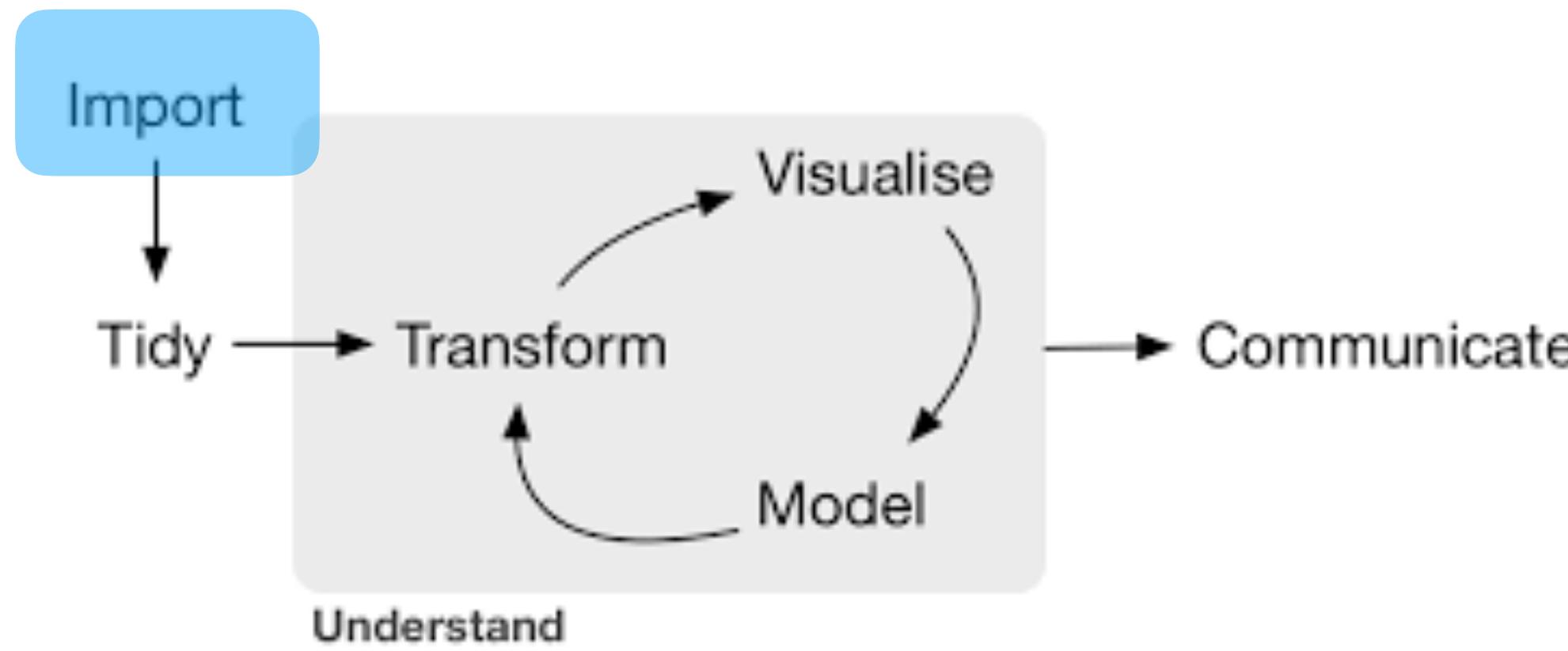


# **dbcop** Turn any database into an R package

David Robinson  
2021-09-10

# A lot of the EDA flow is built around reading data into memory

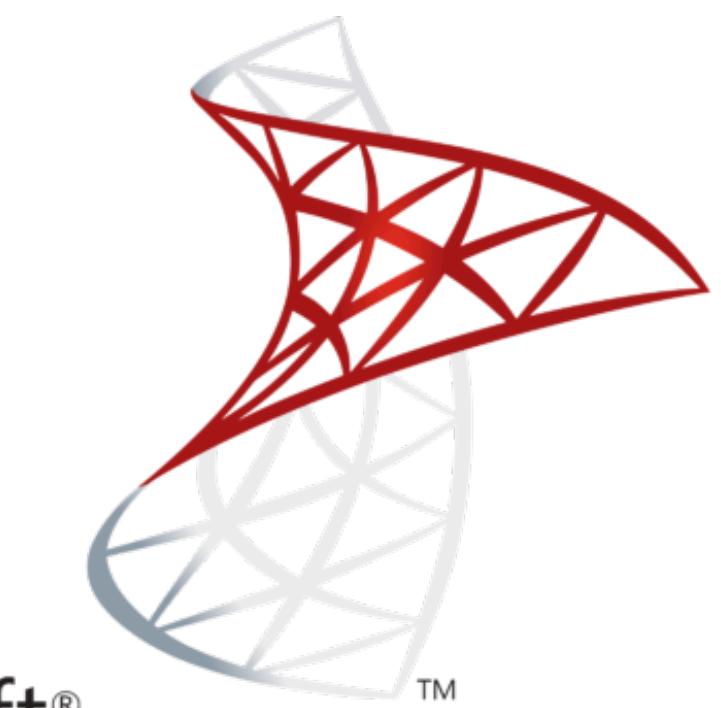
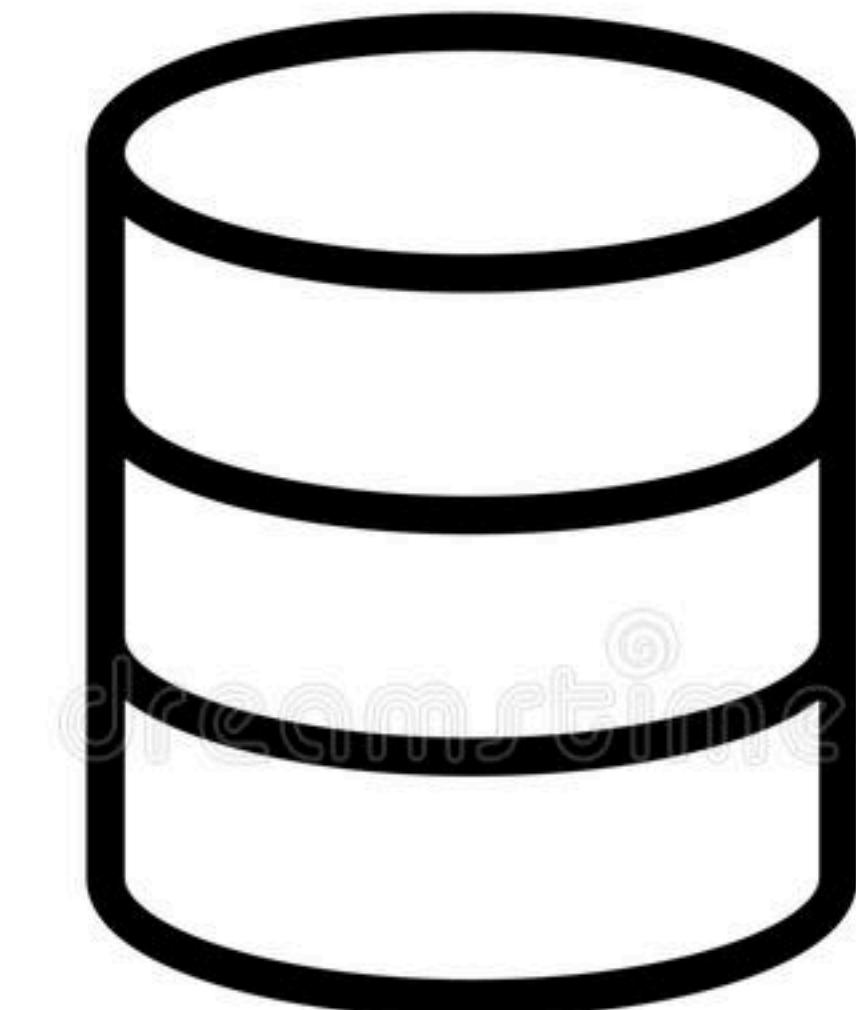


<code>A B C 1 2 3 4 5 NA</code>	→	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1	2	3	4	5	NA	<b>read_delim("file.txt", delim = " ")</b> Read files with any delimiter. If no delimiter is specified, it will automatically guess. To make file.txt, run: <code>write_file("A B C\n1 2 3\n4 5 NA", file = "file.txt")</code>
A	B	C										
1	2	3										
4	5	NA										
<code>A,B,C 1,2,3 4,5,NA</code>	→	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1	2	3	4	5	NA	<b>read_csv("file.csv")</b> Read a comma delimited file with period decimal marks. <code>write_file("A,B,C\n1,2,3\n4,5,NA", file = "file.csv")</code>
A	B	C										
1	2	3										
4	5	NA										
<code>A;B;C 1,5;2;3 4,5;5;NA</code>	→	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1.5</td><td>2</td><td>3</td></tr><tr><td>4.5</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1.5	2	3	4.5	5	NA	<b>read_csv2("file2.csv")</b> Read semicolon delimited files with comma decimal marks. <code>write_file("A;B;C\n1,5;2;3\n4,5;5;NA", file = "file2.csv")</code>
A	B	C										
1.5	2	3										
4.5	5	NA										
<code>A B C 1 2 3 4 5 NA</code>	→	<table border="1"><thead><tr><th>A</th><th>B</th><th>C</th></tr></thead><tbody><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>NA</td></tr></tbody></table>	A	B	C	1	2	3	4	5	NA	<b>read_tsv("file.tsv")</b> Read a tab delimited file. Also <b>read_table()</b> . <b>read_fwf("file.tsv", fwf_widths(c(2, 2, NA)))</b> Read a fixed width file. <code>write_file("A\tB\tC\n1\t2\t3\n4\t5\tNA", file = "file.tsv")</code>
A	B	C										
1	2	3										
4	5	NA										

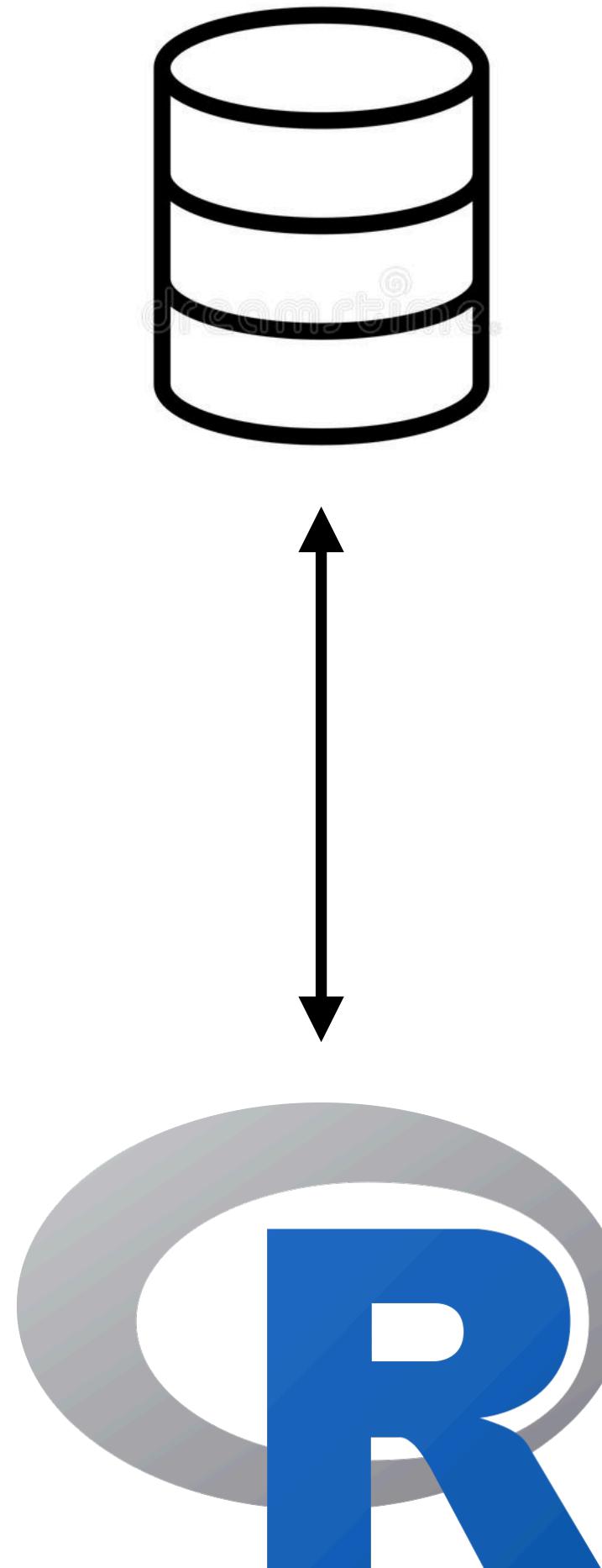
# Most organizations work with data in a database



Postgre**SQL**



# Working with a db should be as easy for your team as possible



## Build a Corporate R Package for Pleasure and Profit

Brad Lindblad Mar 5, 2019 · 9 min read

[Twitter](#) [Facebook](#) [LinkedIn](#) [Email](#) [Link](#)



Using R packages and education to scale Data Science at Airbnb

AirbnbEng [Follow](#) [Email](#)  
Mar 29, 2016 · 9 min read

[Twitter](#) [Facebook](#) [LinkedIn](#) [Email](#) [Link](#)

By Ricardo Bion



oRganization

Emily Riederer @emilyriederer emily.rbind.io

How to make internal R packages a part of your team



I first started appreciating internal packages when I was working as a data scientist at Stack Overflow

```
library(sqlstackr)
```

[dgrtwo / dbcooper](https://github.com/dgrtwo/dbcooper) Public

Code Issues 1 Pull requests Actions Projects Wiki Security Insights Settings

master 2 branches 0 tags Go to file Add file Code

**dgrtwo Merge pull request #5 from dgrtwo/development ...** ba369c7 17 minutes ago 29 commits

R	Merge pull request #5 from dgrtwo/development	17 minutes ago
man-roxygen	Many revisions. Renamed to dbcooper package. See README	2 years ago
man	Adjustments to README; removed dbc_bigquery	21 minutes ago
.gitignore	Add .Rproj file back in	2 years ago
.travis.yml	Changes to prep for initial release	2 years ago
CODE_OF_CONDUCT.md	Changes to prep for initial release	2 years ago
DESCRIPTION	Development changes- WIP	3 days ago
LICENSE	Changes to prep for initial release	2 years ago
NAMESPACE	Adjustments to README; removed dbc_bigquery	21 minutes ago
README.Rmd	Adjustments to README; removed dbc_bigquery	21 minutes ago
README.md	Adjustments to README; removed dbc_bigquery	21 minutes ago
dbcooper.Rproj	Many revisions. Renamed to dbcooper package. See README	2 years ago
remotedb.Rproj	Add .Rproj file back in	2 years ago

README.md

## dbcooper

lifecycle experimental build failing

The dbcooper package turns a database connection into a collection of functions, handling logic for keeping track of connections and letting you take advantage of autocompletion when exploring a database.

It's especially helpful to use when authoring database-specific R packages, for instance in an internal company package or one wrapping a public data source.

The package's name is a reference to the bandit D.B. Cooper.

### Installation

You can install the development version from GitHub with:

```
# install.packages("devtools")
devtools::install_github("dgrtwo/dbcooper")
```

About

Create user-friendly accessor functions from a database connection

Readme View license

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Contributors 2

 **chriscardillo** Chris Cardillo

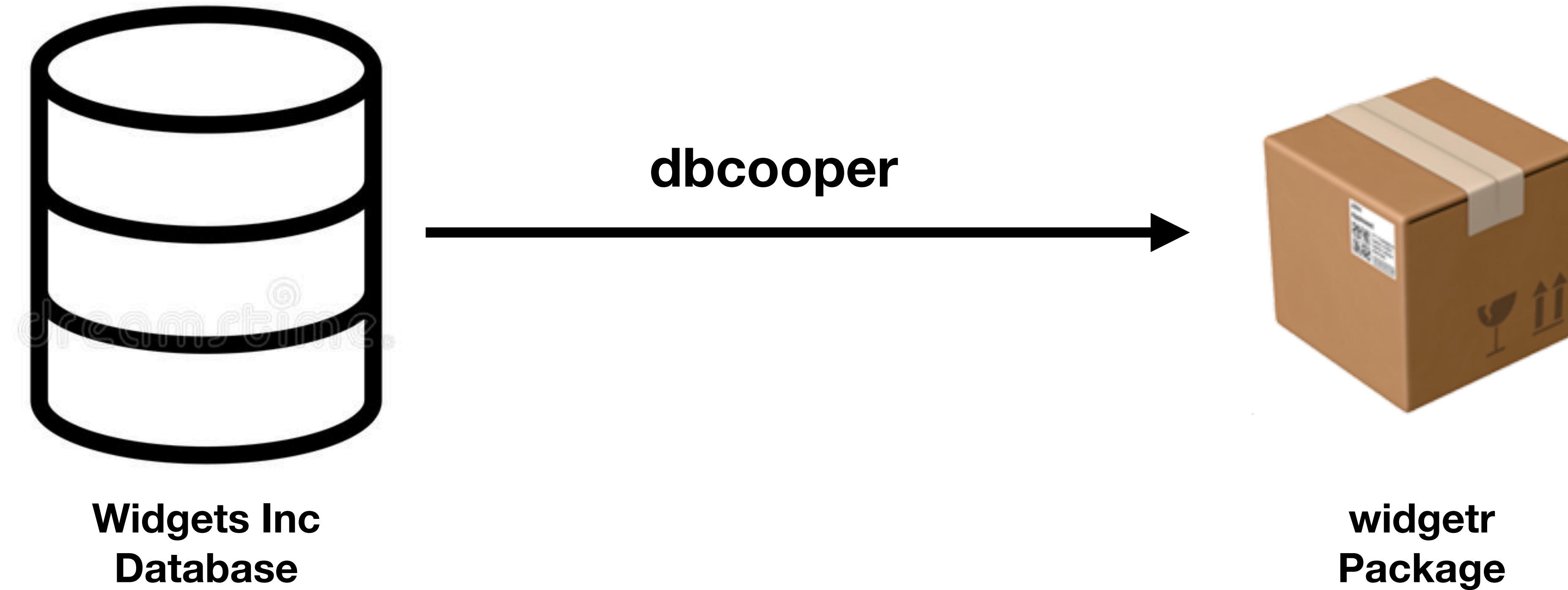
 **dgrtwo** David Robinson

Languages

R 100.0%

<https://github.com/dgrtwo/dbcooper>

# The goal of dbcooper: one step to a database-specific package



**Problem:**  
**What's it like to work with a**  
**database?**

# What do you want to do when analyzing a database?

- List tables
- Work with tables using dplyr
- Run SQL queries
- Execute commands

# Example: Creating a Connection

```
library(dplyr)
library(dbplyr)

lahman_db <- lahman_sqlite()

lahman_db
#> <SQLiteConnection>
#>   Path: /var/folders/8p/xzrrqphx2qb3d2s_fgqrk5xr000gn/T//Rtmp90cGbM/lahman.sqlite
#>   Extensions: TRUE
```

# Example: List tables

```
DBI:::dbListTables(lahman_db)
#> [1] "AllstarFull"           "Appearances"          "AwardsManagers"
#> [4] "AwardsPlayers"        "AwardsShareManagers" "AwardsSharePlayers"
#> [7] "Batting"               "BattingPost"           "CollegePlaying"
#> [10] "Fielding"             "FieldingOF"            "FieldingOFsplit"
#> [13] "FieldingPost"         "HallOfFame"            "HomeGames"
#> [16] "LahmanData"           "Managers"              "ManagersHalf"
#> [19] "Master"                "Parks"                 "People"
#> [22] "Pitching"              "PitchingPost"          "Salaries"
#> [25] "Schools"               "SeriesPost"            "Teams"
#> [28] "TeamsFranchises"      "TeamsHalf"             "sqlite_stat1"
#> [31] "sqlite_stat4"
```

# Example: Get a remote table

```
tbl(lahman_db, "Batting")
#> # Source:  table<Batting> [?? x 22]
#> # Database: sqlite 3.36.0
#> #   [/var/folders/8p/xzrrqphx2qb3d2s_fgqrk5xr0000gn/lahman.sqlite]
#> playerID  yearID  stint  teamID  lgID      G    AB    R     H    X2B    X3B    HR
#> <chr>      <int> <int> <chr> <chr> <int> <int> <int> <int> <int> <int> <int>
#> 1 abercda01  1871     1 TR0    NA     1     4     0     0     0     0     0     0
#> 2 addybo01   1871     1 RC1    NA    25    118    30    32     6     0     0
#> 3 allisar01  1871     1 CL1    NA    29    137    28    40     4     5     0
#> 4 allisdo01  1871     1 WS3    NA    27    133    28    44    10     2     2
#> 5 ansonca01  1871     1 RC1    NA    25    120    29    39    11     3     0
#> 6 armstbo01  1871     1 FW1    NA    12    49     9    11     2     1     0
#> 7 barkeal01  1871     1 RC1    NA     1     4     0     1     0     0     0
#> 8 barnero01  1871     1 BS1    NA    31    157    66    63    10     9     0
#> 9 barrebi01  1871     1 FW1    NA     1     5     1     1     1     0     0
#> 10 barrofr01 1871     1 BS1   NA    18    86    13    13     2     1     0
#> # ... with more rows, and 10 more variables: RBI <int>, SB <int>, CS <int>,
#> #   BB <int>, S0 <int>, IBB <int>, HBP <int>, SH <int>, SF <int>, GIDP <int>
```

# Example: Run a SQL query

```
tbl(lahman_db, sql("SELECT playerID, COUNT(*) FROM Batting GROUP BY playerID"))
#> # Source:   SQL [?? x 2]
#> # Database: sqlite 3.36.0
#> #   [/var/folders/8p/xzrrqphx2qb3d2s_fgqrk5xr0000gn/lahman.sqlite]
#>   playerID `COUNT(*)`
#>   <chr>      <int>
#> 1 aardsda01      9
#> 2 aaronha01     23
#> 3 aaronto01      7
#> 4 aasedo01     13
#> 5 abadan01      3
#> 6 abadfe01     10
#> 7 abadijo01      2
#> 8 abbated01     10
#> 9 abbeybe01      6
#> 10 abbeych01      5
#> # ... with more rows
```

# Example: Execute a command

```
DBI::dbExecute(lahman_db, "DROP TABLE Batting")
#> [1] 0
```

```
DBI::dbListTables(lahman_db)
```

```
tbl(lahman_db, sql("Batting"))
```

```
tbl(lahman_db, sql("SELECT playerID, COUNT(*)
                      FROM Batting
                      GROUP BY playerID"))
```

```
DBI::dbExecute(lahman_db, "DROP TABLE Batting")
```

```
DBI::dbListTables(lahman_db)
```

```
tbl(lahman_db, sql("Batting"))
```

```
tbl(lahman_db, sql("SELECT playerID, COUNT(*)  
                      FROM Batting  
                     GROUP BY playerID"))
```

```
DBI::dbExecute(lahman_db, "DROP TABLE Batting")
```



R examples.R x

Source on Save



Run

Source

```

1 library(dplyr)
2 library(dbplyr)
3
4 # Create a connection
5 lahman_db <- lahman_sqlite()
6
7 DBI::dbListTables(lahman_db)
8
9 tbl(lahman_db, sql("Batting"))
10
11 tbl(lahman_db, sql("SELECT playerID, COUNT(*)
12           FROM Batting
13           GROUP BY playerID"))
14
15 DBI::dbExecute(lahman_db, "DROP TABLE Batting")

```

15:48 (Top Level) R Script

Environment
History
Connections
Build
Git

287 MiB

R Global Environment

Data

	lahman_db	Formal class SQLiteConnection	
--	-----------	-------------------------------	--

Console Terminal x Jobs x

R 4.1.0 · ~/Repositories/rpackages/stackbigquery/

```

n.sqlite
playerID `COUNT(*)` 
<chr>      <int>
1 aardsda01     9
2 aaronha01    23
3 aaronto01     7
4 aasedo01     13
5 abadan01      3
6 abadfe01     10
7 abadijo01      2
8 abbated01    10
9 abbeybe01      6
10 abbeych01     5
# ... with more rows
>
> DBI::dbExecute(lahman_db, "DROP TABLE Batting")
[1] 0
>

```

Files Plots Packages Help Viewer

R: Cache and retrieve an 'src\_sqlite' of the Lahman baseball... Find in Topic

lahman {dbplyr}

## Cache and retrieve an `src_sqlite` of the Lahman baseball database.

### Description

This creates an interesting database using data from the Lahman baseball data source, provided by Sean Lahman at <http://www.seanlahman.com/baseball-archive/statistics/>, and made easily available in R through the **Lahman** package by Michael Friendly, Dennis Murphy and Martin Monkman. See the documentation for that package for documentation of the individual tables.

### Usage

```

lahman_sqlite(path = NULL)

lahman_postgres(dbname = "lahman", host = "localhost", ...)

```

R examples.R x

Source on Save



Run
Source

```

1 library(dplyr)
2 library(dbplyr)
3
4 # Create a connection
5 lahman_db <- lahman_sqlite()
6
7 DBI::dbListTables(lahman_db)
8
9 tbl(lahman_db, sql("Batting"))
10
11 tbl(lahman_db, sql("SELECT playerID, COUNT(*)
12           FROM Batting
13           GROUP BY playerID"))
14
15 DBI::dbExecute(lahman_db, "DROP TABLE Batting")

```

15:48 (Top Level) R Script

Environment
History
Connections
Build
Git

287 MiB

R Global Environment

Data

lahman_db	Formal class SQLiteConnection
-----------	-------------------------------

Console Terminal x Jobs x

R 4.1.0 · ~/Repositories/rpackages/stackbigquery/ ↗

```

n.sqlite
playerID `COUNT(*)` 
<chr>      <int>
1 aardsda01     9
2 aaronha01   23
3 aaronto01     7
4 aasedo01    13
5 abadan01     3
6 abadfe01   10
7 abadijo01     2
8 abbated01   10
9 abbeybe01     6
10 abbeych01    5
# ... with more rows
>
> DBI::dbExecute(lahman_db, "DROP TABLE Batting")
[1] 0
>

```

Files Plots Packages Help Viewer

R: Cache and retrieve an 'src\_sqlite' of the Lahman baseball... Find in Topic

lahman {dbplyr}

## Cache and retrieve an `src_sqlite` of the Lahman baseball database.

### Description

This creates an interesting database using data from the Lahman baseball data source, provided by Sean Lahman at <http://www.seanlahman.com/baseball-archive/statistics/>, and made easily available in R through the **Lahman** package by Michael Friendly, Dennis Murphy and Martin Monkman. See the documentation for that package for documentation of the individual tables.

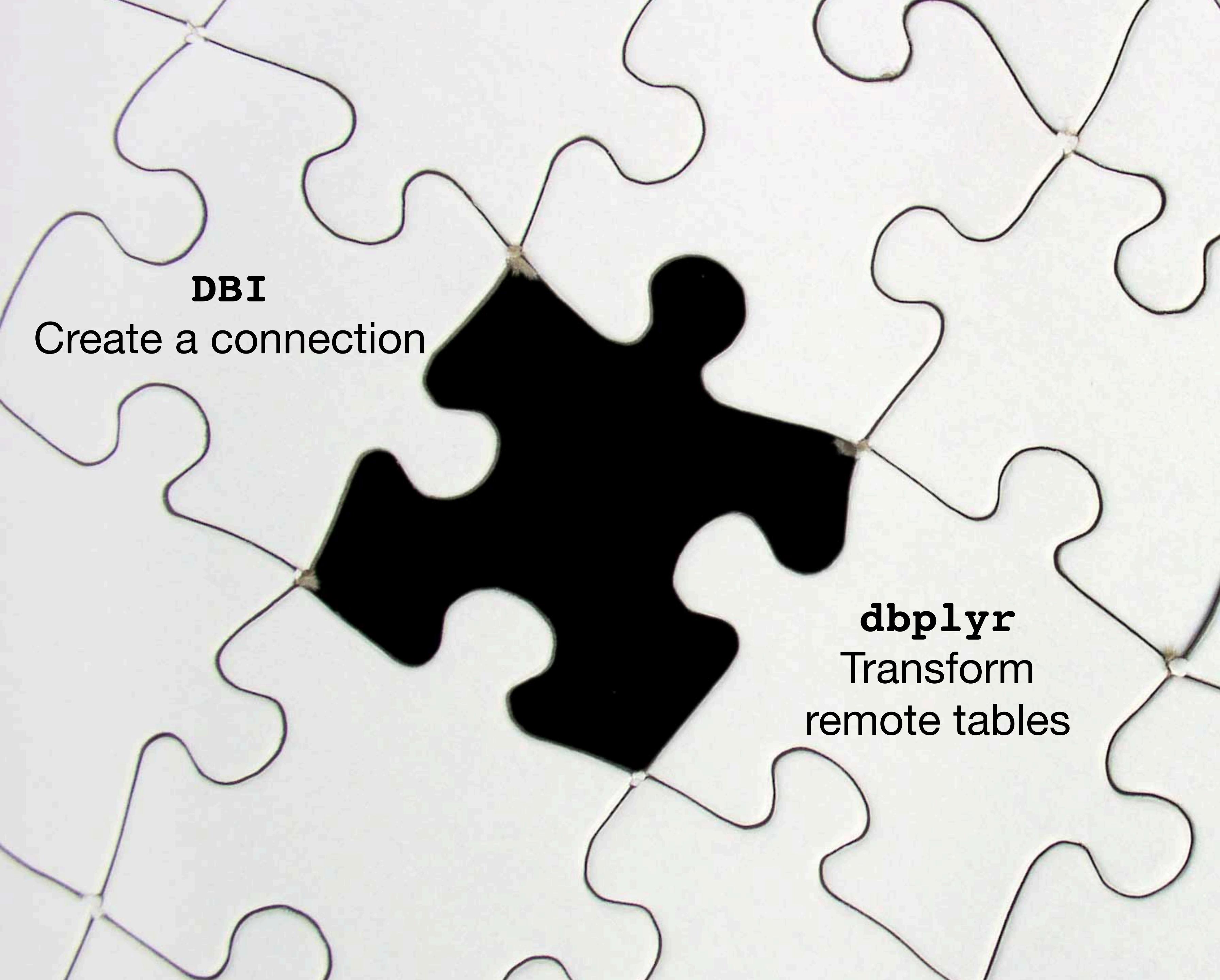
### Usage

```

lahman_sqlite(path = NULL)

lahman_postgres(dbname = "lahman", host = "localhost", ...)

```



**DBI**  
Create a connection

**dbplyr**  
Transform  
remote tables

# Solution

**dbcooper turns a database into a  
collection of functions**

```
dbcooper::dbc_init(lahman_db, "lahman")
```

- Stores and manages the connection globally
- Generates prefixed functions for tasks
- Generates autocomplete-friendly accessors for tables

# dbcoppper creates prefixed functions for common tasks

```
dbcoppper::dbc_init(lahman_db, "lahman")
```

```
DBI::dbListTables(lahman_db)
```

```
lahman_list()
```

```
tbl(lahman_db, sql("Batting"))
```

```
lahman_tbl("Batting")
```

```
tbl(lahman_db, sql("SELECT playerID, COUNT(*)  
FROM Batting  
GROUP BY playerID"))
```

```
lahman_query("SELECT playerID, COUNT(*)  
FROM Batting  
GROUP BY playerID")
```

```
DBI::dbExecute(lahman_db, "DROP TABLE Batting")
```

```
lahman_execute("DROP TABLE Batting")
```

# Access tables with autocomplete

```
lahman_tbl("Batting")
```

```
lahman_tbl("Fielding")
```

```
lahman_tbl("Teams")
```

```
lahman_batting()
```

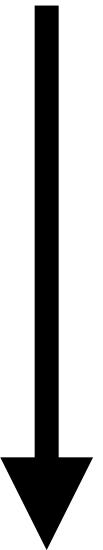
```
lahman_fielding()
```

```
lahman_teams()
```



# Access tables with autocomplete

```
tbl(lahman_db, "Batting") %>%  
  inner_join(tbl(lahman_db, "Fielding"), by = "playerID") %>%  
  left_join(tbl(lahman_db, "Salaries"), by = c("playerID", "yearID"))
```



```
lahman_batting() %>%  
  inner_join(lahman_fielding(), by = "playerID") %>%  
  left_join(lahman_salaries(), by = c("playerID", "yearID"))
```

**Example:**  
**NYT COVID-19 Database**

# BigQuery Public Databases

The screenshot shows the BigQuery web interface. At the top, there are three navigation links: 'FEATURES & INFO', 'SHORTCUT', and 'DISABLE EDITOR TABS'. Below this is the 'Explorer' section, which includes a search bar ('Type to search') and a list of pinned projects. The 'us\_states' table is selected in the Explorer, indicated by a blue highlight. To the right, the 'EDITOR' tab is active, showing the 'us\_states' table schema. The schema details five columns:

Field name	Type	Mode	Policy Tags	Description
date	DATE	NULLABLE		
state_name	STRING	NULLABLE		
state_fips_code	STRING	NULLABLE		
confirmed_cases	INTEGER	NULLABLE		
deaths	INTEGER	NULLABLE		

At the bottom of the schema view, there are two buttons: 'EDIT SCHEMA' and 'VIEW ROW ACCESS POLICIES'.

Learn more: [BigQuery Public Databases](#)

# Example: NYT COVID-19 Data

```
library(dbcooper)

# Use any approach to construct the connection
con <- DBI::dbConnect(bigrquery::bigquery(),
                      project = "bigquery-public-data",
                      dataset = "covid19_nyt",
                      billing = Sys.getenv("BIGQUERY_BILLING_PROJECT"))
```

# Example: NYT COVID-19 Data

```
library(dbcooper)

# Use any approach to construct the connection
con <- DBI::dbConnect(bigrquery::bigquery(),
                      project = "bigquery-public-data",
                      dataset = "covid19_nyt",
                      billing = Sys.getenv("BIGQUERY_BILLING_PROJECT"))

# Create functions with prefix covid_
dbc_init(con, "covid")
```

# Exploring COVID-19 Data

```
# Create functions with prefix covid_
dbc_init(con, "covid")
```

# Exploring COVID-19 Data

```
# Create functions with prefix covid_
dbc_init(con, "covid")

covid_list()
#> [1] "excess_deaths"      "mask_use_by_county" "us_counties"
#> [4] "us_states"
```

# Exploring COVID-19 Data

```
# Create functions with prefix covid_
dbc_init(con, "covid")

covid_list()
#> [1] "excess_deaths"      "mask_use_by_county" "us_counties"
#> [4] "us_states"

covid_us_states()
#> # Source:   SQL [?? x 5]
#> # Database: BigQueryConnection
#>   date      state_name state_fips_code confirmed_cases deaths
#>   <date>    <chr>        <chr>           <int>      <int>
#> 1 2020-03-15 Guam       66                  3          0
#> 2 2020-03-16 Guam       66                  3          0
#> 3 2020-03-17 Guam       66                  3          0
#> 4 2020-03-18 Guam       66                  8          0
#> 5 2020-03-19 Guam       66                 12          0
#> 6 2020-03-20 Guam       66                 14          0
#> 7 2020-03-21 Guam       66                 15          0
#> 8 2020-03-22 Guam       66                 27          1
#> 9 2020-03-23 Guam       66                 29          1
#> 10 2020-03-24 Guam      66                 32          1
#> # ... with more rows
```

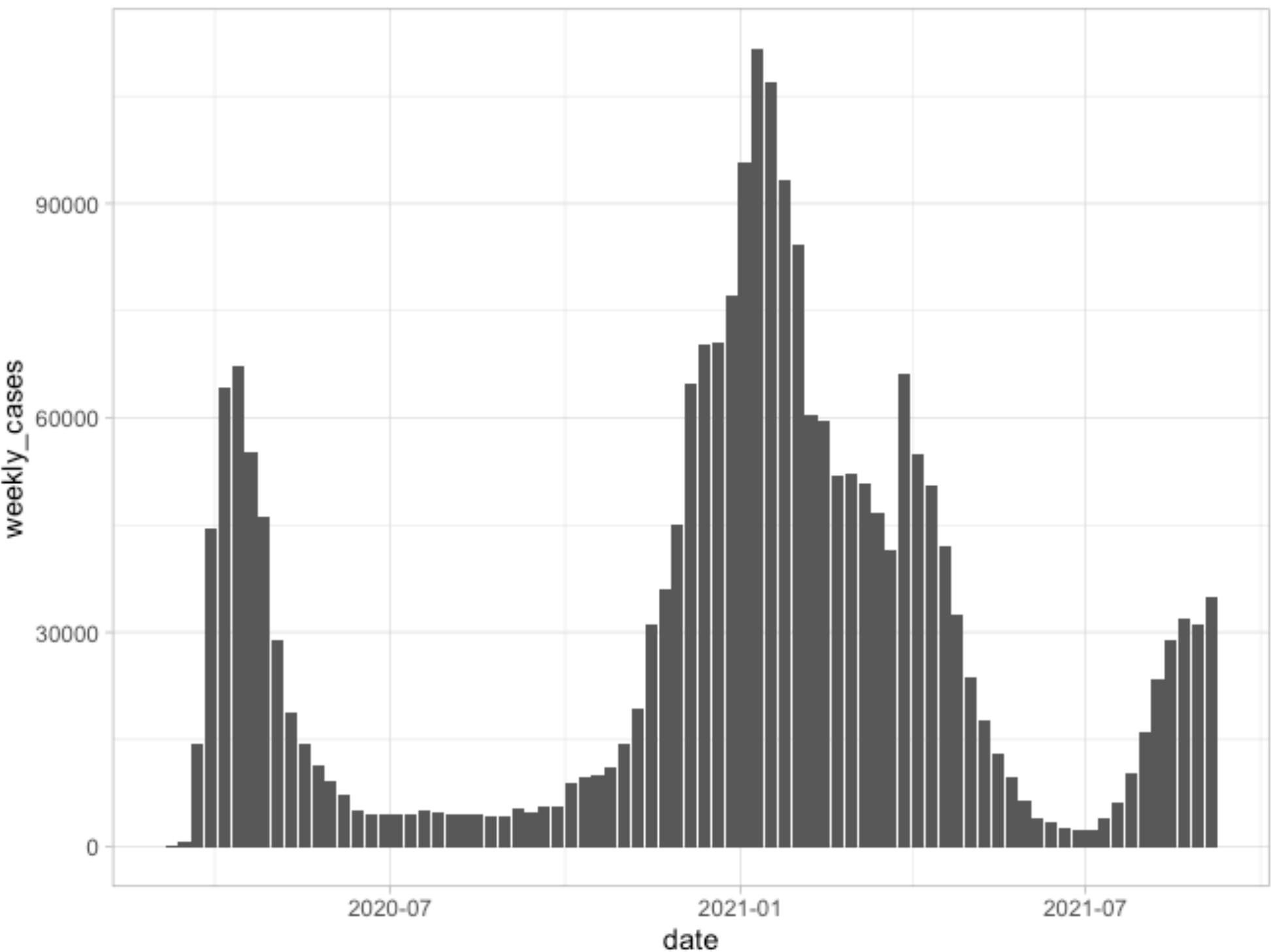
# Exploring COVID-19 Data

```
covid_us_states() %>%
  filter(date == max(date)) %>%
  arrange(desc(confirmed_cases))

#> # Source:      lazy query [?? x 5]
#> # Database:    BigQueryConnection
#> # Ordered by: desc(confirmed_cases)
#>   date      state_name      state_fips_code confirmed_cases deaths
#>   <date>    <chr>          <chr>                <int>     <int>
#> 1 2021-09-08 California       06                  4545832    66722
#> 2 2021-09-08 Texas            48                  3752776    59090
#> 3 2021-09-08 Florida          12                  3378772    46977
#> 4 2021-09-08 New York         36                  2310837    54088
#> 5 2021-09-08 Illinois         17                  1558768    26769
#> 6 2021-09-08 Georgia          13                  1436172    22639
#> 7 2021-09-08 Pennsylvania      42                  1329111    28446
#> 8 2021-09-08 North Carolina    37                  1269208    14925
#> 9 2021-09-08 Ohio              39                  1268841    21020
#> 10 2021-09-08 New Jersey       34                  1108291    27007
#> # ... with more rows
```

# Exploring COVID-19 Data: New York Cases

```
covid_us_states() %>%
  filter(state_name == "New York",
        EXTRACT(DAYOFWEEK %FROM% date) == 1) %>%
  arrange(date) %>%
  mutate(weekly_cases = confirmed_cases - lag(confirmed_cases)) %>%
  ggplot(aes(date, weekly_cases)) +
  geom_col()
```



# Creating a package

# BigQuery Database: Stack Overflow

The screenshot shows the BigQuery web interface. On the left, the 'Explorer' sidebar lists pinned projects, including 'sdo...', 'stackoverflow' (expanded), and various Stack Overflow-related datasets like 'badges', 'comments', etc. The main area is titled 'tags' and displays a preview of the dataset with columns: Row, id, tag\_name, count, excerpt\_post\_id, and wiki\_post\_id. The data shows 13 rows of tag information. At the bottom, pagination controls allow for navigating through 61,059 rows.

Row	id	tag_name	count	excerpt_post_id	wiki_post_id
1	1862	msg	256	18181214	18181213
2	9891	wsadmin	256	7321606	7321605
3	19069	class-hierarchy	256	26495981	26495980
4	22543	inline-editing	256	16039296	16039295
5	33674	sse2	256	5219912	5219911
6	37122	spatial-index	256	11083398	11083397
7	39494	linkify	256	13050345	13050344
8	41067	bitwise-and	256	19254471	19254470
9	57788	subshell	256	41697046	41697045
10	63966	smali	256	11953555	11953554
11	67475	django-taggit	256	6688244	6688243
12	69718	pyephem	256	7912529	7912528
13	79854	docpad	256	10406453	10406452

Learn more: [Stack Overflow on BigQuery](#)

# Example: stackbigquery package

[dgrtwo / stackbigquery](#) Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

dgrtwo README adjustments	Initial commit	7060e07 33 seconds ago	3 commits
R	Initial commit	3 minutes ago	
man	Initial commit	3 minutes ago	
.Rbuildignore	Initial commit	3 minutes ago	
.gitignore	Initial commit	3 minutes ago	
CODE_OF_CONDUCT.md	Initial commit	3 minutes ago	
DESCRIPTION	Initial commit	3 minutes ago	
LICENSE	Initial commit	3 minutes ago	
LICENSE.md	Initial commit	3 minutes ago	
NAMESPACE	Initial commit	3 minutes ago	
README.Rmd	README adjustments	33 seconds ago	
README.md	README adjustments	33 seconds ago	
stackbigquery.Rproj	Initial commit	3 minutes ago	

README.md

## stackbigquery

stackbigquery is a package wrapping the Stack Overflow database on Google BigQuery.

This is a minimal example of using [dbcooper](#) to create a database package:

- Create a connection in [connections.R](#)
- Run `dbcooper::dbc_init()` on that connection in [zzz.R](#)
- Put package-specific functions in other files like [summarize.R](#)

About

Database-specific package for the Stack Overflow data on Google BigQuery

Readme View license

Releases

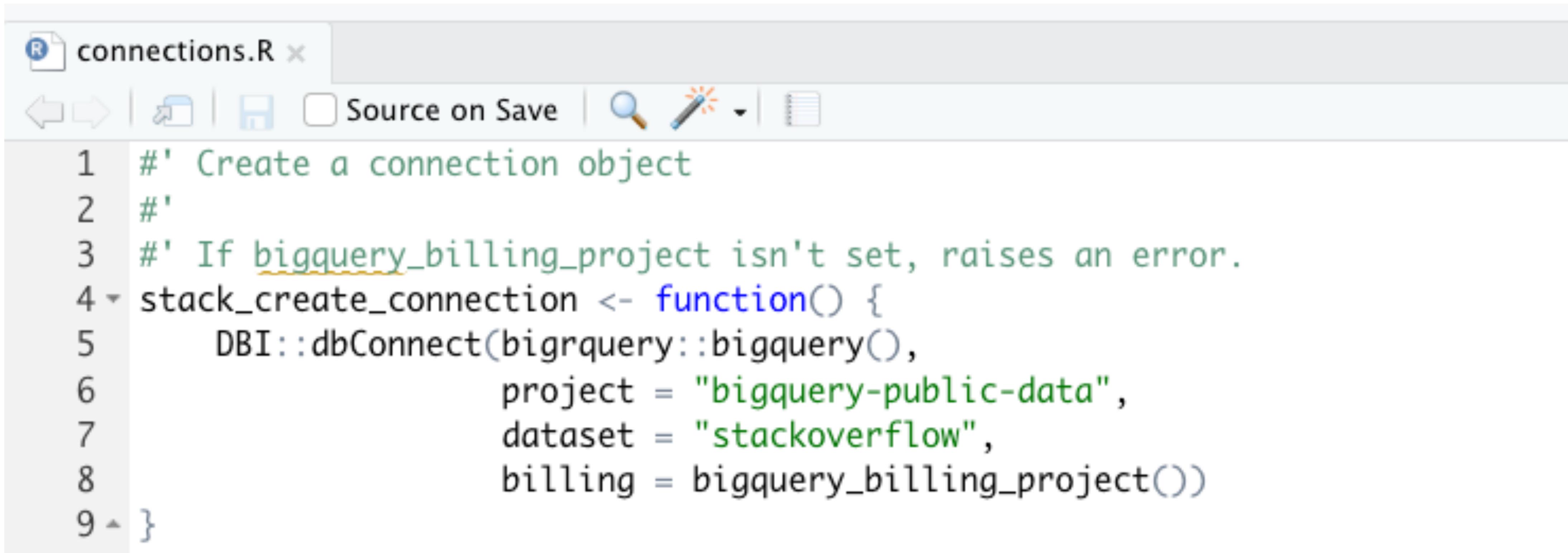
No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

[Link](#)

# 1. Write database-specific connection code

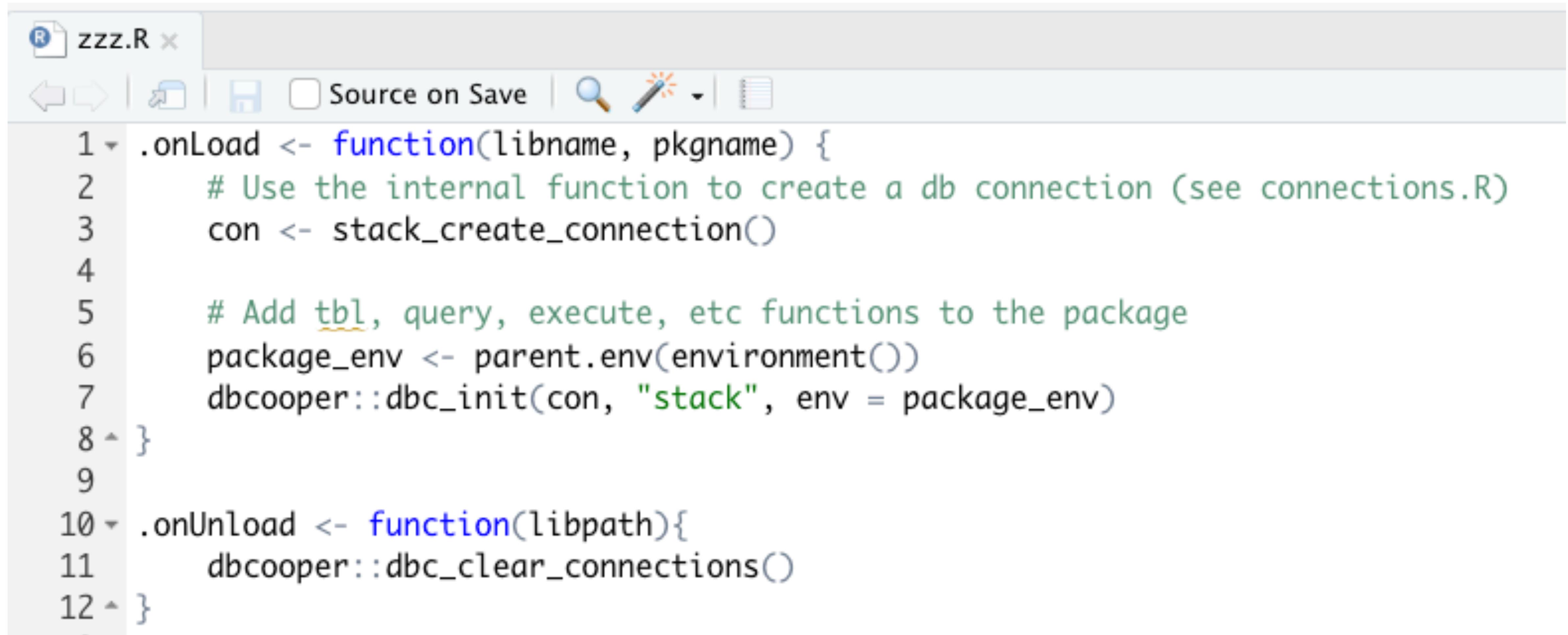


The screenshot shows an RStudio interface with a file named "connections.R" open. The code in the editor is as follows:

```
1 #' Create a connection object
2 #
3 #' If bigrquery_billing_project isn't set, raises an error.
4 stack_create_connection <- function() {
5   DBI::dbConnect(bigrquery::bigquery(),
6                 project = "bigquery-public-data",
7                 dataset = "stackoverflow",
8                 billing = bigrquery_billing_project())
9 }
```

The code defines a function `stack\_create\_connection` that uses the `DBI` package to connect to a BigQuery dataset. It specifies the project as "bigquery-public-data", the dataset as "stackoverflow", and the billing configuration using the `bigrquery\_billing\_project()` function.

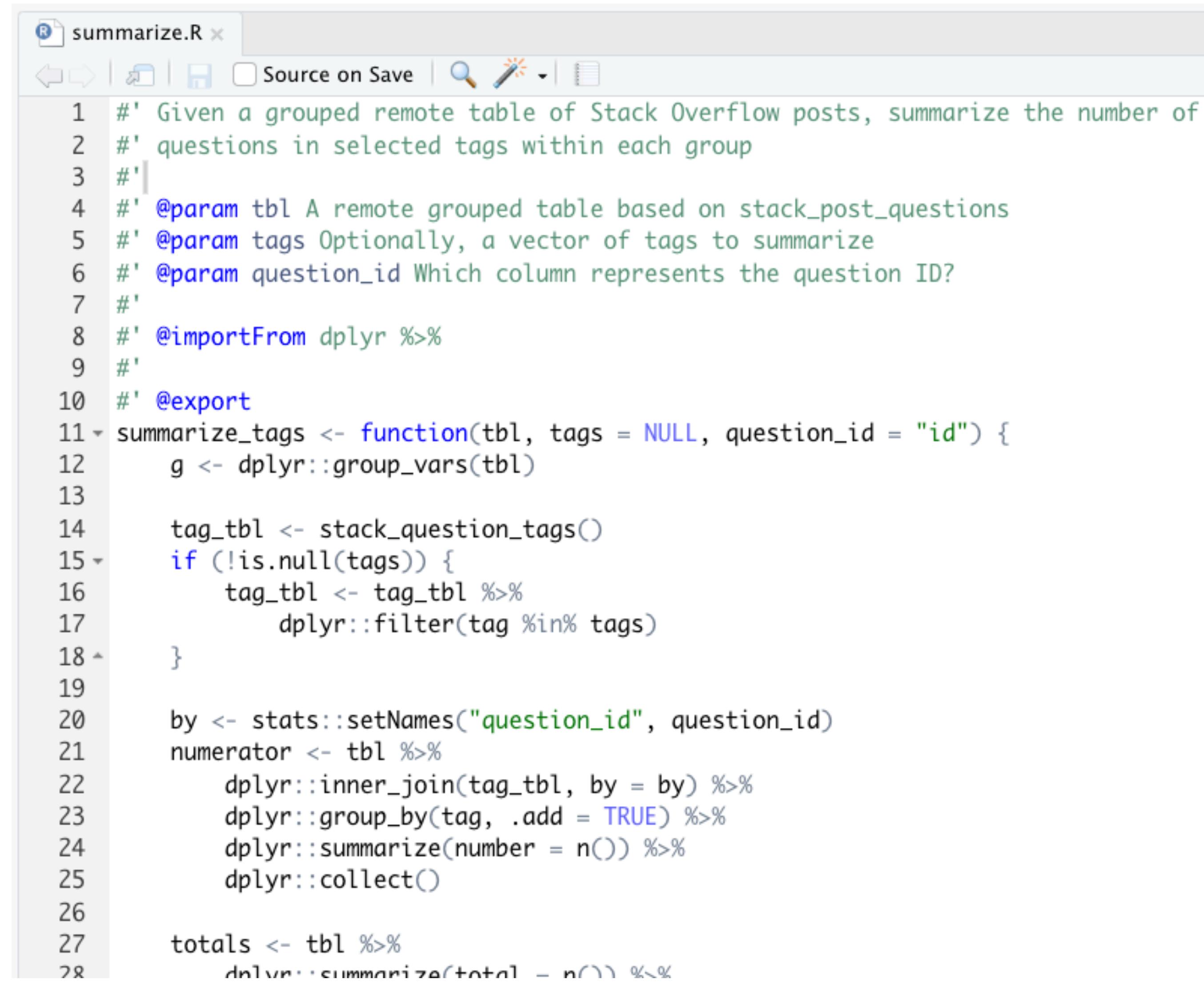
## 2. Add code to zzz.R to run dbc\_init()



The screenshot shows the RStudio interface with the zzz.R file open. The code in the editor is as follows:

```
1 .onLoad <- function(libname, pkgname) {  
2     # Use the internal function to create a db connection (see connections.R)  
3     con <- stack_create_connection()  
4  
5     # Add tbl, query, execute, etc functions to the package  
6     package_env <- parent.env(environment())  
7     dbcooper::dbc_init(con, "stack", env = package_env)  
8 }  
9  
10 .onUnload <- function(libpath){  
11     dbcooper::dbc_clear_connections()  
12 }
```

# 3. Add any database-specific verbs



The screenshot shows an RStudio interface with a file named "summarize.R" open. The code is a function named "summarize\_tags" that takes a grouped remote table "tbl", a vector of tags "tags", and a column name "question\_id". It uses dplyr functions to filter the tags, inner join with a stack\_question\_tags table, group by tag, and then summarize the count of posts. Finally, it collects the results and adds a total row.

```
R summarize.R x
Source on Save | 🔎 | 🖊 | 📄
1 #' Given a grouped remote table of Stack Overflow posts, summarize the number of
2 #' questions in selected tags within each group
3 #
4 #' @param tbl A remote grouped table based on stack_post_questions
5 #' @param tags Optionally, a vector of tags to summarize
6 #' @param question_id Which column represents the question ID?
7 #
8 #' @importFrom dplyr %>%
9 #
10 #' @export
11 summarize_tags <- function(tbl, tags = NULL, question_id = "id") {
12   g <- dplyr::group_vars(tbl)
13
14   tag_tbl <- stack_question_tags()
15   if (!is.null(tags)) {
16     tag_tbl <- tag_tbl %>%
17       dplyr::filter(tag %in% tags)
18   }
19
20   by <- stats::setNames("question_id", question_id)
21   numerator <- tbl %>%
22     dplyr::inner_join(tag_tbl, by = by) %>%
23     dplyr::group_by(tag, .add = TRUE) %>%
24     dplyr::summarize(number = n()) %>%
25     dplyr::collect()
26
27   totals <- tbl %>%
28     dplyr::summarize(total = n()) %>%
```

# And you have a package!

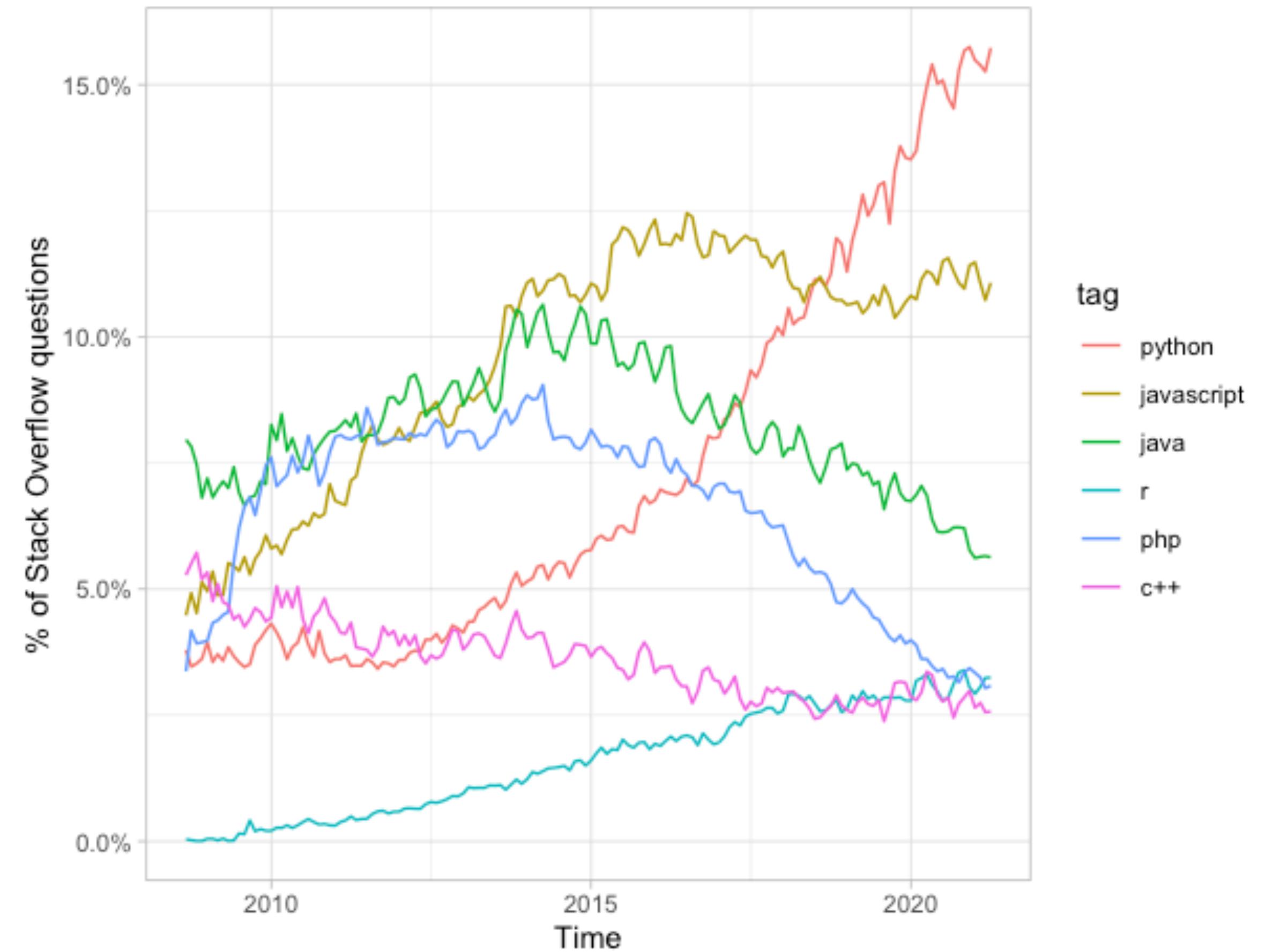
```
library(stackbigquery)
stack_posts_questions()
#> # Source: SQL [?? x 20]
#> # Database: BigQueryConnection
#>   id title          body accepted_answer... answer_count comment_count
#>   <int> <chr>        <chr>           <int>       <int>       <int>
#> 1 3.58e7 Camera with... "<p>current..." NA          2          0
#> 2 3.59e7 Configure a... "<p>How can..." NA          2          0
#> 3 3.59e7 How does Ex... "<p>I notice..." NA          1          0
#> 4 3.59e7 CXF with OD... "<p>Could a..." NA          1          0
#> 5 3.59e7 Why does ch... "<p>So I'm ..." NA          1          0
#> 6 3.59e7 Algolia sea... "<p>I'm usi..." NA          1          0
#> 7 3.60e7 How to use ... "<p>We are ..." NA          4          0
#> 8 3.60e7 How to prin... "<p>Im work..." NA          1          0
#> 9 3.60e7 Can I write... "<p>I'm try..." 35978083      1          0
#> 10 3.60e7 XQuery join... "<p>I have ..." 35988035      2          0
#> # ... with more rows, and 14 more variables: community_owned_date <dttm>,
#> # creation_date <dttm>, favorite_count <int>, last_activity_date <dttm>,
#> # last_edit_date <dttm>, last_editor_display_name <chr>,
#> # last_editor_user_id <int>, owner_display_name <chr>, owner_user_id <int>,
#> # parent_id <chr>, post_type_id <int>, score <int>, tags <chr>,
#> # view_count <int>
```

# Exploring Stack Overflow: Tags over Time

```
library(tidyverse)
library(stackbigquery)

by_month_tag <- stack_posts_questions() %>%
  group_by(month = DATE_TRUNC(DATE(creation_date), MONTH)) %>%
  summarize_tags(c("javascript", "java", "python",
                  "php", "c++", "r"))

by_month_tag %>%
  filter(month != max(month),
        month != min(month)) %>%
  arrange(month) %>%
  mutate(tag = fct_reorder(tag, -percent, last)) %>%
  ggplot(aes(month, percent, color = tag)) +
  geom_line() +
  scale_y_continuous(labels = scales::percent_format()) +
  expand_limits(y = 0) +
  labs(x = "Time",
       y = "% of Stack Overflow questions")
```



- Don't underestimate the value of small usability improvements that affect you every day
- A good database package lets you forget about the package completely

examples.R x

```

1 library(tidyverse)
2 library(dbcooper)
3 theme_set(theme_light())
4
5 # Create the connection the same way you always would
6 con <- DBI::dbConnect(bigrquery::bigrquery(),
7                         project = "bigquery-public-data",
8                         dataset = "covid19_nyt",
9                         billing = Sys.getenv("BIGQUERY_BILLING_PROJECT"))
10
11 # Create functions with prefix covid_
12 dbc_init(con, "covid")
13
14 covid_us_states() %>%
15   filter(state_name == "New York",
16         EXTRACT(DAYOFWEEK %FROM% date) == 1) %>%
17   arrange(date) %>%
18   mutate(weekly_cases = confirmed_cases - lag(confirmed_cases)) %>%
19   ggplot(aes(date, weekly_cases)) +
20   geom_col()

```

22:1 (Top Level) R Script

Console Terminal Find in Files Jobs

R 4.1.0 · ~/Repositories/rpackages/stackbigquery/ ↗

```

+ project = "bigquery-public-data",
+ dataset = "covid19_nyt",
+ billing = Sys.getenv("BIGQUERY_BILLING_PROJECT")
> # Create functions with prefix covid_
> dbc_init(con, "covid")
> covid_us_states() %>%
+   filter(state_name == "New York",
+         EXTRACT(DAYOFWEEK %FROM% date) == 1) %>%
+   arrange(date) %>%
+   mutate(weekly_cases = confirmed_cases - lag(confirmed_cases)) %>%
+   ggplot(aes(date, weekly_cases)) +
+   geom_col()
Complete
Billed: 0 B
Downloading first chunk of data.
First chunk includes all requested rows.
Warning message:
Removed 1 rows containing missing values (position_stack).
>

```

Environment History Connections Build Git

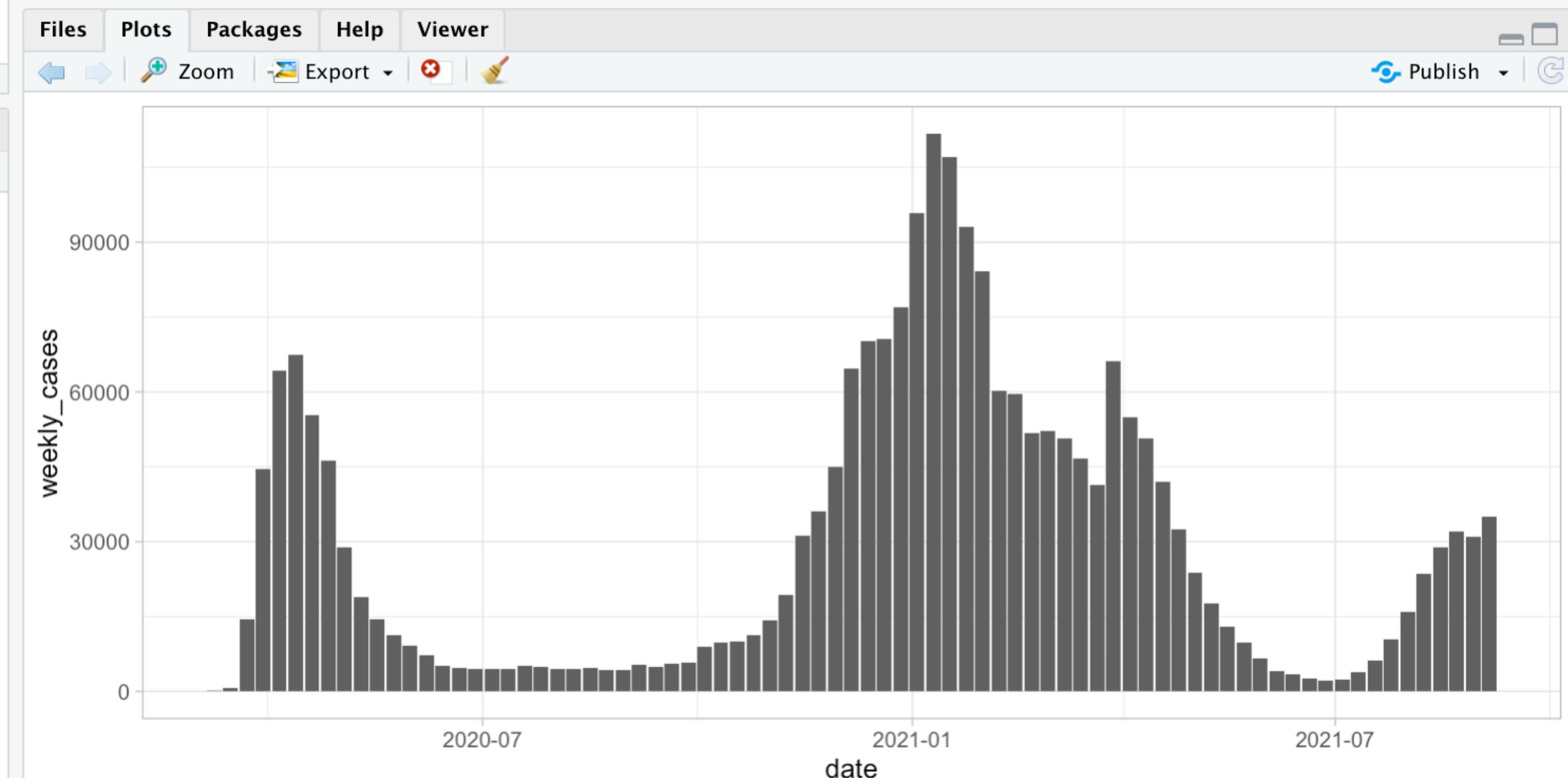
Import Dataset 219 MiB

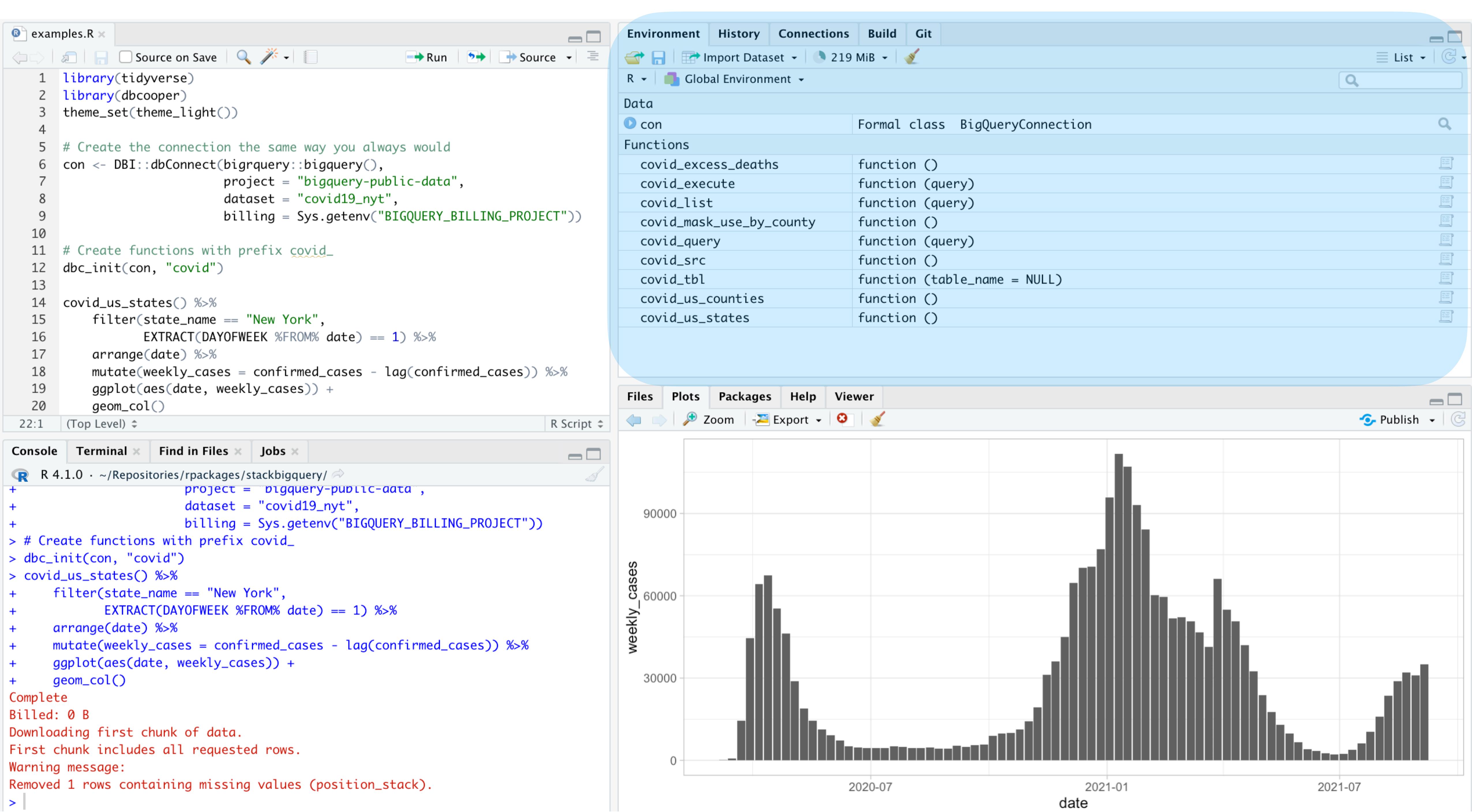
R Global Environment

con Formal class BigQueryConnection

Functions

covid_excess_deaths	function ()
covid_execute	function (query)
covid_list	function (query)
covid_mask_use_by_county	function ()
covid_query	function (query)
covid_src	function ()
covid_tbl	function (table_name = NULL)
covid_us_counties	function ()
covid_us_states	function ()





# Thank you

- **Lander Analytics**

- Jared Lander

- Nicole DelGiudice

- **Collaborators**

- Chris Cardillo

- Ramnath Vaidyanathan

@drob

[www.varianceexplained.org](http://www.varianceexplained.org)

VARIANCE EXPLAINED

ABOUT ME    POSTS    TEXT MINING IN R    INTRODUCTION TO EMPIRICAL BAYES



David Robinson

Principal Data Scientist at  
Heap, works in R and  
Python.

- [✉ Email](#)
- [🐦 Twitter](#)
- [🐙 Github](#)
- [.Stack Overflow](#)

Subscribe

Your email

[Subscribe to this blog](#)

Recommended

This is the homepage and blog of David Robinson. For more about me, [see here](#).

## Recent Posts

### **Machine learning in a hurry: what I've learned from the SLICED ML competition** July 21, 2021

Over this summer I've competed in the SLICED ML competition, where contestants have two hours to create a Kaggle submission. I share what I learned about competitive machine learning and R.

### **The 'circular random walk' puzzle: tidy simulation of stochastic processes in R**

Solving a puzzle from 538's The Riddler column: if you pass cranberry sauce randomly around a table of 20, who is most likely to be the last person to get it?

### **The 'prisoner coin flipping' puzzle: tidy simulation in R**

Solving a puzzle from 538's the Riddler column: if N prisoners have a choice to flip a coin, and go free as long as one coin is flipped and all coins are heads, what strategy should they take to maximize their chances? Another demonstration of probabilistic reasoning and tidy simulation.

### **The 'spam comments' puzzle: tidy simulation of stochastic processes in R**

Solving a puzzle from 538's The Riddler column: if new spam comments appear at an average rate of one per day, including on other spam comments, how many can we expect after three days?

### **Feller's coin-tossing puzzle: tidy simulation in R**

If you toss n coins, what's the probability there are no streaks of k heads?

May 04, 2020

April 13, 2020

January 17, 2020