可重复性数据分析及其工业实践

第 13 届中国 R 语言会议

黄湘云

2021 年 01 月 03 日

文学编程 [1]

Literate programming is a programming paradigm introduced by Donald Knuth in which a computer program is given an explanation of its logic in a natural language, such as English, interspersed with snippets of macros and traditional source code, from which compilable source code can be generated.

[1] https://en.wikipedia.org/wiki/Literate_programming

Sweave I

下面以 utils 包提供的测试文档为例

```
testfile <- system.file("Sweave", "Sweave-test-1.Rnw",
                        package = "utils")
Sweave(testfile)
tools::texi2pdf("Sweave-test-1.tex")
```

注意

若已安装 LaTeX 发行版 TinyTeX[2]，需要再装三个 LaTeX 宏包 a4wide、ntgclass 和 ae，以供文档编译

```
tinytex::tlmgr_install(c('a4wide', 'ntgclass','ae'))
```

[2] https://github.com/yihui/tinytex

Sweave II

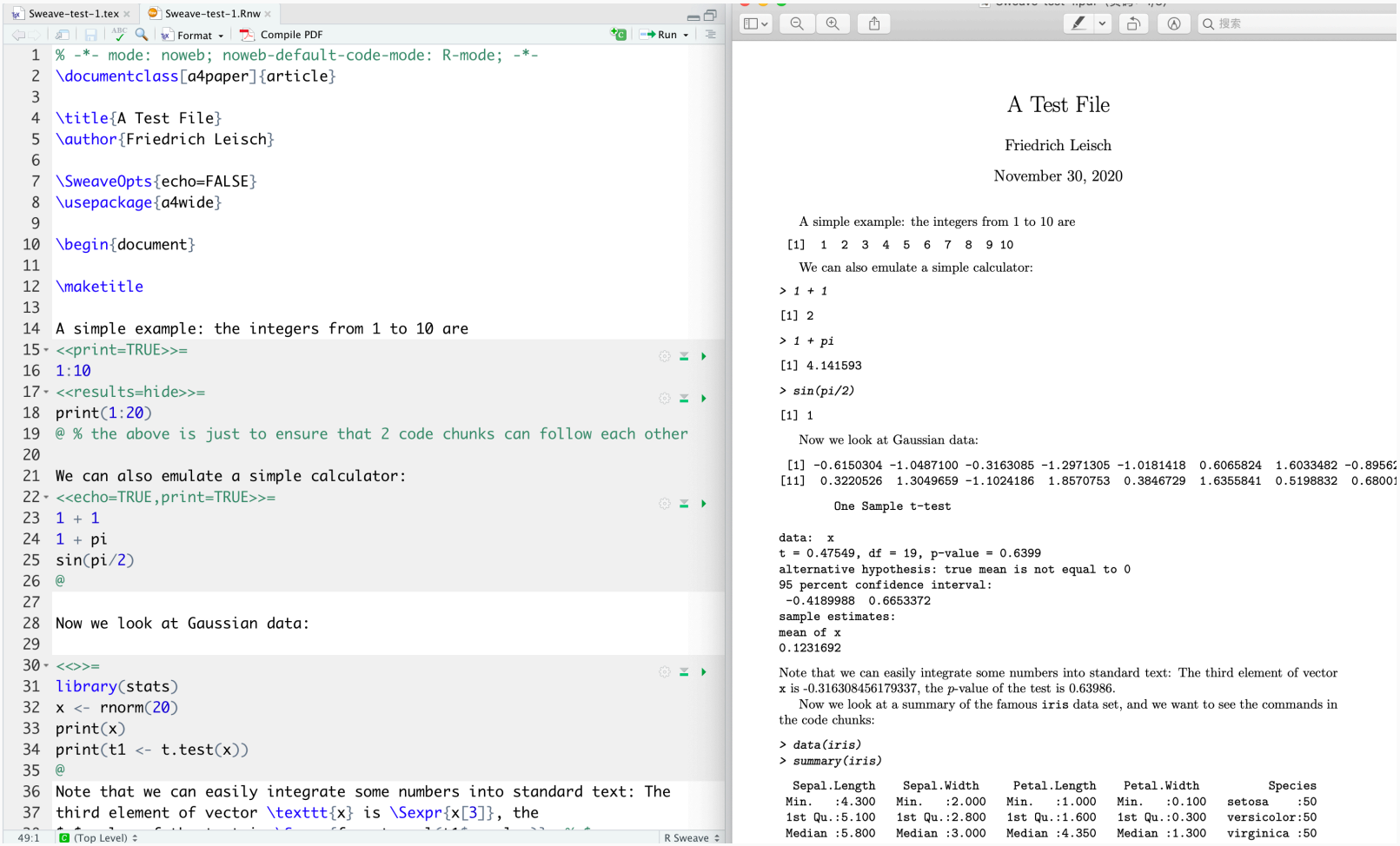


图 1: 形式上是 LaTeX 和 R 代码的混合物

A minimal R Markdown document [3]

```
---
title: "A Simple Regression"  #----
author: "Yihui Xie"           #    |
date: "2019-01-02"            #    |
output:                       #    |--> metadata
  html_document:              #    |
    toc: true                 #----
---

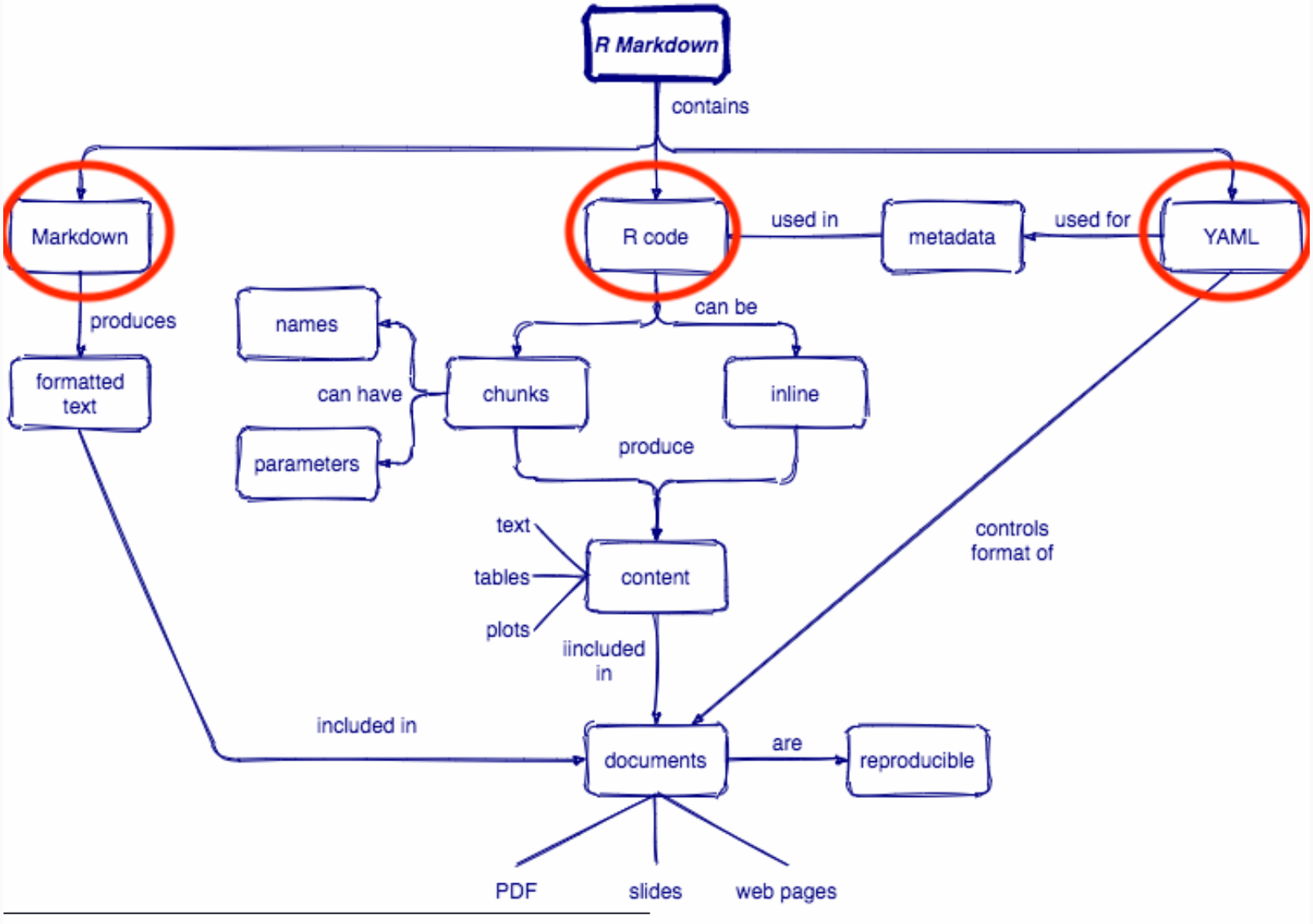
We built a linear regression model.  <!-- narrative -->
```{r}
fit <- lm(dist ~ speed, data = cars)
b   <- coef(fit)
plot(fit)
```

The slope of the regression is `r b[1]`. <!-- narrative w/ code --
```

[3] https://slides.yihui.org/2020-covid-rmarkdown.html#7

Knit R Markdown to Markdown

````
---
title: "A Simple Regression"  #----
author: "Yihui Xie"           #    |
date: "2019-01-02"            #    |
output:                       #    |--> metadata
  html_document:              #    |
    toc: true                 #----
---

We built a linear regression model.  <!-- narrative -->
```r
fit <- lm(dist ~ speed, data = cars)
b   <- coef(fit)
plot(fit)
```

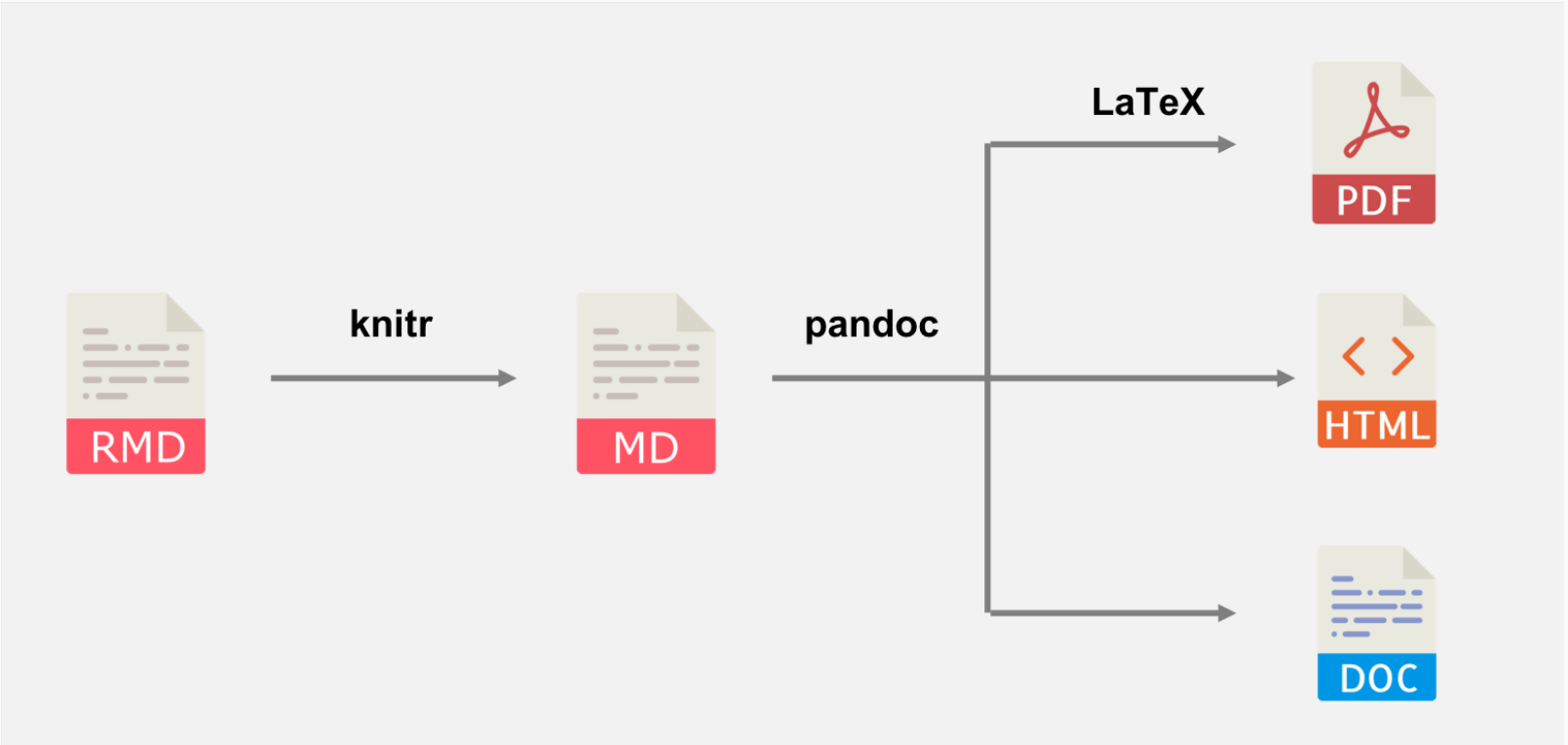
![a plot](input_files/figure-html/unnamed-chunk-1.png)
The slope of the regression is -17.58.  <!-- narrative w/ code -->
````

R Markdown 思维导图[4]



[4] https://github.com/rstudio/concept-maps#r-markdown

编译 R Markdown 文档的过程[5]



R Markdown = knitr (Literate Programming)

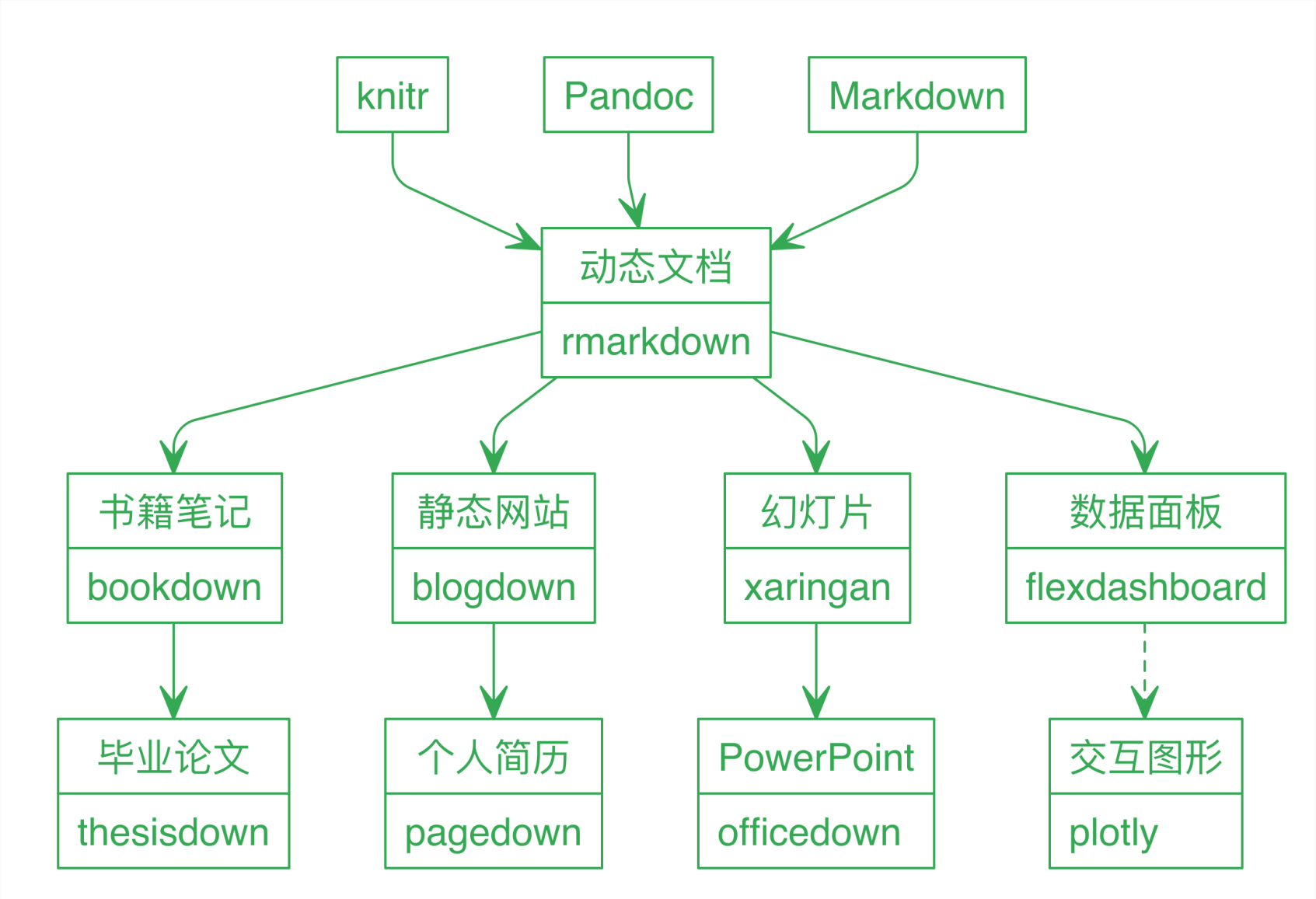
\+ Pandoc

(+ LaTeX for PDF output)

[5] https://slides.yihui.org/2020-covid-rmarkdown.html#7

从 Sweave 到 R Markdown

R Markdown 生态环境

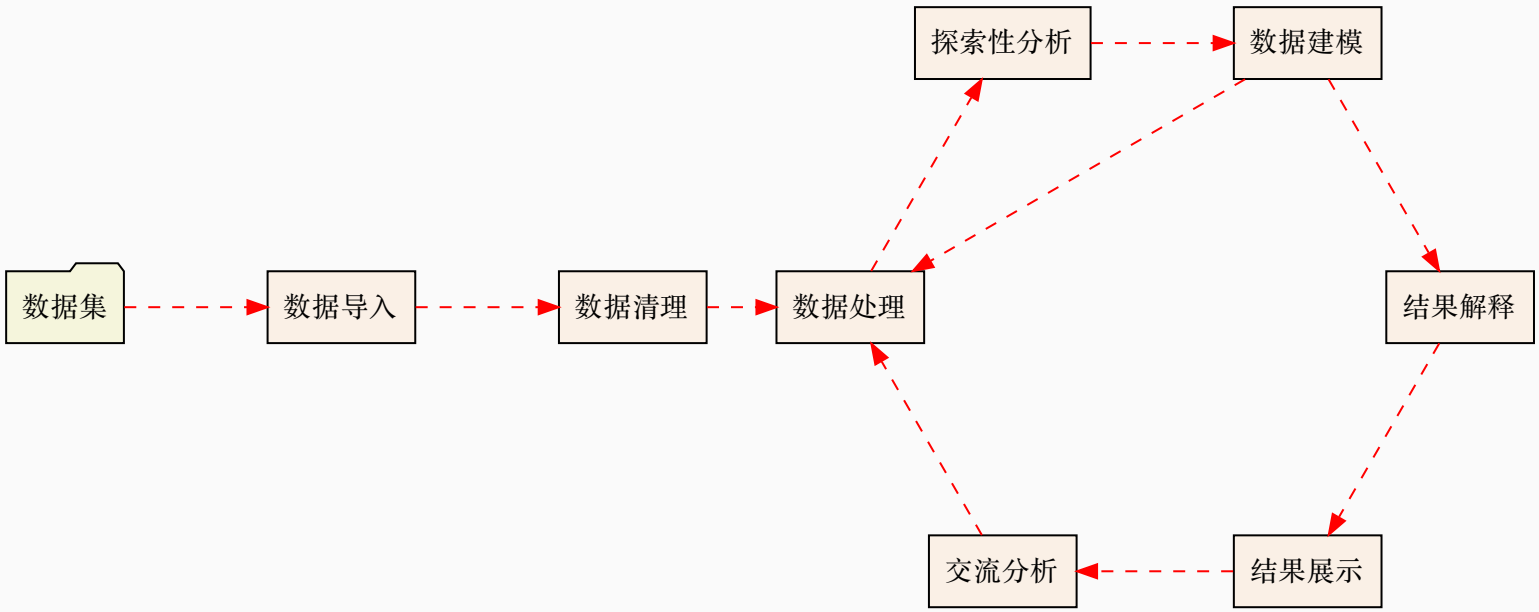


R Markdown 过去 vs 现在

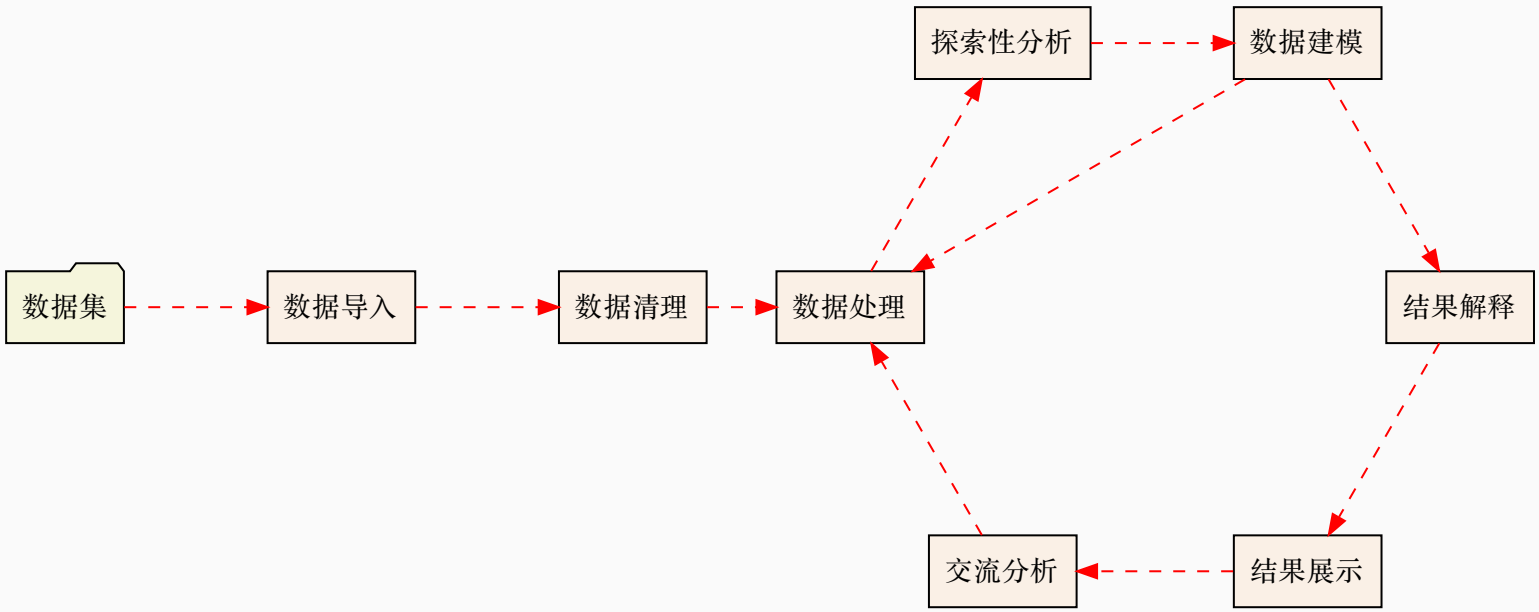
最大的不同在于稳健性和生态圈，基本覆盖 Word / LaTeX 支持的文档处理功能，满足大部分数据分析的场景，比如毕业论文、期刊论文、课程作业、幻灯片、分析报告等。高级的特性依然需要相当的技术实力支撑，包括 Pandoc、Lua、LaTeX、CSS、HTML、JavaScript、Git 和 R，在数据分析师这个行业，它们形成了一簇相当陡峭的学习曲线。

然鹅，有时候真正干活的是搜索引擎和 Ctrl + C/V！

数据分析工作流



数据分析工作流



面对大量的数据时，探索性分析是一件极其困难的事情！

探索性分析

```r
# 连接 Spark
library(sparklyr)
sc <- spark_connect(
  master = "local[2]",
  spark_home = Sys.getenv("SPARK_HOME")
)
# 准备数据
tbl <- copy_to(sc, ggplot2::diamonds, "diamonds")

# 数据查询
library(DBI)
diamonds_db <- dbGetQuery(sc, "
  SELECT count(*) as cnt, cut FROM diamonds GROUP BY cut
")
```

探索性分析 （续）

```
# 关闭连接
spark_disconnect(sc)
# 数据探索
library(ggplot2)
ggplot(diamonds_db, aes(cut, cnt)) +
  geom_col() +
  theme_minimal()

# 数据呈现
library(plotly)
diamonds_db %>%
  plot_ly(x = ~cut, y = ~cnt, type = "bar") %>%
  add_text(
    text = ~ scales::comma(cnt), y = ~cnt,
    textposition = "top middle",
```

探索性分析 （续）

```
    cliponaxis = FALSE, showlegend = FALSE
) %>%
config(displayModeBar = FALSE)
```

然鹅。。。

请从配置环境开始

```
# 安装 openjdk 11
brew install openjdk@11
# 全局设置 JDK 11
sudo ln -sfn /usr/local/opt/openjdk@11/libexec/openjdk.jdk \
  /Library/Java/JavaVirtualMachines/openjdk-11.jdk
# Java 11 JDK 添加到 .zshrc
export CPPFLAGS="-I/usr/local/opt/openjdk@11/include"
export PATH="/usr/local/opt/openjdk@11/bin:$PATH"
# 配置 R 环境
sudo R CMD javareconf
# 安装 rJava 包
Rscript -e 'install.packages(c("rJava", "sparklyr"))'
```

然鹅，请从下载软件开始

手动从官网下载 Spark 软件，配置变量。。。

然鹅，请从下载软件开始

手动从官网下载 Spark 软件，配置变量。。。

警告

Spark 依赖特定版本的 Java、Hadoop，三者之间的版本应该要相融。

然鹅，请从下载软件开始

手动从官网下载 Spark 软件，配置变量。。。

警告

Spark 依赖特定版本的 Java、Hadoop，三者之间的版本应该要相融。

回到开头，重新下载软件，配置环境，再来三遍。

一周过去了，终于可以开始干点活了。。。

数据可视化

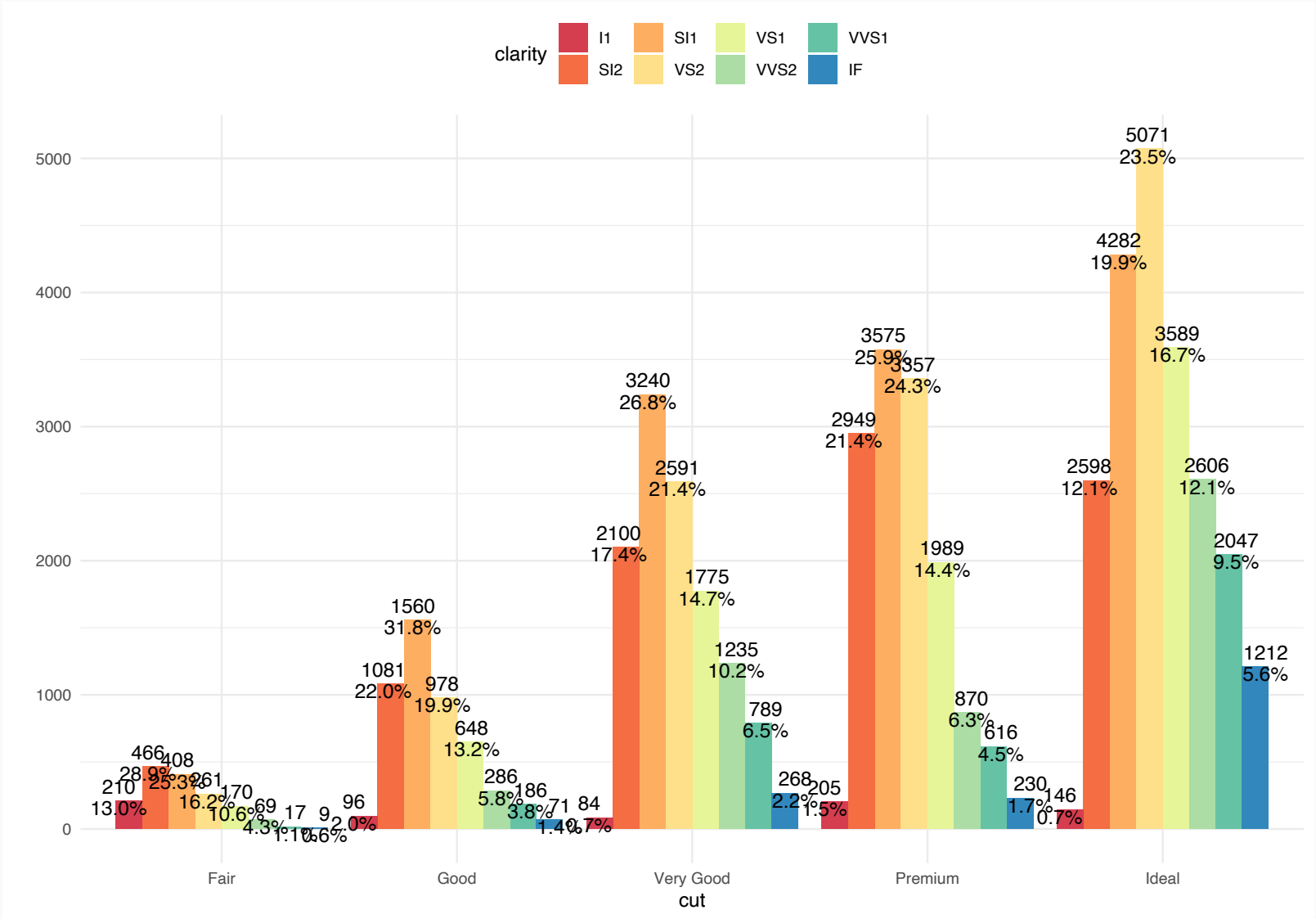

clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
5000
4000
3000
2000
1000
0
Fair
Good
Very Good
Premium
Ideal
cut
210
13.0%
466
28.9%
408
25.3%
261
16.2%
170
10.6%
69
4.3%
17
1.1%
9
0.6%
96
2.0%
1081
22.0%
1560
31.8%
978
19.9%
648
13.2%
286
5.8%
186
3.8%
71
1.4%
84
1.7%
2100
17.4%
3240
26.8%
2591
21.4%
1775
14.7%
1235
10.2%
789
6.5%
268
2.2%
205
1.5%
2949
21.4%
3575
25.9%
3357
24.3%
1989
14.4%
870
6.3%
616
4.5%
230
1.7%
146
0.7%
2598
12.1%
4282
19.9%
5071
23.5%
3589
16.7%
2606
12.1%
2047
9.5%
1212
5.6%

图 2: 柱形图

数据可视化（续）

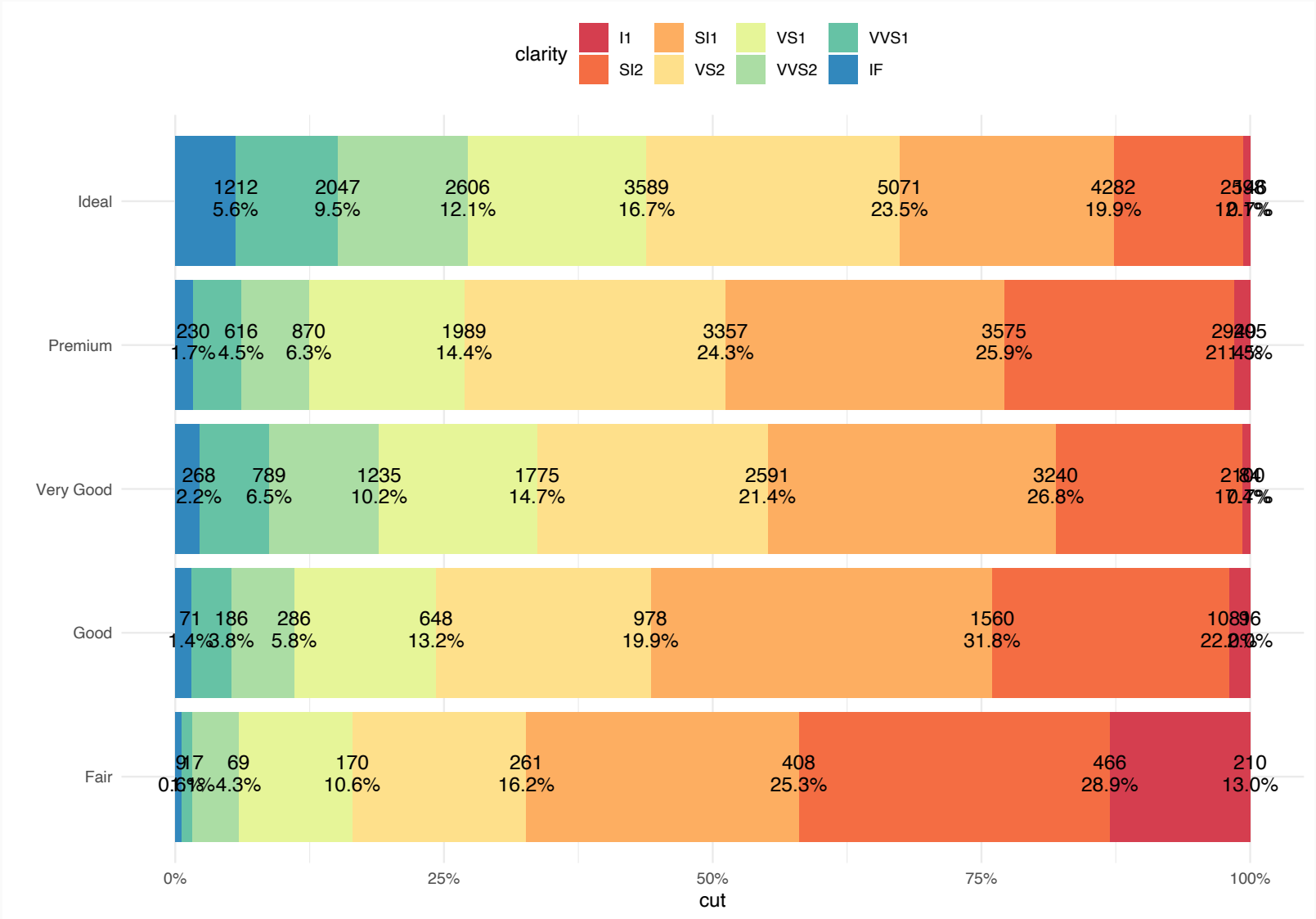


图 3: 条形图

然鹅，要这样。。。

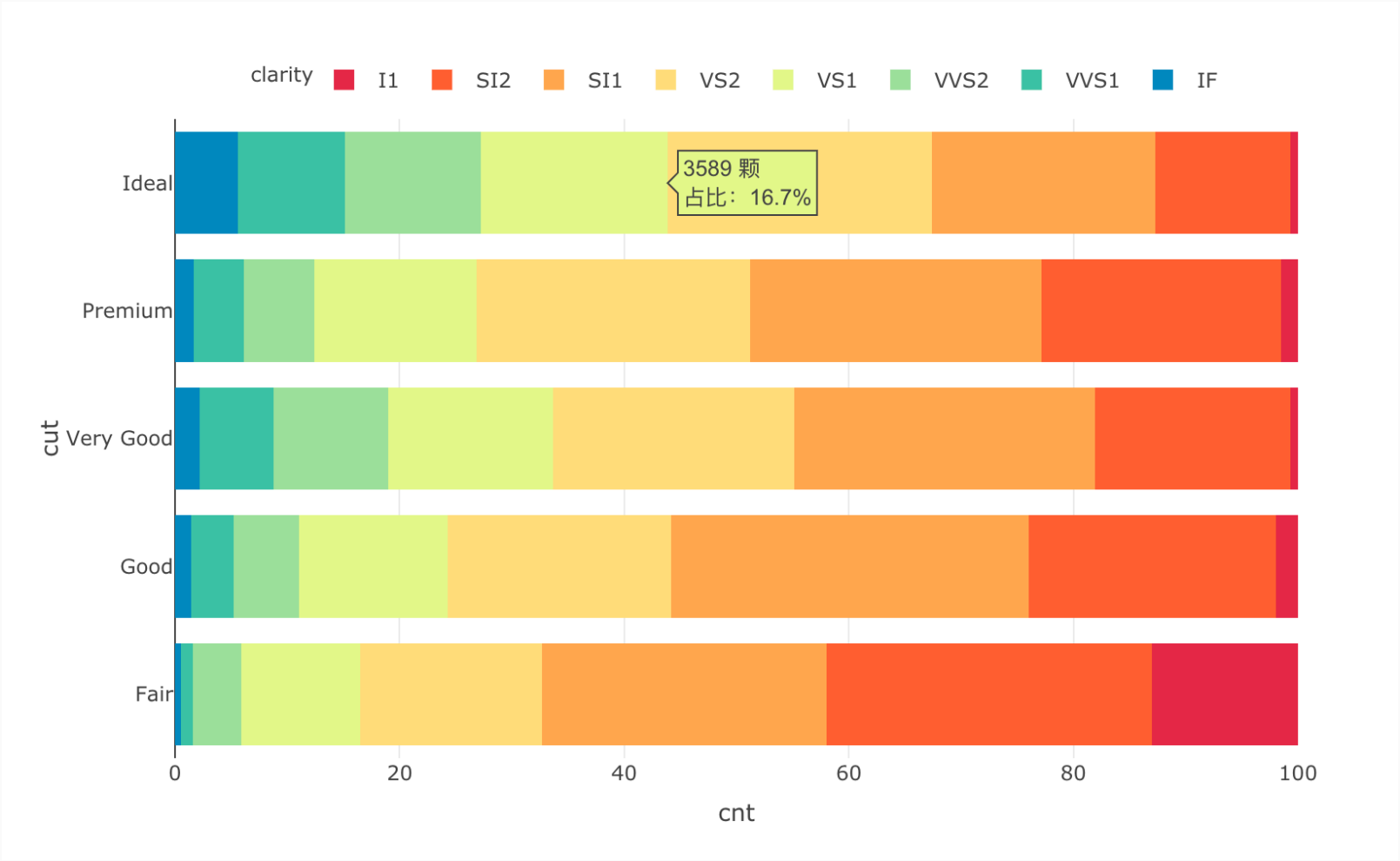
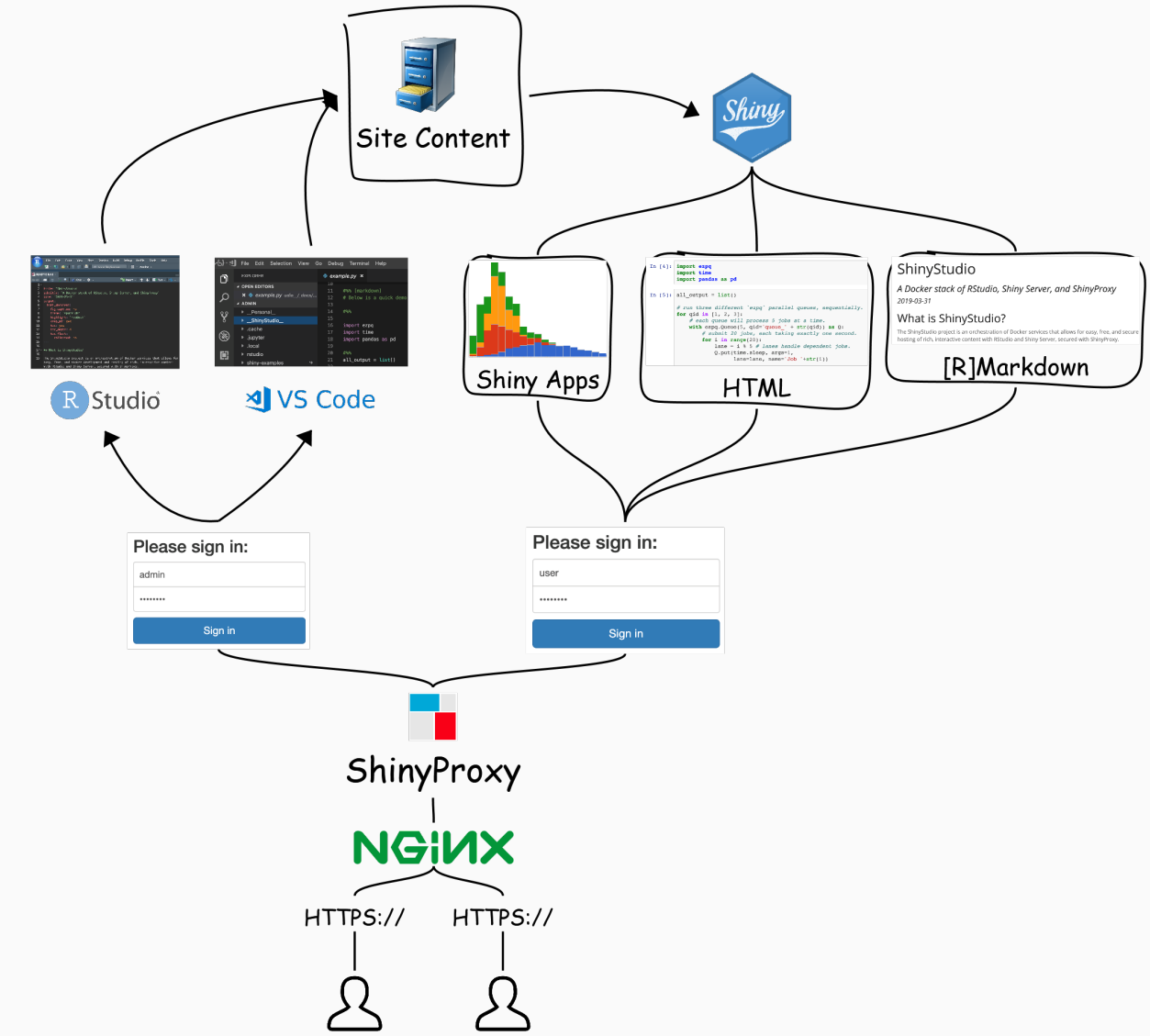


图 4: 交互条形图

工业实践

基础设施 [6]



[6] https://github.com/openanalytics

分层结构

Q & A

谢谢