算环境有助于社区输入配置细节。R 及其许多软件包和相关软件可以配置各种选项、编译器、不同的线性代数库等。虽然这种灵活性反映了不同的需求，但许多用户依赖于默认设置，这些设置可能更多地是为了简化安装而不是为了性能而优化的。Rocker 配方反映了社区对这些选择的大量意见，以及 20 年来 R Debian 维护者在使用哪些配置选项方面的丰富经验和专业知识。这有助于创建更精细、优化的 R 环境参考实现，以及一个用于比较和讨论其他地方经常忽略的这些问题的平台。GitHub 上 Rocker 存储库中的问题和拉取请求证明了这些讨论和改进。广泛使用 Rocker 映像有助于促进对这些选择的测试以及来自 R 社区许多成员的进一步调整配置的贡献。

## 版本控制

对于那些比拥有任何软件的最新版本更需要计算再现性的用户来说，访问特定版本的软件可能很重要，因为后续版本可能会改变代码的行为、引入错误或以其他方式改变以前的结果。版本化堆栈 （r-ver、rstudio、tidyverse、verse、和地理空间提供旨在每次构建相同软件堆栈的映像，无论新库和软件包的发布如何。用户应在 Docker 映像名称中指定 R 版本标签以请求版本稳定的映像，*例如*， 摇杆/诗歌：3.4.0。如果没有明确请求标签，Docker 将提供带有标签的镜像： 最新的，它将始终拥有该软件的最新可用版本（每晚构建）。

在版本标记镜像上构建的用户将默认使用 MRAN 快照镜像 (革命分析,2017年) 与该镜像最新的日期相关联。这确保 Dockerfile 构建来自摇滚乐/诗歌：3.4.1将仅安装 2017 年 6 月 30 日（即 R 3.4.1 发布日）在 CRAN 上提供的 R 软件包版本。当然可以用标准 R 方式覆盖这个默认值，*例如*，通过在任何命令中显式指定不同的 CRAN 镜像来安装.packages(),或者通过调整默认的 CRAN 镜像 选项(repo=<CRAN-MIRROR>)在 .R 配置文件。请注意，与当前版本关联的 MRAN 日期 (*例如*,3.4.2在撰写本文时）将继续推进 Docker-hub 映像，直到下一个 R 版本。软件安装自apt-get这些图像将来自稳定的 Debian 版本（拉紧或者杰西）因此不会更改版本（尽管它会收到安全补丁）。使用生物碱（）实用程序还将安装与系统上找到的 R 版本相适应的版本（Bioconductor 半年发布模型避免了对 MRAN 镜像的需求）。从 GitHub 或其他来源安装软件包的用户可以请求特定的 git 发布标签或哈希以获得更可重复的构建，或者采用其他方法，例如包鼠（Ushey 等人，2016）。关于 Docker 在计算可重复性方面的用途和局限性的更一般性讨论，可以参见博蒂格（2015年）。

### 可扩展

任何便携式计算环境都不可避免地面临着一个极端的“厨房水槽问题”和另一个极端的“发现问题”之间的紧张关系。厨房水槽图像试图在单个图像中容纳太多用例。此类镜像不可避免地会非常大，因此部署、维护和优化会很慢或很困难。在另一个极端，提供太多的专业图像会使用户更难找到他们需要的图像。 Rocker 项目旨在通过提供一套精心策划的图像来避免这两个问题，这些图像可以由个人和社区轻松扩展。

为了使扩展透明且持久，任何用户都可以通过基于适当的 Rocker 映像编写自己的 Dockerfile 来扩展 Rocker 映像。Rocker 堆栈中的 Dockerfile 本身应该提供一个简单的示例。用户首先选择适合其需求的基础映像：如果 RStudio®需要界面，用户可能会从 来自摇杆/rstudio；用于测试 R 包中特定 C 代码的图像可能会使用来自rocker/r-devel-san，用于重现数据分析的图像除了选择合适的基础库外，还可能会选择稳定的版本标签，*例如*,:来自rocker/tidyverse：3.4.1用户可以使用标准 R 和 Debian 机制轻松地将其他软件添加到任何正在运行的 Rocker 映像中。有关如何扩展 Rocker 图像的详细信息，请访问https://rocker-project.org。

共享这些 Dockerfile 还可以促进针对特定社区的扩展的出现。例如，摇杆/地理空间该镜像源自众多 Rocker 用户的意见，他们都在现有 Rocker 镜像的基础上添加了常见的地理空间库和软件包。这种整合有助于创建更精细的镜像，广泛支持各种常用数据格式和库。其他社区镜像是独立于 Rocker 项目开发和维护的，例如波普根国家进化综合中心（NESCent）开发的面向群体遗传学的软件的图像。 Rocker 镜像还被 NSF 赞助的 Whole Tale 项目用作基础 Docker 镜像，以实现可重复计算（Ludaescher 等人,2017年），并被 R-Hub 项目大量用于自动化包测试（恰尔迪,2017年）。

## 摇杆组织和工作流程

Rocker 项目由一套由 Docker Hub 自动构建并托管在 Docker Hub 上的镜像组成， https://hub.docker.com/r/rocker. 源 Dockerfile、支持脚本和文档托管在 GitHub 上的组织下rocker-org，https://github.com/rocker-org问题跟踪器和拉取请求用于社区对这些镜像的输入、讨论和贡献。Rocker 项目 wiki 提供了一个地方来综合社区贡献的文档、用例以及有关使用 Rocker 镜像的其他知识。

### Rocker 项目中的图像

Rocker 项目旨在提供一个小型的 Docker 镜像核心，作为方便的“基础”镜像，其他用户可以通过编写自己的 Dockerfile 来构建自定义 R 环境，同时还提供可使用的“包含电池”的镜像方法盒子外面。平衡由非常不同的用例驱动的多样化需求与创建仍然足够轻量、易于使用和易于维护的图像的总体目标之间的挑战是一个困难

艺术。单独的 Rocker 图像和图像堆栈中的实现永远无法完美地满足每个人的平衡，但今天反映了过去几年中大量的社区投入和测试。

所有 Rocker 镜像都基于 Debian Linux 发行版。Debian 平台提供了一个小型基础镜像，即众所周知的易于包管理系统和丰富的软件库生态系统，使其成为 Docker 镜像的首选基础镜像，其中包括许多由 Docker 自己的开发团队维护的“官方”镜像。 Debian 平台也许也是 R 社区中支持最好的 Linux 平台，包括一个活跃的r-sig-debian列表服务。 Debian 稳定版本之间的相对较长的周期（最近大约两年）意味着 Debian 稳定版本中的软件（*例如*,debian:jessie, debian:stretch)版本可能会明显落后于流行软件（包括 R）的当前版本。更新版本的软件包可以在预发行版中找到，debian:测试，最新的二进制版本可以在 debian：不稳定。Rocker 项目大致可以分为两个堆栈，它们满足不同的需求，这反映在它们基于哪个 Debian 发行版上。第一个堆栈基于Debian：测试。第二个是最近推出的，仅基于 Debian 稳定版本。Rocker 镜像始终指向特定的稳定版本（杰西（伸展）并且不要使用标签Debian：稳定，这是一个滚动标签，始终指向最新的稳定版本。不同的摇杆堆栈具有不同的目标，因此提供不同的图像，如下表 1 和表 2 所示。

## 这debian:测试-基于图像

这debian:测试堆栈旨在最有效地利用上游构建：预编译的
。德布Debian 存储库提供的二进制文件。从二进制文件安装软件既更快又更容易，因为包管理器（易于）管理必要的（二进制）依赖项并绕过从源代码编译的耗时过程。该堆栈基于Debian：测试 这意味着可以使用比 Debian 稳定版本中更新的二进制文件版本。为了提供对最新可用二进制文件的访问，此堆栈使用 apt-pinning (Debian 项目,2017年）以允许易于包管理器还可以安装来自debian：不稳定，它代表了在必要时为 Debian 构建的最新、最前沿的软件包。例如，r-baseRocker 提供的映像将最新版本的 R 作为来自 Debian 的二进制文件安装不稳定。类似地，许多流行的 R 包的最新版本也可以通过包管理器安装，*例如*, apt-get 安装 r-cran-xml。这对于具有外部系统依赖项的包（例如libxml2-dev在这个例子中），它们不能从 R 控制台安装，因为它们是系统依赖项，而不是从 R 内部安装的 R 包。但我们应该注意，在超过 11,000 个 CRAN 包中，只有大约 500 个可用作 Debian 包。

正如名字测试和不稳定这意味着，这种方法并非没有挑战。任何给定包的特定版本都可能随着包从不稳定进入测试。新版本发送至不稳定在 Debian 开发的正常过程中。这有时会破坏 Dockerfile 中以前有效的安装命令，直到维护者重定向包管理器以从不稳定以前可以安装的源测试，反之亦然（使用 -t选择易于）。也就是说，包仅从不稳定到测试经过几天的时间，并且如果特定版本的迁移和安装与其依赖关系图中的其他软件包没有交互。这样， 不稳定作为验证实验室测试相当稳定但最新。

关系到稳定的，这测试因此，堆栈具有以下优点：

1.这些Dockerfile易于开发和扩展，因为几乎所有软件都可以通过包管理器安装。

2.这些 Dockerfile 始终安装最新的可用软件。
3.这些Dockerfile几乎总是可以相对快速地构建。

还有这些缺点：

1. 这些 Dockerfile 需要偶尔进行少量维护以确保成功构建。 （*例如*，更改安装指令不稳定到测试或相反亦然）。
2. 生成的镜像本质上是动态的：几个月或几年后重建相同的 Dockerfile 会生成安装有明显不同软件版本的镜像。
3. apt pinning的使用对于某些用户来说可能比较陌生，用户必须最终负责确保兼容性（Debian 项目, 2017年）。

## 图像概览

这debian:测试-基于堆栈当前包括由 Rocker 开发团队积极维护的七个图像（表 1）。r-base建立在debian:测试，堆栈中的其他六个都直接从r-碱基。这r-base镜像的独特之处在于它被 Docker 组织本身指定为 R 语言的官方镜像。该官方镜像由 Docker Inc 的员工根据 Rocker 团队维护的 Dockerfile 进行审查和构建。因此，用户应该在没有组织名称空间的 Docker 命令中引用此映像，*例如*,docker 运行

- t r 碱基访问官方图片。 Rocker 项目中的所有其他镜像均未由 Docker Inc 单独审核和构建，必须使用摇杆命名空间，*例如*,docker 运行-ti rocker/r-devel。

此堆栈中的一些图像面向 R 开发社区：r-devel、drd、r-devel-san 和 r-devel-ubsan-clang都在 R 的当前版本旁边添加了一个开发版本的副本，由r-碱基。在这些图像上，开发版本别名为研发为了与当前版本区分开来，R。顾名思义，每个镜像都提供略有不同的配置。特别令人感兴趣的是提供开发 R 的镜像，这些镜像支持 C/C++ 地址和未定义行为清理程序，配置起来有些困难（埃德尔比特尔,2014年）。

由于这些镜像主要面向开发人员和/或作为自定义用途的基础镜像，因此此堆栈不包含许多特定的 R 包。其他依赖项和包可以轻松从易于。R 软件包不可用易于可以使用以下任一方式从 CRAN 直接安装存储库R或者较小的脚本，如https://rocker-project.org/use。

该堆栈还包括图像闪亮的和rstudio：测试提供 Shiny 服务器和 RStudio®RStudio 的服务器 IDE®公司，建立在r-base图片。RStudio®和 Shiny 是 RStudio Inc 的注册商标，其使用以及在 Docker Hub 上以二进制形式分发其软件已获得 RStudio 的明确许可，并已授予 Rocker 项目。用户应查看 RStudio 的商标使用政策(http://www.rstudio.com/about/trademark/）并解决有关进一步分发或其他问题的询问权限@rstudio.com。 Rocker 项目还通过 RStudio 提供图像®服务器和稳定版本堆栈中的 Shiny 服务器。

**构建时间表**： 官方r-base任何官方更新后，Docker 都会重建镜像德比安图像（大约每隔几周）。堆栈的其余部分使用构建触发器，无论何时都可以重建图像r-base已更新，或者 Dockerfile 源已在相应的 GitHub 存储库中更新。此堆栈中唯一的例外是数据冗余图像，每周由计划任务扳机。

**表格1：**这debian:测试图像堆栈

| 图像 | 描述 | 尺寸 | 下载 |
|---|---|---|---|
| r-base | 当前版本 R R-devel 的官方图像添加了并排 r-base | 254MB | 632,000 |
| r-devel | （使用别名研發) 轻量级 r-devel，每周构建 | 1 GB | 4,000 |
| 博士 | | 571MB | 4,000 |
| r-开发-桑 | 与 r-devel 一样，但使用编译器清理程序 | 1.1 GB | 1,000 |
| r-devel-ubsan-clang | Sanitizers、clang c 编译器（而不是 gcc） | 1.1 GB | 525 |
| rstudio：测试 | rstudio 在 debian:testing 上构建 | 1.1 GB | 1,000 |
| 闪亮的 | r-base 上的闪亮服务器 | 409MB | 123,000 |

**表 2：**这摇杆版本图像堆栈

| 图像 | 描述 | 尺寸 | 下载 |
|---|---|---|---|
| r-版本 | 版本稳定的基础 R 和 src 构建工具 添加 | 219 兆 | 6,000 |
| 工作室 | rstudio | 334 兆 | 314,000 |
| 整洁宇宙 | 添加 tidyverse 和开发工具 | 656MB | 83,000[1] |
| 诗 | 添加 java、tex 和出版相关的包 添加地理空间库 | 947 兆 | 9,000 |
| 地理空间 | | 1.3GB | 4,000 |

[1]该数字包括旧名称下的 49,000 次下载哈德利诗。

### 这debian:稳定-基础堆栈

该堆栈强调 Docker 构建的稳定性和可重复性。该堆栈是最近（2016 年 11 月）为了响应大量用户输入和请求而引入的。该堆栈的主要功能是能够运行旧版本的 R 以及当时版本的 R 软件包。用户使用图像标签指定所需的版本，*例如*,摇杆/R-版本：3.3.1 将引用安装了 R 3.3.1 版本的图像。省略标签相当于使用标签 最新的，顾名思义，它将始终指向使用当前 R 版本的镜像。因此，如果用户想要创建基于当前版本的下游 Dockerfile（但在新 R 版本发布后将继续重建相同的环境），则应明确包含相应的版本标签，*例如*,摇杆/r-ver：3.4.2在撰写本文时，而不是最新的标签。用户还可以使用标签运行当前的R开发版本 开发，这是每晚从 R-devel 源构建的颠覆。

**MRAN 档案**：为了便于在这些映像上安装 R 软件包的同期版本，安装 R 软件包的默认 CRAN 镜像固定为与该版本 R 最新版本的最后日期相对应的 CRAN 快照。这些快照由 Revolution Analytics（现为 Microsoft 的一部分）创建的 MRAN 档案提供。它存档了所有 CRAN 的每日快照，用户可以从中安装软件包，使用通常 安装.packages()功能 （革命分析,2017年）。用户始终可以通过显式传递任何当前的 CRAN 存储库来覆盖此默认值。与 CRAN 不同，Bioconductor 仅通过与 R 春季发布时间表一致的半年发布更新其存储库。因此，Bioconductor 包可以使用通常的方式安装生物岩，它会自动选择与正在使用的 R 版本相对应的 Bioconductor 版本。

**版本标签**：版本标签在整个堆栈中传播：*例如*,摇杆/tidyverse：开发 将在整洁宇宙（威克姆,2017年）安装在 R-devel 的夜间版本上。我们还鼓励在此堆栈上构建软件包的开发人员相应地标记他们的图像。表 3 指示堆栈中当前可用的 R 版本，返回到3.1.0。虽然较旧的版本可能会在以后添加到堆栈中，但我们注意到 MRAN 快照始于 2014-09-17，因此只能追溯到 R3.1时代。每个标签都必须从单独的 Dockerfile 构建，从而允许在构建指令中存在细微差异以适应不断变化的依赖关系。过去版本的 Dockerfile（*例如*， 之前3.4.2目前）旨在长期保持静态，而当前版本的标签，最新的，和 开发可能会进行调整以适应新功能或依赖项。版本标签也遵循语义，因此省略标签的第二个或第三个位置与请求最新版本相同：即摇杆/诗句：3.3是相同的摇杆/诗歌：3.3.3，和摇滚/诗歌：3（在撰写本文时），摇杆/主歌：3.4.2。这是使用 Docker Hub 中的构建后钩子实现的，请参阅以下示例https://github.com/rocker-org/rocker-versioned/了解详情。

**安装**：在此堆栈中，所需的 R 版本始终直接从源代码而不是易于存储库。编译器和依赖项仍然从稳定版安装易于存储库，因此落后于在测试堆栈。版本标签3.3.3及更早版本基于 Debian 8.0 版本，代号为杰西，尽管3.4.0 - 3.4.2，开发，和最新的 基于 Debian 9.0，拉紧， （发布于 2017-06-17，而 R 位于3.4.0),从而可以访问更新版本的常见系统依赖项和编译器。安装 R 后，编译 R 所需的依赖项（运行时不需要的依赖项）就会被删除，从而保持基础映像的轻量级，从而加快下载速度。虽然常见 R 包所需的大多数系统依赖项仍然可以从易于存储库，有时必须从源代码编译较新的版本（*例如*, 吉布斯采样程序 JAGS (普卢默,2017年）和地理空间工具包 GDAL 都必须从源代码编译Debian:杰西图像）。在此堆栈中，用户应避免使用以下方式安装 R 包：易于无需仔细考虑，因为这将从 Debian 存储库安装第二个（可能不同的）R 版本，以及自任何版本以来 R 软件包的过时版本r-cran-pkgnameDebian 存储库中的软件包将依赖于r-base在易于 以及。

**构建时间表**：所有镜像都是从其相应的 Dockerfile 自动构建的（可以在 GitHub 存储库中找到）rocker-org/rocker-versioned和地理空间 (geospatial)。A计划任务作业将夜间构建触发器发送到 Docker Hub 以重建最新的和开发整个堆栈中的标记图像。为了减少集线器上的负载，每月发送数字版本标签的构建触发器。尽管旧版 R 版本的 Dockerfile 每次都会安装几乎相同的软件环境，但每月在 Docker Hub 上重建这些映像可确保它们继续从上游接收 Debian 安全更新，并证明构建配方仍然可以成功执行。请注意，使用外部存储库中的软件重建图像永远不会产生按位相同的图像，因此图像标识符哈希值将在每次构建时发生变化。

### 图像概览

在此堆栈中，每个图像都构建在前一个图像的基础上，而不是所有其他图像都直接构建在基础图像上，如测试堆栈。表 2 列出了此堆栈中五个镜像的名称和描述，以及镜像大小和 Docker Hub 的近似下载次数。大小反映（压缩）累积大小：已经下载最新版本的用户r-版本然后拉出一个副本工作室图像只需要下载额外的115 MB工作室层，而不是列出的全部 334 MB。这种线性设计限制了灵活性（没有选项整洁宇宙没有rstudio）但简化了使用和维护。虽然没有一个环境对每个人来说都是最佳的，但在此堆栈中选择的软件包和堆栈排序都反映了相当多的社区输入和调整。

这工作室图像包括一个轻量级、易于使用且对docker友好的在里面系统，s6（贝尔科特， 2017年）用于运行持久服务，包括 RStudio®服务器。该系统为下游 Dockerfile 开发人员提供了一种方便的方式来添加额外的持久服务（例如 远程控制服务器）到单个容器，或在容器启动或关闭时应运行的其他启动或关闭脚本。这工作室image 使用这样的启动脚本，在运行时通过环境变量配置登录密码和权限等用户设置。

这整洁宇宙图像包含常用的所有必需和建议的依赖项 整洁宇宙和开发者工具R 包，包括外部数据库库（*例如*、MariaDB 和 PostgreSQL）。用户应查阅 Dockerfiles 包或安装的.packages()直接列出已安装软件包的完整列表。这诗库添加了常用的依赖项，特别是庞大但不全面的 LaTeX 环境和 Java 开发库。此前，Rocker 项目提供了图像哈德利诗后来又被分为整洁宇宙和诗 通过社区的投入。

**表3：**可用的标签摇杆版本堆。

| 标签 | apt 仓库 | MRAN 日期 | 构建频率 | 带有标签的图像 |
|---|---|---|---|---|
| 开发 | 拉紧 | 当前日期 | 每晚 | r-ver、rstudio、tidyverse、verse、地理空间 |
| 最新的 | 拉紧 | 当前日期 | 每晚 | r-ver、rstudio、tidyverse、verse、地理空间 |
| 3.4.2 | 拉紧 | 当前日期 | 每月 | r-ver、rstudio、tidyverse、verse、地理空间 |
| 3.4.1 | 拉紧 | 2017-09-28 | 每月 | r-ver、rstudio、tidyverse、verse、地理空间 |
| 3.4.0 | 拉紧 | 2017-06-30 | 每月 | r-ver、rstudio、tidyverse、verse、地理空间 |
| 3.3.3 | 杰西 | 2017-04-21 | 每月 | r-ver、rstudio、tidyverse、verse、地理空间 |
| 3.3.2 | 杰西 | 2017-03-06 | 每月 | r-ver、rstudio、tidyverse、verse、地理空间 |
| 3.3.1 | 杰西 | 2016-10-31 | 每月 | r-ver、rstudio、tidyverse、verse、地理空间 |
| 3.3.0 | 杰西 | 2016-06-21 | 每月 | r-版本 |
| 3.2.0 | 杰西 | 2015-06-18 | 每月 | r-版本 |
| 3.1.0 | 杰西 | 2014-09-17 | 每月 | r-版本 |

中的几张图片摇杆版本当本地构建时，可以使用以下命令在构建上自定义堆栈（而不是从 Docker Hub 中提取预构建的映像）：构建参数选择码头工人构建。在里面r-版本图像，用户可以设置R_版本和建造日期(MRAN默认快照）。 在 工作室图片用户可以设置RSTUDIO_版本（否则默认为最新版本），并且PANDOC_TEMPLATES_VERSION 。

该堆栈还利用了由以下项定义的 Docker 元数据标签：http://schema-label.org，表示图像执照 （GPL-2.0）vcs-url (GitHub 存储库）以及小贩 （Rocker Project）。这些元数据可以在下游图像中被更改或扩展。

## 结论

在过去的几年里，Docker 在工业界和学术界得到了广泛的采用。开放容器倡议（Linux 基金会：项目,2017年）现在提供了一个开放标准

通过 Singularity 等项目，进一步将这种容器方法扩展到研究环境（劳伦斯伯克利国家实验室,2017年），允许用户在没有 root 访问权限的机器（例如集群或私有服务器）上部署 Rocker 等容器化环境。容器化有望解决研究计算中的许多挑战，例如可移植性和可复制性，这些挑战通常依赖于复杂且异构的软件堆栈（博蒂格,2015年）。然而，在容器中实现这样的环境并不是一项简单的任务，而且并非所有实现都提供相同的可用性、可移植性或可再现性。在这里，我们详细介绍了 Rocker 项目通过开放和社区驱动的流程创建和维护这些环境所采取的方法。 Rocker 项目的这种结构已经在三年多的运营过程中不断发展，同时吸引了不断扩大的学术研究人员、大学教师和行业用户。我们相信这个概述不仅对对 Rocker 项目感兴趣的用户和开发人员具有指导意义，而且可以作为围绕其他环境或领域的类似工作的模型。

## 参考书目


L・贝尔科特。 *s6: skarnet.org 的小型安全监管软件套件*，2017。 网址 https://skarnet.org/software/s6/。 [p]

C.博蒂格。介绍用于可重复研究的 Docker，并提供来自 R 环境的示例。*ACM SIGOPS 操作系统评论*, 49(1):71–79, 2015.doi: 10.1145/2723872.2723882。网址https://dl.acm.org/citation.cfm?id=2723882http://arxiv.org/abs/1410.0846。 [p]

C. Boettiger 和 D. Eddelbuettel。 https://rocker-project.org/blog/blog/2014/12/27/ 摇杆介绍：Docker for R，2014年。 网址 introducing-rocker。 [p]

M. Cetinkaya-Rundel 和 CW Rundel。在整个统计课程中教授计算的基础设施和工具。*PeerJ 预印本*，5：e3181v1，2017 年 8 月。ISSN 2167-9843。 doi：10.7287/peerj。预印本.3181v1。网址https://doi.org/10.7287/peerj.preprints.3181v1。 [p]

D. 克拉克、A. 库利奇、B. 哈姆林和 R. 洛维特。 BCE：伯克利用于研究和教育的通用科学计算环境。*第十三届 Python 科学大会 (SciPy 2014) 论文集*，第 1-8 页，2014 年。[p]

G.恰尔迪。*rhub：从“R”连接到“R-hub”*，2017 年。网址https://github.com/r-hub/rhub。R 包版本 1.0.1。[p]

Debian 项目。 apt-preferences 概述，2017 年。URLhttps://wiki.debian.org/AptPreferences。 [页]

Docker。什么是 Docker？，2015 年。URLhttps://www.docker.com/what-docker。 [p]

D. Eddelbuettel。消毒剂，2014 年。URLhttp://dirk.eddelbuettel.com/code/sanitizers.html。 [p]

劳伦斯伯克利国家实验室。Singularity，2017 年。网址http://singularity.lbl.gov/。 [页]

B. Ludaescher、K. Chard、M. Turk、V. Stodden 和 N. Gaffney。整个故事：科学与网络基础设施路径的融合，2017 年。URLhttps://wholetale.org/。 [p]

M・普卢默。 JAGS：使用吉布斯采样分析贝叶斯图形模型的程序，2017 年。URLhttp://mcmc-jags.sourceforge.net/。版本 4.3.0。 [p]

Revolution Analytics。Microsoft R 应用程序网络，2017 年。URLhttps://mran.microsoft.com。 [页]

CA Stewart、G. Turner、M. Vaughn、NI Gaffney、TM Cockerill、I. Foster、D. Hancock、N. Merchant、E. Skidmore、D. Stanzione、J. Taylor 和 S. Tuecke。Jetstream：自配置、可扩展的科学和工程云环境。*2015 年 XSEDE 增强型网络基础设施促进科学进步会议记录 - XSEDE '15*，第 1–8 页，纽约，美国纽约，2015 年。ACM 出版社。 ISBN 9781450337205。doi：10.1145/2792745.2792774。网址 http://dl.acm.org/citation.cfm?doid=2792745.2792774。 [p]

Linux 基金会： 项目。 开放容器倡议，2017 年。 网址 https://www.opencontainers.org/。 [p]

加州大学伯克利分校。 课程概述 | 数据科学部，2017 年。 网址 http://data.berkeley.edu/education/curriculum-overview。 [p]

K. Ushey、J. McPherson、J. Cheng、A. Atkins 和 J. Allaire。 *打包机：一个依赖-项目及其 R 包依赖项的 dency 管理系统*，2016。 https://CRAN.R-project.org/package=packrat。 R 包版本 0.4.8-1。 [p] 网址

H.威克姆。 *整洁诗句：轻松安装和加载“Tidyverse”软件包*，2017。 https://CRAN.R-project.org/package=tidyverse。 R 包版本 1.1.1。 [p] 网址

*卡尔·波提格*
*加州大学伯克利分校*
*ESPM 系，加州大学，130 Mulford Hall*
*Berkeley，CA 94720-3114，美国*
cboettig@berkeley.edu

*德克·埃德比特尔*
*Debian 和 R 项目；Ketchum Trading 美国伊利诺伊州芝加哥*
edd@debian.org