Tidy R programming with the OMOP common data model

Edward Burn, Adam Black, Marti Catala, Berta Raventós

2022-10-26T00:00:00+01:00

Table of contents

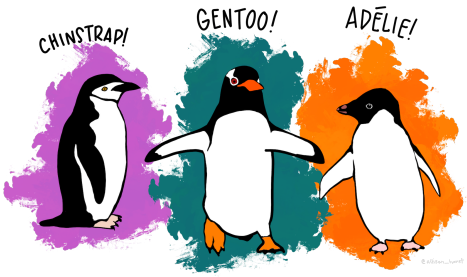
Preface

This book is written for analysts writing analytic code with R to run against the OMOP CDM. This source code for the book can be found at this Github repository Please open an issue there if you have a question or suggestion. Pull requests with suggested changes and additions are also most welcome.

1 Getting started with R

1.1 Installing R and R Studio

1.2 A first data analysis



Artwork by @allison_horst

For a quick example of a data analysis with R, let´s use the data from palmerpenguins package (https://allisonhorst.github.io/palmerpenguins/), which contains data on penguins collected from the Palmer Station in Antarctica.

Because we´ll be using a few packages not included in base R, first we need to install these if we don´t already have them.

```
install.packages("dplyr")
install.packages("ggplot2")
install.packages("palmerpenguins")
```

Once installed, we can load them like so.

```
library(dplyr)
library(ggplot2)
library(palmerpenguins)
```

We can get an overview of the data using the `glimpse()` command.

```
glimpse(penguins)
```

```
Rows: 344
Columns: 8
$ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
$ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
$ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
$ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
$ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
$ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
$ sex               <fct> male, female, female, NA, female, male, female, male~
$ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

Let´s get a count by species

```
penguins %>%
  group_by(species) %>%
  count()
```

```
# A tibble: 3 x 2
# Groups:   species [3]
  species       n
  <fct>     <int>
1 Adelie      152
2 Chinstrap    68
3 Gentoo      124
```

Now suppose we are particularly interested in the body mass variable. We can first notice that there are a couple of missing records for this.

```
penguins %>%
  group_by(species) %>%
  summarise(not_missing_body_mass_g=sum(!is.na(body_mass_g)==TRUE),
            missing_body_mass_g=sum(is.na(body_mass_g)==TRUE))
```

```
# A tibble: 3 x 3
  species   not_missing_body_mass_g missing_body_mass_g
  <fct>                       <int>               <int>
1 Adelie                        151                   1
2 Chinstrap                      68                   0
3 Gentoo                        123                   1
```

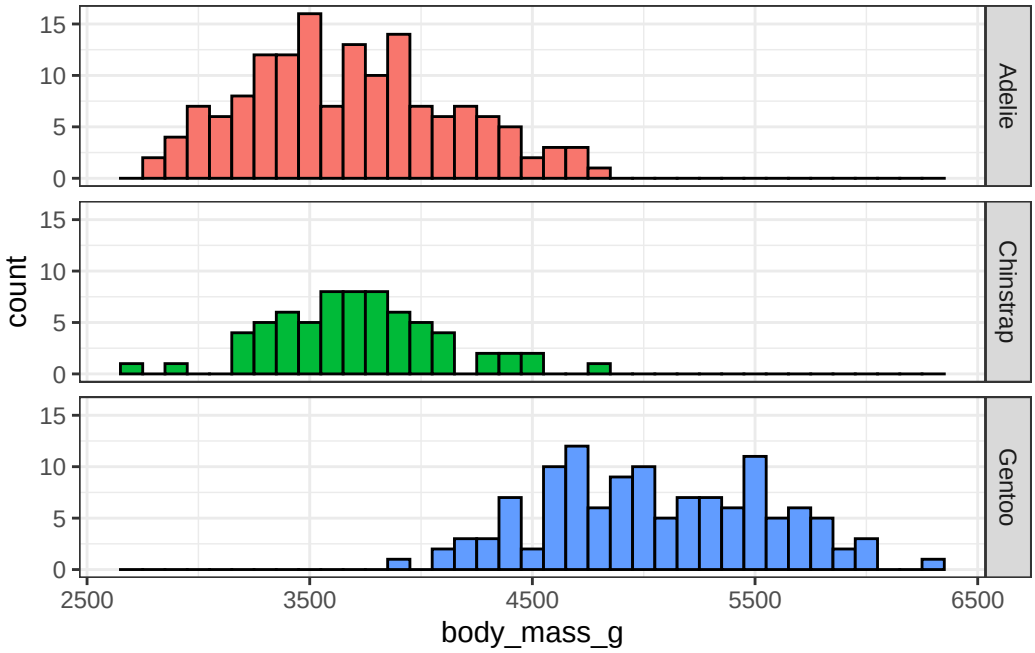
We can get the mean for each of the species (dropping those two missing records).

```
penguins %>%
  group_by(species) %>%
  summarise(mean_body_mass_g=round(mean(body_mass_g, na.rm=TRUE)))
```

```
# A tibble: 3 x 2
  species   mean_body_mass_g
  <fct>                <dbl>
1 Adelie                3701
2 Chinstrap             3733
3 Gentoo                5076
```

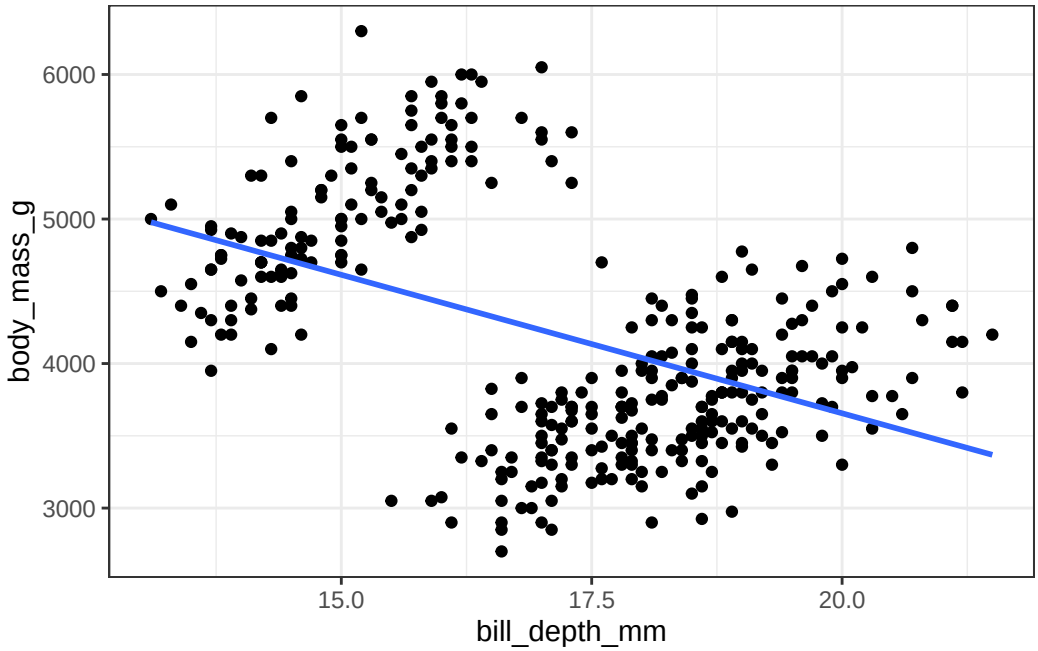
We can then also do a histogram for each of the species.

```
penguins %>%
  ggplot(aes(group=species, fill=species))+
  facet_grid(species~ .) +
  geom_histogram(aes(body_mass_g), colour="black", binwidth = 100)+
  theme_bw()+
  theme(legend.position = "none")
```



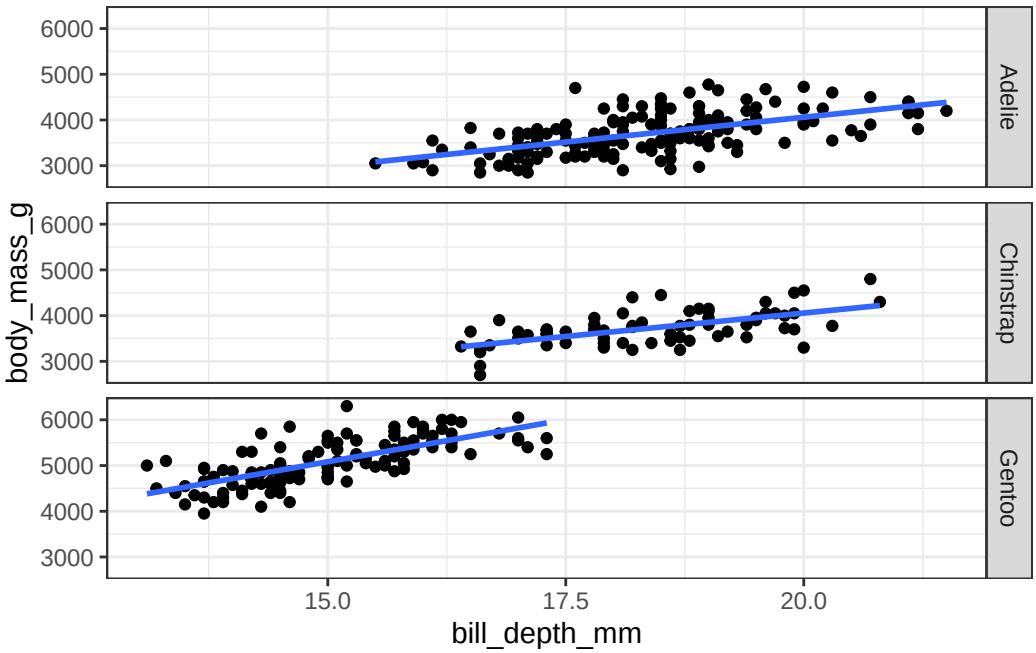
How about the relationship between body mass and bill depth?

```
penguins %>%
  ggplot(aes(x=bill_depth_mm,y=body_mass_g))+
  geom_point()+
  geom_smooth(method="lm",se=FALSE )+
  theme_bw()+
  theme(legend.position = "none")
```



But what about by species?

```
penguins %>%
  ggplot(aes(x=bill_depth_mm,y=body_mass_g))+
  facet_grid(species~ .) +
  geom_point()+
  geom_smooth(method="lm",se=FALSE )+
  theme_bw()+
  theme(legend.position = "none")
```



Oh, your first data analysis and you have already found an example of Simpson´s paradox!

2 Creating a reference to the common data model

2.0.1 Connecting to a database from R using DBI

Database connections from R can be made using the DBI package. The back-end for `DBI` is facilitate by database specific driver packages, with applications then using the front-end API. As an example, lets say we want to work with a local duckdb from R. In this case the we can use the duckdb R package as the driver. In this case we can also create the database in-memory

```
library(DBI)
db<-dbConnect(duckdb::duckdb(), dbdir=":memory:")
```

If we instead wanted to connect to other database management systems, these connections could look like

```
# Postgres
db <- DBI::dbConnect(RPostgres::Postgres(),
                     dbname = Sys.getenv("CDM5_POSTGRESQL_DBNAME"),
                     host = Sys.getenv("CDM5_POSTGRESQL_HOST"),
                     user = Sys.getenv("CDM5_POSTGRESQL_USER"),
                     password = Sys.getenv("CDM5_POSTGRESQL_PASSWORD"))
# Redshift (almost identical to Postgres)
db <- DBI::dbConnect(RPostgres::Redshift(),
                     dbname   = Sys.getenv("CDM5_REDSHIFT_DBNAME"),
                     host     = Sys.getenv("CDM5_REDSHIFT_HOST"),
                     port     = Sys.getenv("CDM5_REDSHIFT_PORT"),
                     user     = Sys.getenv("CDM5_REDSHIFT_USER"),
                     password = Sys.getenv("CDM5_REDSHIFT_PASSWORD"))
# SQL Server
db <- DBI::dbConnect(odbc::odbc(),
                     Driver   = "ODBC Driver 18 for SQL Server",
                     Server   = Sys.getenv("CDM5_SQL_SERVER_SERVER"),
                     Database = Sys.getenv("CDM5_SQL_SERVER_CDM_DATABASE"),
```

```
                        UID      = Sys.getenv("CDM5_SQL_SERVER_USER"),
                        PWD      = Sys.getenv("CDM5_SQL_SERVER_PASSWORD"),
                        TrustServerCertificate="yes",
                        Port     = 1433)
```

2.0.2 Creating a reference to the OMOP common data model

If we have connected to a database which contains data mapped to the format of the OMOP common data model the CDMConnector provides functionality to simplify our work with a database. Because we already know the structure of the common data model, CDMConnector can be used to create a reference to the various tables that are used.

```
library(CDMConnector)
```

```
db <- DBI::dbConnect(duckdb::duckdb(),
                     dbdir = CDMConnector::eunomia_dir())
cdm <- CDMConnector::cdm_from_con(db,
                                  cdm_schema = "main")
cdm
```

```
# OMOP CDM reference (tbl_duckdb_connection)

Tables: person, observation_period, visit_occurrence, visit_detail, condition_occurrence, dru
```

From this reference we we can read the tables with "$" operator or [[" "]].

```
cdm$observation_period
```

```
# Source:   table<main.observation_period> [?? x 5]
# Database: DuckDB 0.5.0 [eburn@Windows 10 x64:R 4.2.1/C:\Users\eburn\AppData\Local\Temp\Rtmp
   observation_period_id person_id observation_period_start~1 observat~2 perio~3
                   <dbl>     <dbl> <date>                     <date>       <dbl>
 1                     6         6 1963-12-31                 2007-02-06  4.48e7
 2                    13        13 2009-04-26                 2019-04-14  4.48e7
 3                    27        27 2002-01-30                 2018-11-21  4.48e7
 4                    16        16 1971-10-14                 2017-11-02  4.48e7
 5                    55        55 2009-05-30                 2019-03-23  4.48e7
 6                    60        60 1990-11-21                 2019-01-23  4.48e7
 7                    42        42 1909-11-03                 2019-03-13  4.48e7
```

```
 8                   33        33 1986-05-12                 2018-09-10  4.48e7
 9                   18        18 1965-11-17                 2018-11-07  4.48e7
10                   25        25 2007-03-18                 2019-04-07  4.48e7
# ... with more rows, and abbreviated variable names
#   1: observation_period_start_date, 2: observation_period_end_date,
#   3: period_type_concept_id
# i Use `print(n = ...)` to see more rows
```

```
cdm[["observation_period"]]
```

```
# Source:   table<main.observation_period> [?? x 5]
# Database: DuckDB 0.5.0 [eburn@Windows 10 x64:R 4.2.1/C:\Users\eburn\AppData\Local\Temp\Rtmp
   observation_period_id person_id observation_period_start~1 observat~2 perio~3
                   <dbl>     <dbl> <date>                     <date>       <dbl>
 1                     6         6 1963-12-31                 2007-02-06  4.48e7
 2                    13        13 2009-04-26                 2019-04-14  4.48e7
 3                    27        27 2002-01-30                 2018-11-21  4.48e7
 4                    16        16 1971-10-14                 2017-11-02  4.48e7
 5                    55        55 2009-05-30                 2019-03-23  4.48e7
 6                    60        60 1990-11-21                 2019-01-23  4.48e7
 7                    42        42 1909-11-03                 2019-03-13  4.48e7
 8                    33        33 1986-05-12                 2018-09-10  4.48e7
 9                    18        18 1965-11-17                 2018-11-07  4.48e7
10                    25        25 2007-03-18                 2019-04-07  4.48e7
# ... with more rows, and abbreviated variable names
#   1: observation_period_start_date, 2: observation_period_end_date,
#   3: period_type_concept_id
# i Use `print(n = ...)` to see more rows
```

When we create our cdm reference we could have also specified the tables we want to read:

```
cdm <- CDMConnector::cdm_from_con(db,
                                  cdm_tables = c("person","observation_period"))
cdm
```

```
# OMOP CDM reference (tbl_duckdb_connection)

Tables: person, observation_period
```

Moreover, we can also specify the writable schema and the tables that we are interested on it. For example, if we wanted to create a reference to the person and observation period tables in

the common data model along with exposure and outcome cohort tables in a schema we have write access to, we could do this like so:

```
cdm <- CDMConnector::cdm_from_con(db,
  cdm_schema = "main",
  cdm_tables = c("person","observation_period"),
  write_schema = "results",
  cohort_tables = c("exposure_cohort", "outcome_cohort"))
```

3 Exploring the CDM

3.0.1 tally()

3.0.2 distinct()

3.0.3 rename()

3.0.4 group_by()

3.0.5 summarise()

4 filter(), select(), mutate()

4.0.1 if_else()

4.0.2 paste0 / glue

4.0.3 working with strings

5

strings

6

Dates

7 right_join(), left_join(), inner_join(), and anti_join()

7.0.1 and union() and union_all()

8 Getting to tidy data

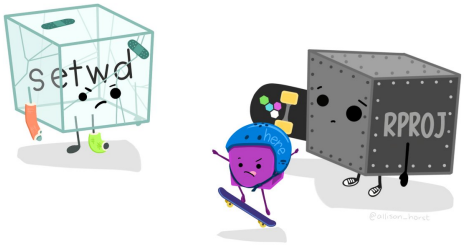
8.0.1 compute()

8.0.2 collect()

8.0.3 pull()

9 Analysis in R

10 Organising data analyses with projects and renv



Artwork by @allison_horst

References

11

Learning R