Projects

Teams of 2 - no individual projects, no larger groups.
No teams with all members from the same department!

Email us (`boyanbejanov@cmail.carleton.ca`; `olga.baysal@carleton.ca`) your *team name* (optional), *team members* and *proposal* by January 25, 2016, 11:59 PM.

Proposal should describe your *problem statement*, *set of objectives/research questions*, and the *dataset* you plan to explore.
Keep it short - within 1-2 page limit.

Data Project Roles

- **Sponsor** - the person who wants the result; decides if the project is success or failure.
- **Client** - expert in the field; end user of the resulting model.
- **Data Scientist** - performs the necessary analysis and delivers results to sponsor and client.
- **Data Architect** - responsible for all the data and its storage.
- **Operations** - manages infrastructure.

Stages of a Data Science Project

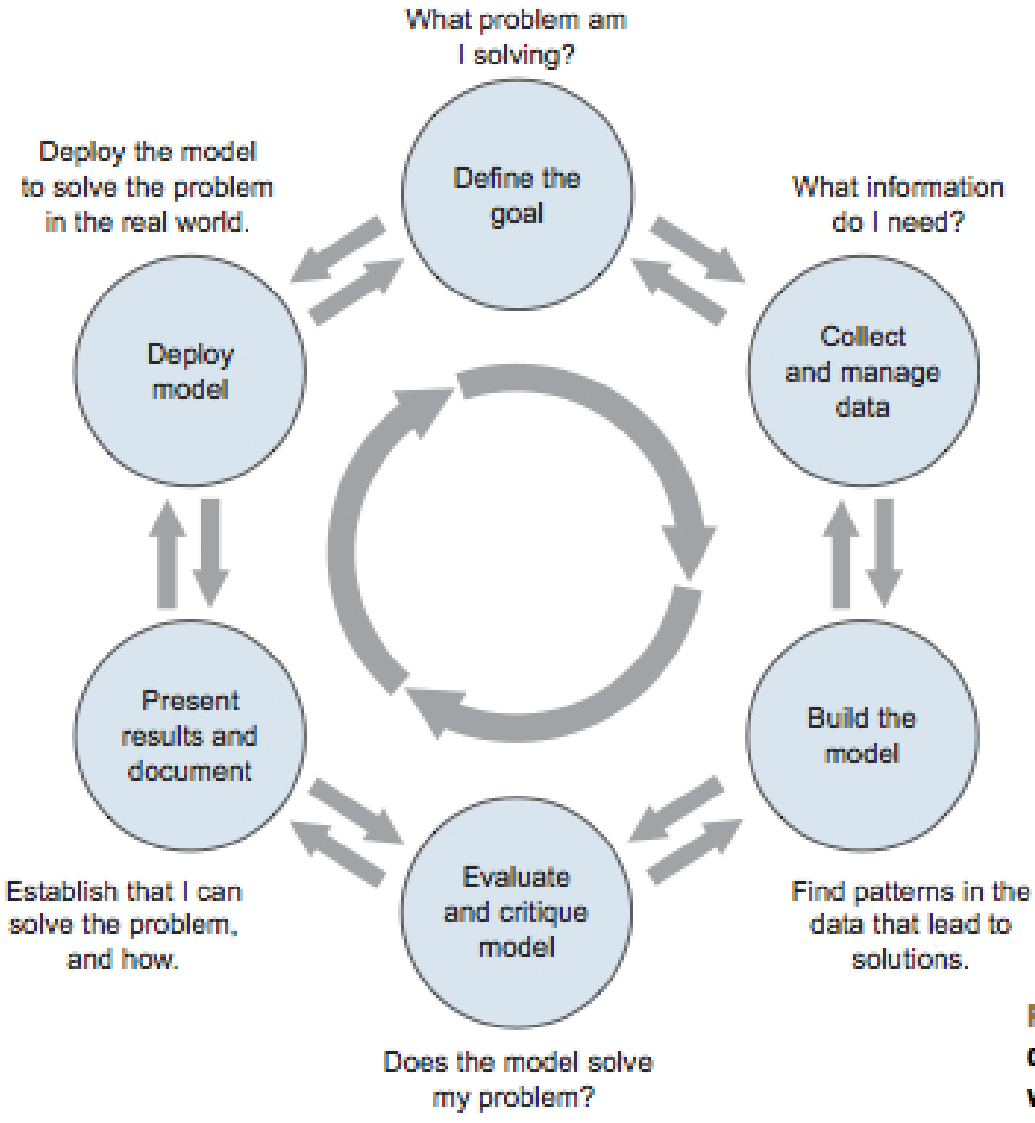

Figure 1.1 The lifecycle of a data science project: loops within loops

VCL Cloud

- Support provided by Sylvain Pitre: `Sylvain.Pitre@carleton.ca`.
- Having VCL problems? Contact Blake Henderson: `blake.henderson@carleton.ca`
- Create reservation using your browser
  - `http://orec.rdc.uottawa.ca`, select CarletonUniversity and login with your MyCarletonOne
  - In "New reservation" select the CognosBI_sshfs_RedHat_6.5_64bit image
  - Click "Create Reservation"
- To use IBM Cognos
  - in browser go to "Current Reservations" and click "Connect!"
  - in ssh client run the `ssh` command from browser, then *wait 5 min*
    change *acct@machine* with User ID and ip address
  - in Firefox go to `http://127.0.0.1:8080/ibmcognos`
- To use R, Python
  - in browser go to "Current Reservations" and click "Connect!"
  - in ssh client run `ssh -X acct@machine`
    change *acct@machine* with User ID and IP address
- Suggested ssh client for Windows: MobaXterm
  `http://mobaxterm.mobatek.net/download.html`

Basic R

- Load one of the standard datasets in R

```
> data(cars)
```

- It's a data frame with two variables. Compute the mean of each variable.

```
> colMeans(cars)
```

- Compute the minimum of each variable.

```
> for(i in 1:2) print(min(cars[,i]))
> sapply(cars, min)
```

- Plot

```
> plot(cars)
```

- Linear regression

```
> reg <- lm(dist ~ speed, data=cars)
> abline(reg, col="blue")
```

- Summary

```
> summary(cars)
> summary(reg)
```

Basic R

- Use `? name` to get help about *name*
- Functions `getwd` and `setwd` get and set the working directory
- Working with matrices:

```
M <- matrix(1:12, nrow=3, ncol=4); M
V <- matrix( runif(4), 4, 1); V
# Use %*% to multiply matrices
M %*% V
# Use * for element-wise multiplication
M * 2
# Beware recycling of arguments of different lengths!
M * c(2, -2)
# Use t() to transpose
t(M) * c(1,1,1)
# Functions rbind and cbind add rows/columns
cbind(M, V)
```

Basic R

- Working with strings

```
# Double quotes are preferred, single quotes work as well
'Hello World!'
# Strings are scalar values, not vectors of characters
length("Hello World!")
# Function c() creates a vector of strings
c("Hello", "World")
# Use paste to concatenate strings
paste("Hello", "World")
paste("Hello", "World", sep=", ")
# function strsplit does the opposite of paste
strsplit("Hello, World", ", ")
# strsplit uses regular expressions
strsplit('2 + x + x^2', '\\+')
# also available functions: grep, sub
```

Working with Data

18 January, 2016

From last class

- Ask a question
- Get relevant data
- Prepare data for analysis
  - outliers, missing values, incorrect values
- Explore data
  - understand the world as it is (was)

} Today

- Statistical model
  - estimate/train and validate model
  - predict what will (likely) happen
- Communicate results
  - tell a story
  - recommend

Getting data

- Data stored in files
- Data stored in a database
- API access to online data
- Data scraping

The R Data Import/Export manual covers the first two in detail

`http://cran.r-project.org/doc/manuals/r-release/R-data.html`

Data stored in files

Data file formats

- Delimited values
  - Comma-separated (.csv)
  - Tab-separated (.tsv)
  - Other, e.g. pipe-separated (.psv) with separator `'|'`
- Fixed width field
- Other structured data formats
  - Spreadsheets
  - HDF - Hierarchical Data Format
  - NetCDF - Network Common Data Form
- Markup languages
- XML - Extensible Markup Language
- JSON - JavaScript Object Notation
- Ad hoc file formats

Data stored in files

Delimited values

Use `read.table`

```
titanic.url <-
  "http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic.txt"

titanic <- read.table( titanic.url,  sep=",", header=TRUE)
```

Specializations:

- `read.csv` and `read.csv2` set defaults for comma-separated values
- `read.delim` and `read.delim2` set defaults for tab-separated values

Data stored in files

Fixed width field text files

- Use `read.fwf`

Ad hoc text files

- `scan`
- `readLines`
- `readChar`, `readBin`

Data stored in files

Excel spreadsheets

- Use package `XLConnect` or `xlsx`. Do not require MS Excel to be installed.

HDF5 / NetCDF4

- Use package `ncdf4` for NetCDF.
- Use package `rhdf5` for HDF5.
- Binary files suitable for numerical scientific data.
- Datasets (arrays), groups, attributes.

Data stored in files

XML

- Extensible Markup Language
- Text file containing markup elements and content.
- Markup elements
  - Can be nested to form tree structure.
  - Content enclosed between tags:
    `<p> Paragraph text. </p>`
  - Or empty-element tags: `<img src="lake.png">`
- Use `XML` package.

```
library(XML)
doc <- xmlParse(
    "http://www.w3schools.com/xml/cd_catalog.xml")
df <- xmlToDataFrame(doc)
df
```

Data stored in files

JSON

- JavaScript Object Notation
- Text file with name-value pairs in the syntax of JavaScript

```
{ "TITLE": [ "Stop", "Unchain my heart" ],
  "ARTIST": [ "Sam Brown", "Joe Cocker" ],
  "COUNTRY": [ "UK", "USA"],
  "YEAR": [ 1988, 1987 ] }
```

- Use one of packages `jsonlite`, `rjson`, or `RJSONIO`

Data stored in database

Why use a DBMS?

- DBMS = Database Management System
- Multiple tables with relationships among them
- Fast access to parts of the data
- Concurrent access
- Standardized access
- Scalability
- Reliability
- Security
- Fault-tolerance

Data stored in database

Types of DBMS

- Relational database model (RDBMS)
  - Relational data model divides variables into multiple tables (relations) to minimize redundancy.
  - This process is called normalization; guarantees data integrity.
  - RDBMS is transaction oriented (OLTP)
- Dimensional database model
  - Special case of relational model
  - Two types of tables: facts and dimensions
  - Not necessarily normalized
  - Used in analytically based DBMS (OLAP)

Data stored in database

About RDBMS

- Commercial: Oracle, Microsoft SQL Server, DB2, Sybase.
- Small-system: MySQL, PostgreSQL, Microsoft Access.
- Commercial ones put more emphasis on features such as security, high-availability, disaster recovery.
- Standard interface to all RDBMS is a language called SQL (Structured Query Language)
  - SQL Tutorial at `http://www.w3schools.com/sql/default.asp`

Data stored in database

SQL operations

- Working with records (rows) of data:
- `SELECT` - extract rows from tables
- `INSERT INTO` - insert rows into table
- `UPDATE` - update values
- `DELETE` - delete rows from table
- Operations on tables:
- `CREATE/ALTER/DROP TABLE`
- Operations on databases:
- `CREATE/ALTER/DROP DATABASE`

Data stored in database

SQL functions

- `AVG`, `MIN`, `MAX`, `COUNT`, `SUM`, `FIRST`, `LAST`
- and few others.

The `SELECT` statement:

```
SELECT columns
FROM table
WHERE condition
ORDER BY columns
```

Data stored in database

Joining tables

```
SELECT columns
FROM table1
kind JOIN table2
ON table1.column=table2.column
```

where `kind` is one of

- `INNER` - rows that have the same value in both tables
- `LEFT` - all rows from the left table with matching rows from the right table and `NULL` where there is no match
- `RIGHT` - similar to `LEFT`
- `FULL` - all rows from both tables with `NULL` where there is no match

Data stored in database

Access from R

- Mostly standardized with a front-end back-end model
- Front-end is package `DBI` (Database Interface)
- Multiple back-ends, each specific to a DBMS, e.g `RMySQL`, `ROracle`, `RPostgreSQL`, `RSQLite`, `RJDBC`
- `RJDBC` requires a JDBC driver

- `dbDriver`,
- `dbConnect`, `dbDisconnect`
- `dbGetQuery`
- `dbListTables`, `dbReadTable`, `dbWriteTable`, `dbRemoveTable`

Data stored in database

Access from R

- Open Database Connectivity defines a standard client-server model for access to DBMS
- Package `RODBC` provides R interface to ODBC
- ODBC support is built into Windows
- in Linux/UNIX requires a driver, such as unixODBC
- System-wide configuration to set up DSN (Data Source Name)

- `odbcConnect`, `odbcClose`
- `sqlQuery`
- `sqlTables`, `sqlFetch`, `sqlSave`, `sqlDrop`

Data stored in database

Example using DBI

```
library(DBI)
library(RSQLite)
con <- dbConnect(dbDriver("SQLite"), "titanic.db")
dbWriteTable(con, name="titanic", value=titanic)
dbListTables(con)
a <- dbGetQuery(con, paste0("select pclass, survived ",
       "from titanic where sex='female' order by age"))
# compare next two
with(a, table(pclass, survived))
with(titanic, table(pclass, survived, sex))
# compare next two
mean(titanic$age, na.rm=TRUE)
dbGetQuery(con, paste0("select avg(age) from titanic"))
# close connection when done
dbDisconnect(con)
```

API access to online data

API = Application Programming Interface

- Facebook: `Rfacebook`
- Twitter: `streamR`
- Tumblr: `tumblR`
- Google: `RGoogleTrends`
- Wikipedia: `WikipediR`
- New York Times: `rtimes`
- Many more, check:

  `http://cran.r-project.org/web/views/WebTechnologies.html`

Data scraping

Wikipedia: a computer program extracts data from human-readable output coming from another program.

- Write a program that
  - downloads a web page and parses it (usually HTML)
  - collects data from the web pages
  - follows links according to rules and repeats
- It may or may not be legal
- Risky and unreliable source of data

- In R one can use package `XML` with XPath expressions to parse HTML code for data and links to follow. Also package `RCurl` allows more protocols than R's internal methods.

Data scraping

Easy example

```
library(XML)
mlb <- readHTMLTable(
    "http://www.baseball-reference.com/leagues/MLB/2012.shtml")
length(mlb)
names(mlb)
head(mlb2012$teams_standard_batting)
```

Note: this is not really scraping, because our program does not look for and follow links.

Prepare Data for Analysis

- Put data in the necessary format
- Make sure data variables are of the correct type
- Apply conversions and transformations as needed
- Address missing values
- Check for and deal with incorrect values

Data Types

- Data Semantics:

  The real world interpretation of the data, e.g. height, date, name.

- Data Type:

  Determined from the possible values and operations, e.g. numerical, categorical.

Stevens' Levels of Measurement

- Nominal or Categorical
  - e.g. colour, gender, race
  - Finite set of values without quantity
  - Allowed operation: $=$, $\neq$
  - Allowed statistics: mode
- Ordinal
  - e.g. education, IQ scores
  - Finite set of quantity values
  - New allowed operations: $<$, $>$
  - New allowed statistics: median
- Quantitative (interval or additive)
  - Numerical values, no well-defined zero
  - e.g. date, temperature, location
  - New allowed operations: $+$, $-$
  - New allowed statistics: mean (average)
- Quantitative (ratio or multiplicative)
  - e.g. amount
  - New allowed operations: $*$, $/$

Semantics affects data type

- Sometimes it makes sense to aggregate quantitative data into ordinal, or to ignore ranking and treat ordinal as nominal.
- Date is usually additive quantity, but it my be converted to categorical (season), ordinal (day of the week), or even promoted to multiplicative quantity by fixing the origin (days since some event)
- Colour is usually nominal, but it may be ordinal (colors of the rainbow) or even quantitative (greyscale)
- IQ scores are ordinal, but not quantitative. Why?

Dimensional data model

- In a dimensional data model usually:
  - dimensions are nominal, ordinal, or additive quantities
    e.g. date/time, make-and-model, supplier
  - facts are ratio or additive quantities
    e.g. volume of sales, retail price.

Types of data sets

- Structured data
  - arrays (1-d, 2-d, 3-d, N-d)
  - temporal data (e.g. time series, 1-d)
  - location data (e.g. maps, 1-d, 2-d, 3-d, 4-d)
  - relational databases (N-d)
  - hierarchical data (trees)
  - network data (general graphs)
  - images, audio
- Unstructured data
  - text analytics - tweets, emails, articles, others

R Data Types

Data types in R

- for nominal with few categories use factors
- if dichotomous, may use logical
- if nominal has too many levels a vector may be better, e.g. name, address, SIN
- for ordinal use ordered factors
  `ordered(x,...)` is the same as
  `factor(x,...,ordered=TRUE)`
- for dates use the `Date` class `as.Date` function
  see also classes `POSIXct` and `POSITlt`

Missing Values

- Why are data missing?
  - Missing completely at random (MCAR)
  - Missing at random (MAR)
  - Missing not at random (MNAR)
- How to deal with missing values?
  - Delete records - okay in a large dataset with few missing values
  - Impute values
- Also check for incorrect values