Otago Data School

Murray Cadzow

2018-11-21

Contents

1 Prerequisites **5**

2 Introduction **7**

3 Getting Set Up **9**
- 3.1 Required packages . . . 9
- 3.2 Project Setup . . . 9
- 3.3 Config Files . . . 11
- 3.4 Getting to know our data . . . 11

4 Data Manipulation and Visualisation **13**
- 4.1 Learning objectives . . . 13
- 4.2 Data manipulation with dplyr . . . 13
- 4.3 Data visualisation with ggplot2 . . . 14

5 Reports and Documentation **15**
- 5.1 Learning objectives . . . 15
- 5.2 Topics . . . 15

6 Generic and Functional **17**
- 6.1 functions . . . 17
- 6.2 dplyr . . . 17
- 6.3 purrr . . . 17

7 Databases and API's **19**
- 7.1 Learning objectives . . . 19
- 7.2 SQL . . . 20
- 7.3 API's . . . 20

8 Bioconductor **21**
- 8.1 Introduction to Bioconductor . . . 21
- 8.2 Dealing with genomic ranges . . . 21
- 8.3 DBI . . . 21

9 Annotation **23**
- 9.1 Genomic Annotation formats . . . 23
- 9.2 Getting annotations . . . 23
- 9.3 Creating Annotations . . . 23

10 Containers **25**
- 10.1 Docker . . . 25
- 10.2 Singularity . . . 25

11 Putting It All Together **27**

12 Summary **29**

Chapter 1

Prerequisites

This course is designed to follow on from:

Data Carpentry Ecology

Software Carpentry - Shell, R, and Git

Chapter 2

Introduction

This the course book for the Otago Data School. The course is under development.

Chapter 3

Getting Set Up

Learning objectives

- Organize files and directories for a set of analyses as an R Project, and understand the purpose of the working directory.
- Configure git the first time it is used on a computer.
- Understand the meaning of the –global configuration flag.
- Understand the use of config files
- Create a git repository

3.1 Required packages

```
install.packages('testthat')
install.packages('devtools')
install.packages('usethis')
install.packages('tidyverse')
install.packages('bookdown')
install.packages('here')
```

3.2 Project Setup

Create a new project in RStudio called *data_school*

3.2.1 Project directory setup

Now we want to create the following directory structure inside the project directory

```
data_school/
  |- data/
  |- data_output/
  |- documents/
  |- fig_output/
  |- scripts/
```

We also want to create a README file to describe the contents of the project to remind ourselves (and others) about what this project is about and what to expect in each directory

In R we can do this with the `use_readme_md()` function from the usethis package.

```
usethis::use_readme_md()
```

3.2.2 Adding version control

Throughout these lessons we're going to try and replicate a workflow that follows best practices and as such including version control for our scripts and documents is needed.

There are multiple ways this can be done, we're going to focus initially on the command line method to set this up.

The first thing we need to do is to create the git repository that is going to watch the files we tell it to so that we can keep track of the changes between versions.

Before we do anything with git we first need to make sure that everything is configured. This usually a single setup that only needs to be done once per machine.

If you have a github account set your username and email to match your github details

Set your user name

```
git config --global user.name="my name"
```

Set your email

```
git config --global user.email="my@email.com"
```

set the way git interprets line endings

```
# On macOS and Linux:
git config --global core.autocrlf input

# On Windows:
git config --global core.autocrlf true
```

3.2.2.1 Setup the Git repository

```
git init
```

Next we're going to tell git which directories to not watch

```
nano .gitignore
```

Now we need to tell git about the .gitignore file

```
git add .gitignore
```

3.2.2.2 Adding a remote repository

create a new repository in github (Don't add a README or LICENSE).

Now grab the url and add it as a remote for your local repository

```
git remote --add origin https://github.com/username/reponame
```

3.3 Config Files

An often overlooked part of setting up is the creation and maintainence of config files to control various settings in your environment and can be used to customise the environments to how you work.

We're going to look at a few

- bash config
- Rprofile

3.4 Getting to know our data

Chapter 4

Data Manipulation and Visualisation

4.1 Learning objectives

dplyr

- Describe the purpose of the dplyr and tidyr packages.
- Select certain columns in a data frame with the dplyr function select.
- Select certain rows in a data frame according to filtering conditions with the dplyr function filter .
- Link the output of one dplyr function to the input of another function with the 'pipe' operator %>%.
- Add new columns to a data frame that are functions of existing columns with mutate.
- Use the split-apply-combine concept for data analysis.
- Use summarize, group_by, and count to split a data frame into groups of observations, apply a summary statistics for each group, and then combine the results.
- Describe the concept of a wide and a long table format and for which purpose those formats are useful.
- Describe what key-value pairs are.
- Reshape a data frame from long to wide format and back with the spread and gather commands from the tidyr package.
- Export a data frame to a .csv file.

ggplot2

- Produce scatter plots, boxplots, and time series plots using ggplot.
- Set universal plot settings.
- Describe what faceting is and apply faceting in ggplot.
- Modify the aesthetics of an existing ggplot plot (including axis labels and color).
- Build complex and customized plots from data in a data frame.

4.2 Data manipulation with dplyr

- select()
- filter()
- mutate()
- arrange()
- tally()/count()
- group_by()
- summarise()

4.3 Data visualisation with ggplot2

- ggplot()
- aes()
- geom_point()
- geom_line()
- facet_wrap()
- theme()

Chapter 5

Reports and Documentation

5.1 Learning objectives

- understand the contents of a YAML header
- create and compile a markdown document
- understand the contents of code chunk
- compile a document to multiple output formats

5.2 Topics

- Value of reproducible reports
- Basics of Markdown
- R code chunks
- Chunk options
- Inline R code
- Other output formats

Chapter 6

Generic and Functional

Learning objectives

- Define a function that takes arguments.
- Return a value from a function.
- Check argument conditions with stopifnot() in functions.
- Test a function.
- Set default values for function arguments.
- Explain why we should divide programs into small, single-purpose functions.

6.1 functions

6.2 dplyr

- {summarise,mutate}_at()
- {summarise,mutate}_if()
- {summarise,mutate}_all()

6.3 purrr

- map()
  - _dbl
  - _int
  - _lgl
  - _chr
  - _df
  - _dfr
  - _dfc
- map2()
- pmap()
- imap()
- flattern
- at_depth

Chapter 7

Databases and API's

This lesson is going to focus on how to programatically retrieve (genomic) data from online repositorys such as Ensembl, UCSC, and, BioMart

7.1 Learning objectives

7.1.1 SQL

- Explain the difference between a table, a record, and a field.
- Explain the difference between a database and a database manager.
- Write a query to select all values for specific fields from a single table.
- Write queries that display results in a particular order.
- Write queries that eliminate duplicate values from data.
- Write queries that select records that satisfy user-specified conditions.
- Explain the order in which the clauses in a query are executed.
- Write queries that calculate new values for each selected record.
- Explain how databases represent missing information.
- Explain the three-valued logic databases use when manipulating missing information.
- Write queries that handle missing information correctly.
- Define aggregation and give examples of its use.
- Write queries that compute aggregated values.
- Trace the execution of a query that performs aggregation.
- Explain how missing data is handled during aggregation.
- Explain the operation of a query that joins two tables.
- Explain how to restrict the output of a query containing a join to only include meaningful combinations of values.
- Write queries that join tables on equal keys.
- Explain what primary and foreign keys are, and why they are useful.
- Explain what an atomic value is.
- Distinguish between atomic and non-atomic values.
- Explain why every value in a database should be atomic.
- Explain what a primary key is and why every record should have one.
- Identify primary keys in database tables.
- Explain why database entries should not contain redundant information.
- Identify redundant information in databases.
- Write statements that create tables.
- Write statements to insert, modify, and delete records.

- Write short programs that execute SQL queries.
- Trace the execution of a program that contains an SQL query.
- Explain why most database applications are written in a general-purpose language rather than in SQL.

7.2 SQL

7.2.1 dplyr/dbplyr

7.2.2 Refs

Software Carpentry SQL lesson

7.3 API's

7.3.1 REST

Ref https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/

7.3.1.1 BioMart

https://asia.ensembl.org/info/data/biomart/biomart_restful.html

7.3.1.2 Ensembl

https://github.com/Ensembl/ensembl-rest/wiki

7.3.1.3 UCSC

http://genomewiki.ucsc.edu/index.php/Programmatic_access_to_the_Genome_Browser

Chapter 8

Bioconductor

8.1 Introduction to Bioconductor

8.2 Dealing with genomic ranges

8.2.1 IRanges

- IRanges()
- start()
- end()
- width()
- intersect()
- union()
- reduce()
- subsetByOverlaps()

8.2.2 Genomic Ranges

- GRanges()
- seqnames()

8.3 DBI

8.3.1 Homo.sapiens

Chapter 9

Annotation

9.1 Genomic Annotation formats

- BED
- GTF/GFF

9.2 Getting annotations

9.2.1 AnnotationHub

9.3 Creating Annotations

Refs - https://dockflow.org/workflow/annotation-genomic-ranges/

Chapter 10

Containers

10.1 Docker

10.2 Singularity

Chapter 11

Putting It All Together

In this section we're going to create a reproducible workflow that involves aspects of all the previous lessons.

Refs

- https://nanx.me/talks/jsm2018-liftr-nanxiao.pdf
- https://dockflow.org

Chapter 12

Summary