Statistical Computing with R
Masters in Data Science 503 (S10)
Fourth Batch, SMS, TU, 2025

Shital Bhandary
Associate Professor
Statistics/Bio-statistics, Demography and Public Health Informatics
Patan Academy of Health Sciences, Lalitpur, Nepal
Faculty, Masters in Medical Research, NHRC/Kathmandu University
Faculty, FAIMER Fellowship in Health Professions Education, India/USA

Review Preview (Unit 2, Session 4)

- Databases in R
  - Database interface (DBI) and dplyr for database
  - dbplyr as dplyr backend for database
- Big data in R
  - Subsampling
  - ff, ffbase, ffbase2 packages
  - data.table package
  - RevoScaleR & RHadoop packages
  - Microsoft Open R and R Server
  - SparkR and sparklyr packages
  - R for Azure SQL and SQL server

Accessing SQL database in R: Three approaches
https://www.youtube.com/watch?v=Z5LPjh_EkJk

- Database Interface (DBI) and Backend packages
- dplyr and dbplyr packages
- R markdown/notebooks

Database: DBMS

- In-process DBMS: SQLite, duckdb etc.
- Client server DBMS: PostgreSQL, MariaDB, SQL server, Oracle etc.
- Cloud DBMS: Snowflake, Redshift, BigQuery etc.

Connecting to a database:

- To connect to the database from R, you'll use a pair of packages:
- You'll always use DBI package because it provides a set of generic functions that connect to the database, upload data, run SQL queries, etc.
- You'll also use a package tailored for the DBMS you're connecting to.
- This package translates the generic DBI commands into the specifics needed for a given DBMS.
- There's usually one package for each DBMS, e.g. RPostgres for PostgreSQL and RMariaDB for MySQL.

Note:

- If you can’t find a specific package for your DBMS, you can usually use the odbc package instead.
- This uses the ODBC protocol supported by many DBMS.
- odbc requires a little more setup because you’ll also need to install an ODBC driver and tell the odbc package where to find it.

Example: Client server DBMD

```
library(DBI)
library(dbplyr)
library(tidyverse)

con <- DBI::dbConnect(
  RMariaDB::MariaDB(),
  username = "foo"
)
```

```
library(DBI)
library(dbplyr)
library(tidyverse)

con <- DBI::dbConnect(
  RPostgres::Postgres(),
  hostname =
"databases.mycompany.com",
  port = 1234
)
```

Example: In-process DBMS (RSQLite with DBI) https://rsqlite.r-dbi.org/

- install.packages("RSQLite")
- library(RSQLite)
- library(DBI)
- # Create an ephemeral in-memory RSQLite database
- con <- **dbConnect**(RSQLite::SQLite(), ":memory:")
- dbListTables(con)

character(0) # 0 tables

- dbWriteTable(con, "mtcars", mtcars)
- dbListTables(con)
- dbListFields(con, "mtcars")
- dbReadTable(con, "mtcars")
- res <- dbSendQuery(con, "SELECT * FROM mtcars WHERE cyl = 4")
- dbFetch(res)
- dbClearResult(res)
- **dbDisconnect**(con)

It is easy to use "dbplyr" to work with all types of dataset now! Chapter 21, R4DS, 2nd Edition.
https://r4ds.hadley.nz/databases

dplyr and DBI: What happens?

- con <- DBI::dbConnect(duckdb::duckdb())
- dbWriteTable(con, "mpg", ggplot2::mpg)
- dbWriteTable(con, "diamonds", ggplot2::diamonds)
- dbListTables(con)

What happens?

```
con %>%
        dbReadTable("diamonds")
%>%     as_tibble()
```

What happens?

```
sql <- "SELECT carat, cut, clarity,
color, price FROM diamonds WHERE
price > 15000"
as_tibble(dbGetQuery(con,sql))
```

dbplyr as dplyr backend:

- Now that we've connected to a database and loaded up some data, we can start to learn about dbplyr.
- dbplyr is a dplyr **backend**, which means that you keep writing dplyr code but the backend executes it differently.
- In this, dbplyr translates to SQL

- To use dbplyr, you must first use tbl() to create an object that represents a database table

- library(dbplyr)
- diamonds_db <- tbl(con, "diamonds")
- diamonds_db

Notes: 1

- There are two other common ways to interact with a database.
- First, many corporate databases are very large so you need some hierarchy to keep all the tables organized.
- In that case you might need to supply a schema, or a catalog and a schema, in order to pick the table you're interested in:

- diamonds_db <- tbl(con, in_schema("sales", "diamonds"))

- diamonds_db <- tbl(con, in_catalog("north_america", "sales", "diamonds"))

Notes: 2

- Other times you might want to use your own SQL query as a starting point:
- diamonds_db <- tbl(con, sql("SELECT * FROM diamonds"))
- This object is **lazy**; When you use dplyr verbs on it, dplyr doesn’t do any work:
- It just records the sequence of operations that you want to perform and only performs them when needed.
- big_diamonds_db <- diamonds_db %>% filter(price > 15000) %>% select(carat:clarity, price)
- big_diamonds_db

You can see the SQL code used by dbplyr using show_query()

- big_diamonds_db %>%
- show_query()

- <SQL>
- SELECT carat, cut, color, clarity, price
- FROM diamonds
- WHERE (price > 15000.0)

Getting data back into R using collect():

- To get all the data back into R, you call collect() function
- Behind the scenes, this generates the SQL, calls dbGetQuery() to get the data, then turns the result into a tibble

- big_diamonds <-
  big_diamonds_db %>%
  collect()

What happened?

- big_diamonds

Did you get the 1655 x 5 tibble?

What happens now with “dbplyr”?

- dbplyr::copy_nycflights13(con)
- How many tables were copied?
- flights <- tbl(con, "flights")
- planes <- tbl(con, "planes")
- What is done above?

- flights %>% show_query()
- planes %>% show_query()
- What does it do?

```
flights %>%
        filter(dest == "IAH") %>%
        arrange(dep_delay) %>%
        show_query()
```

What happens now with “dbplyr”?

- planes %>%
  select(tailnum, type, manufacturer, model, year) %>%
  show_query()

- planes %>%
  select(tailnum, type, manufacturer, model, year) %>%
  rename(year_built = year) %>%
  show_query()

- planes %>%
  select(tailnum, type, manufacturer, model, year) %>%
  relocate(manufacturer, model, .before = type) %>%
  show_query()

flights %>%
mutate(speed=distance/(air_time/60))
show_query()

Read/practice more using this link: https://r4ds.hadley.nz/databases

Cloud Database (Google BigQuery package for r): https://bigrquery.r-dbi.org/

- install.packages("bigrquery")
- Low-level API (need account!)
- library(bigrquery)
- billing <- bq_test_project()
- sql <- "SELECT year, month, day, weight_pounds FROM `publicdata.samples.natality`"
- Tables:
- tb <- bq_project_query(billing, sql)
- bq_table_download(tb, n_max = 10)
- You can also use:
  - DBI
  - dplyr and dbplyr

  with BigQuery

Cloud database: Snowflake (RODBC needed)

- https://community.snowflake.com/s/article/How-To-Connect-Snowflake-with-R-RStudio-using-RODBC-driver-on-Windows-MacOS-Linux

Cloud database: Redshift

- https://aws.amazon.com/blogs/big-data/connecting-r-with-amazon-redshift/

Working with large datasets in R

(Book: Beginning Data Science in R – Thomas Mailund)

- The concept of Big Data refers to very large datasets, sets of sizes where you need **data warehouses** to store the data, where you typically need sophisticated algorithms to handle the data, and distributed computations to get anywhere with it.
- **How big is "Big data"?**
- Dealing with Big Data is also part of data science. Working with large datasets and how to deal with data that slows down your analysis is very important knowledge and skill for data scientists.

Big Data/Large dataset: Chapter 5
(Book: Beginning Data Science in R – Thomas Mailund)

- If we ignore the Big Data issue, what a large dataset is, depends very much on what you want to do with the data. That comes down to the complexity of what you are trying to achieve.

- The science of what you can do with data in a given amount of time, or a given amount of space (be it RAM or disk space or whatever you need), is called complexity theory and is one of the fundamental topics in computer science.

Working with large datasets: Chapter 5

- **Subsample** your data before you Analyze the Full Dataset

- You very rarely need to analyze a complete dataset to get at least an idea of how the data behaves.

- Unless you are looking for very rare events, you will get as much a feeling for the data looking at a few thousands of data points as you would from looking at a few million.

Working with large datasets: Chapter 5

- Here it is important that you pick a **random sample**.
- Randomizing might remove a subtle signal, but with the power of statistics, we can deal with **random noise**.
- It is much harder to deal with consistent biases we just don’t know about.
- This is same as taking random sample from (target) population in research!

You can use “dplyr” package for sampling:

- iris %>% sample_n(size = 5) #Select random sample of size 5
- iris %>% sample_frac(size = 0.02) #Select 2% random sample

- You need your data in a form that “dplyr” can manipulate, and if the data is too large even to load into R, then you cannot have it in a data frame to sample from, to begin with.

- Luckily, dplyr has support for using data that is stored on disk rather than in RAM, in various backend formats, too.

- **It is, for example, possible to connect a “database to dplyr” and sample from a large dataset this way.**

Problems working with large datasets in R: Chapter 5

- Running out of memory during analysis
  - R can be very wasteful of RAM because R remembers more (than is immediately obvious) but shows less in the output
  - In R, all objects are immutable (unless we use a workaround)
  - Whenever you modify an object, you are actually creating a new object

Plotting problems with large datasets in R: Use “hexbin” and/or 2D Density plot!

- library(ggplot2)
- library(hexbin)
- Too large to plot (Use Hex plot and/or 2-D density plots for this)
  - d <- data.frame(x = rnorm(10000), y = rnorm(10000))
  - d %>% ggplot(aes(x = x, y = y)) + geom_point() # Large/cluttered scatterplot
  - d %>% ggplot(aes(x = x, y = y)) + geom_hex() # Requires “hexbin” package
  - **d %>% ggplot(aes(x = x, y = y)) + geom_density_2d() # 2D Density plot**
  - d %>% ggplot(aes(x = x, y = y)) + geom_hex() + scale_fill_gradient(low = "lightgray", high = "red") + geom_density2d(color = "black") # Hex with 2-D Density plot

What do these packages do for Big Data? https://malouche.github.io/bigdata2018/largedata.html

- install.packages("ff")
- library(ff)
- install.packages("ffbase")
- library(ffbase)
- install.packages("ffbase2")
- library(ffbase2)

ff package

https://bookdown.org/josephine_lukito/j381m_tutorials/ff.html

- “ff” is a package that helps you work with larger datasets.
- “ff” works by storing your data in disc storage.
- This is done using a **flat file (hence the “ff”)**, which are numeric vectors that point to disc memory.
- “ffbase”, a helper package that allows you to perform simple functions with ff objects.

Example 2:

- library(ff) #ff = flat files
  - ffcars <- as.ffdf(cars)
  - summary(ffcars)
- library(ffbase) #Another set of fast models for big data in R!
  - model <- bigglm(dist ~ speed, data = ffcars)
  - summary(model)
- library(ffbase2)
  - dplyr on ff
  - Available from github: https://github.com/edwindj/ffbase2
  - iris_f <- tbl_ffdf(iris)
  - cars_f <- tbl_ffdf(mtcars, src="./db_ff, name_cars")

Book Example: Bureau of Transformation Statistics (https://www.transtats.bts.gov/homepage.asp)

- flights.ff <- read.table.ffdf(file="flights_sep_oct15.txt", sep=",", VERBOSE=TRUE, header=TRUE, next.rows=100000, colClasses=NA)
  - csv-read=34.24sec ffdf-write=6.303sec TOTAL=40.543sec, 426.4 KB
- flights.table <- read.table("flights_sep_oct15.txt", sep=",", header=TRUE) – done in 32 seconds, data size = 101.9 MB

- read.table.ffdf() of 2013-2014 flights of 2 GB data) – done with 28 files of 516.5 KB size (456 seconds with only 380 MB RAM used)
- **read.table()** of 2013-2014 flights of 2 GB) data – done with 1.3 GB size single file (441 seconds with maximum of 4.85 GB RAM)

Working with Big Data in R: Summary 1!

https://rviews.rstudio.com/2019/07/17/3-big-data-strategies-for-r/

Strategies:

- Strategy 1: Sample and Model (**Sample or slice?**)
- Strategy 2: Chunk and pull (**How to? ff, ffbase, ffbase2?**)
- Strategy 3: Push Compute to Data (**DBI and dplyr database backend!**)
- Webinar Video: https://www.youtube.com/watch?v=ybKkdEuxxN8

Working with Big Data in R: Summary from Big Data Analytics with R Book

- Unleashing the power of R from within:
  - Use apply() family of functions instead of loop and pipes for data.frames
  - Use R package such as ff, ffbase, ffbase2 and bigmemory packages
  - Apply statistical methods to large R objects through biglm, bigglm and ffbase packages

Working with Big Data in R: Summary from Big Data Analytics with R Book

- Unleashing the power of R from within:
  - Enhance the speed of data processing with R libraries supporting parallel computing e.g. **parallel** (doParellel backend and forreach package) and **boot** packages for the parallel computing and bootstrapping in R
  - Benefit from faster data manipulation methods available in the **data.table** package

Data.table package: https://okanbulut.github.io/bigdata/wrangling-big-data.html

- What is data.table?
- Why use data.table over tidyverse?
- Reading/writing data with data.table
- Using the i, j and by in data.table

Use “data.table” package in R:

- A frequent criticism of R is its inefficiency in handling large datasets. That's where the R package data.table enters the scene. If your datasets have more than tens of thousands of rows, the data.table package is a must. (https://psrc.github.io/r-data-table/)

- The data.table is an alternative to R’s default data.frame to handle tabular data. The reason it’s so popular is because of the speed of execution on larger data and the **terse** syntax.

Use “data.table” package in R:

- **So, effectively you type less code and get much faster speed. It is one of the most downloaded packages in R and is preferred by Data Scientists.**

- It is probably one of the best things that have happened to R programming language as far as speed is concerned

- Visit: https://www.machinelearningplus.com/data-manipulation/datatable-in-r-complete-guide/ for full tutorial on data.table package and its syntax!

RevoScaleR package:

https://learn.microsoft.com/en-us/machine-learning-server/r-reference/revoscaler/revoscaler

- The **RevoScaleR** library is a collection of **portable, scalable, and distributable R functions** for importing, transforming, and analyzing data at scale.

- Functions run on the **RevoScaleR** interpreter, built on open-source R, **engineered to leverage the multithreaded and multinode architecture of the host platform**.

- You can use it for descriptive statistics, generalized linear models, k-means clustering, logistic regression, classification and regression trees, and decision forests.

RevoScaleR is open sourced by Microsoft on June 2021: https://en.wikipedia.org/wiki/RevoScaleR

RevoScaleR: After Microsoft from 2015!

https://learn.microsoft.com/en-us/machine-learning-server/r-reference/revoscaler/revoscaler

- The **RevoScaleR** library is found in Machine Learning Server and Microsoft R products.

- You can use any R IDE to write R script calling functions in **RevoScaleR**, but the script must run on a computer having the interpreter and libraries.

- **RevoScaleR** is often preloaded into tools that integrate with Machine Learning Server and R Client.

Microsoft R Open with R Studio in Windows 10 PC: https://www.youtube.com/watch?v=fZXjPYd2Q4Y

Hadoop and MapReduce frameworks in R: You will need to know Linux commands!

- RHadoop package: It supports MapReduce, HDFS and Hbase database management directly from the console of R language (**R Studio server is better!**) [rmr2 and rhdfs of Rhadoop are used for MapReduce files!]

- The packages have been developed by **Revolution Analytics**, but due to the acquisition of Revolution Analytics by Microsoft, the latter has recently become the lead maintainer of the packages.

- All five R packages (rhdfs, rhbase, plyrmr, rmr2 and ravro) of RHadoop, their binary files, documentation, and tutorials, are available at a GitHub repository at https://github.com/RevolutionAnalytics/RHadoop/wiki

Revolution R Enterprise, R with sparkR and sparklyr package for R Studio and Azure/SQL:

- Revolution/Microsoft R Enterprise adds proprietary components e.g. scaleR to support statistical analysis of **Big Data**, and is sold as subscriptions for workstations, servers, Hadoop and databases.

- Single-user licenses are available free for academic users (I still have it with VB GUI) as well as users competing in **Kaggle data mining** competitions.

- More on its blog site: https://blog.revolutionanalytics.com/

Revolution R Enterprise, R with sparkR and sparklyr package for R Studio and Azure/SQL:

- R in Azure SQL and SQL server is also available with sparkR & sparklyr!
  - **https://spark.apache.org/docs/latest/sparkr.html**
  - **https://spark.rstudio.com/ #Connection to H2O is also possible with sparklyr!**
  - **https://cloudblogs.microsoft.com/sqlserver/2021/06/30/looking-to-the-future-for-r-in-azure-sql-and-sql-server/**

Big Data in R:
https://www.columbia.edu/~sjm2186/EPIC_R/EPIC_R_BigData.pdf

Strategies:

- Option I: Make the data smaller
- Option II: Get a bigger computer
- Option III: Use data.table rather than data.frame
- Option IV: Buffer the data set on disk
- Option V: Split it up

Question/Queries?

Thank you!

@shitalbhandary