OPERATIONALIZING R WITH ORACLE AND POSTGRES

EXAMPLES OF DATA SCIENCE PROJECTS AT THE CITY OF CHICAGO

PRESENTED TO CHICAGO R USER GROUP, AUGUST 30, 2018

Example Projects

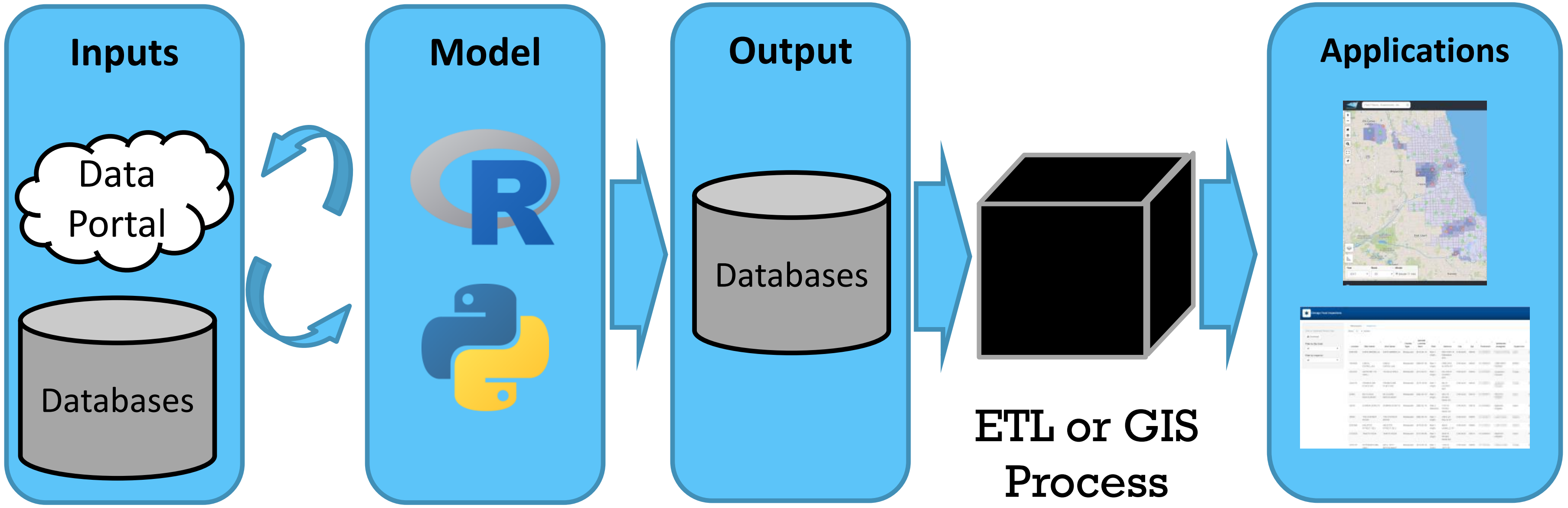
Clear Water

Food Inspection Forecasting

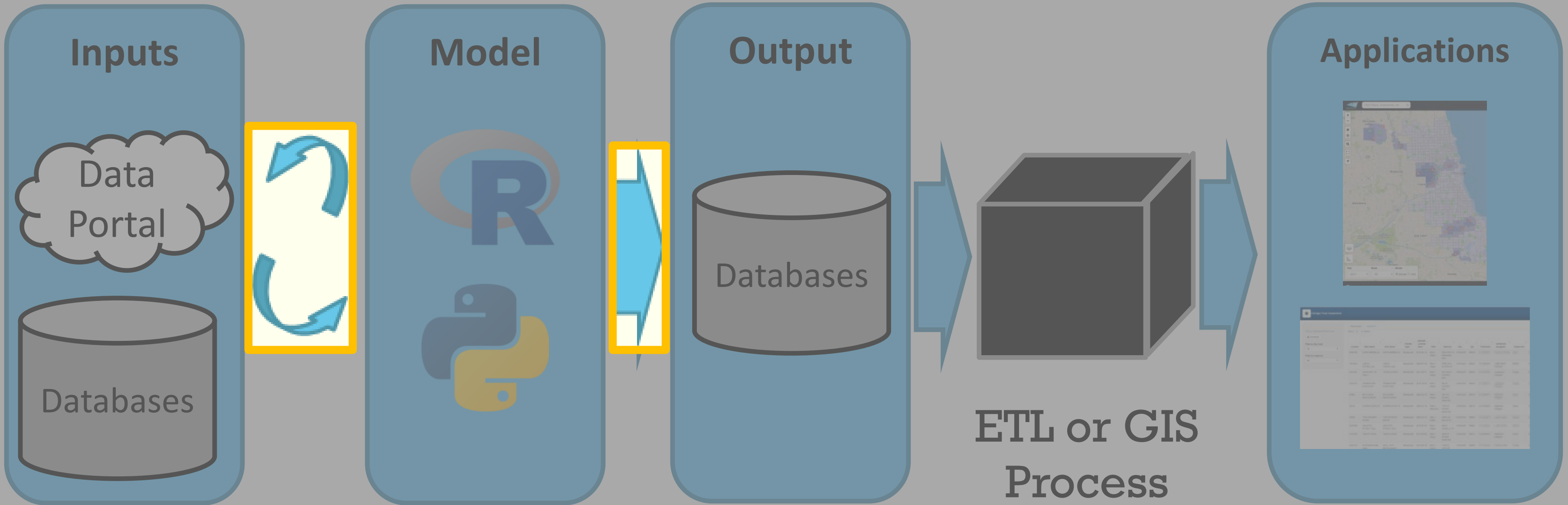
Vector Borne Diseases

Lead Safe

Project Pattern



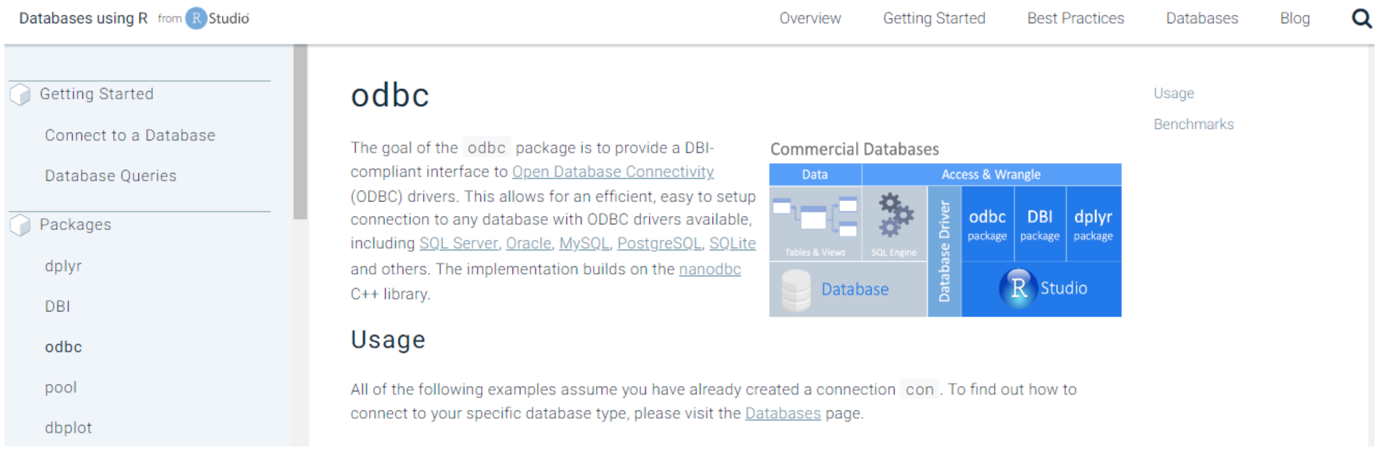
Project Pattern



RESOURCES

RSTUDIO

- RStudio has released a wiki for Database connection info:
  https://db.rstudio.com/odbc/



Getting Started

Connect to a Database

Database Queries

Packages

dplyr

DBI

odbc

pool

dbplot

odbc

The goal of the `odbc` package is to provide a DBI-compliant interface to Open Database Connectivity (ODBC) drivers. This allows for an efficient, easy to setup connection to any database with ODBC drivers available, including SQL Server, Oracle, MySQL, PostgreSQL, SQLite and others. The implementation builds on the nanodbc C++ library.

Usage

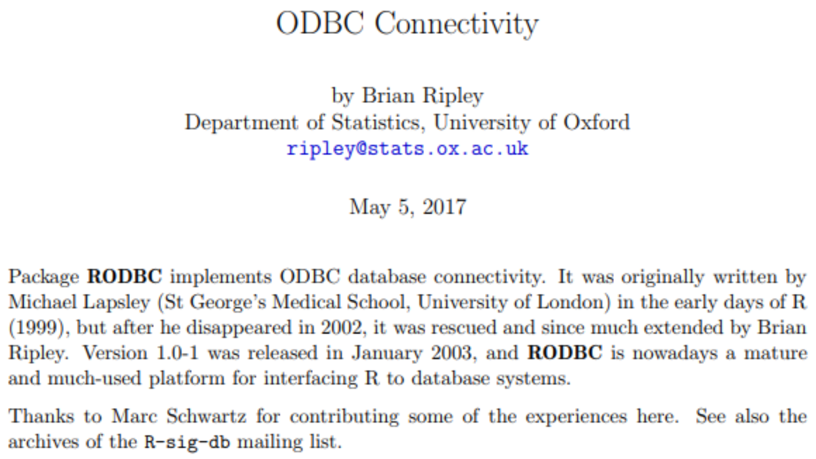
Benchmarks

Usage

All of the following examples assume you have already created a connection `con`. To find out how to connect to your specific database type, please visit the Databases page.

RODBC / Brian Ripley

- Ripley's RODBC vignette is still my first reference, and goes way beyond ODBC:
  https://cran.r-project.org/web/packages/RODBC/vignettes/RODBC.pdf



ODBC Connectivity

by Brian Ripley
Department of Statistics, University of Oxford
ripley@stats.ox.ac.uk

May 5, 2017

Package **RODBC** implements ODBC database connectivity. It was originally written by Michael Lapsley (St George's Medical School, University of London) in the early days of R (1999), but after he disappeared in 2002, it was rescued and since much extended by Brian Ripley. Version 1.0-1 was released in January 2003, and **RODBC** is nowadays a mature and much-used platform for interfacing R to database systems.

Thanks to Marc Schwartz for contributing some of the experiences here. See also the archives of the `R-sig-db` mailing list.

BASICS
DATABASES WITHIN R

DB PACKAGES

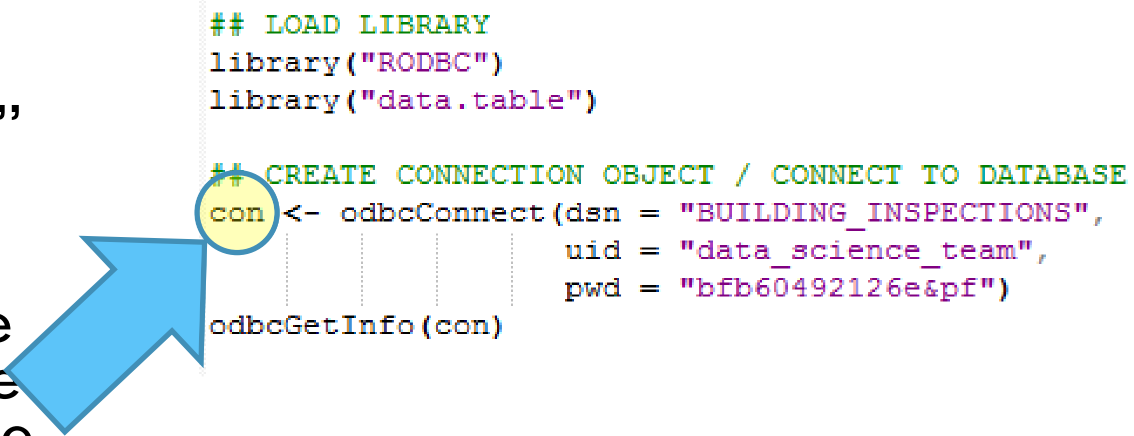
- **RODBC** – This is the original R database connector, written by Brian Ripley and released in 2003
- **ROracle** – Released by Oracle and rarely maintained, but it does work
- **RPostgreSQL** – New, well documented, great dev team
- **mongolite** – This is our default MongoDB package
- **DBI** – Not used directly, but it’s the workhorse behind 2 & 3

BASIC CONCEPTS

- Create a connection to the driver or named ODBC connection
- Execute commands against the connection
  - Commands can be meta (e.g. “ListTables”)
  - Commands can be SQL (e.g. “sendQuery”)
  - Commands can use a “cursor” that allows you to “page” through your response

EXAMPLE – CONNECTING

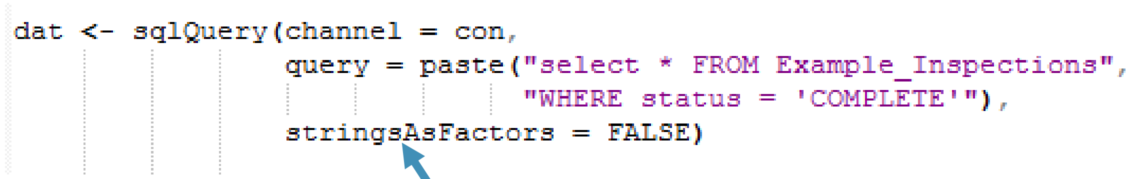
- Create a “channel” or “connection” object
- The connection is typically named “ch” or “con”
- Meta commands like odbcGetInfo provide information about the connection



```
## LOAD LIBRARY
library("RODBC")
library("data.table")

## CREATE CONNECTION OBJECT / CONNECT TO DATABASE
con <- odbcConnect(dsn = "BUILDING_INSPECTIONS",
                   uid = "data_science_team",
                   pwd = "bfb60492126e&pf")
odbcGetInfo(con)
```

EXAMPLE –TYPICAL QUERY

- There are meta commands like "get table" that are basically "select star" commands
- You can put nearly any query into functions that submit generic queries
- The package attempts to format the return, sometimes there are issues

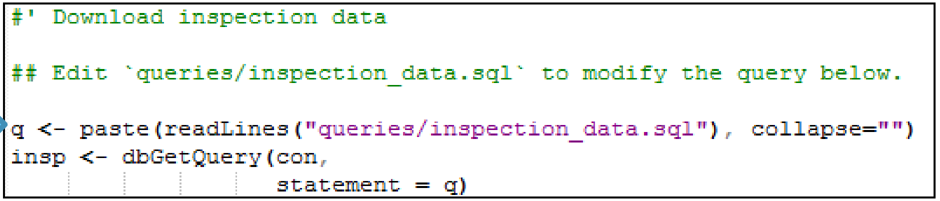


```
dat <- sqlQuery(channel = con,
                query = paste("select * FROM Example_Inspections",
                              "WHERE status = 'COMPLETE'"),
                stringsAsFactors = FALSE)
```

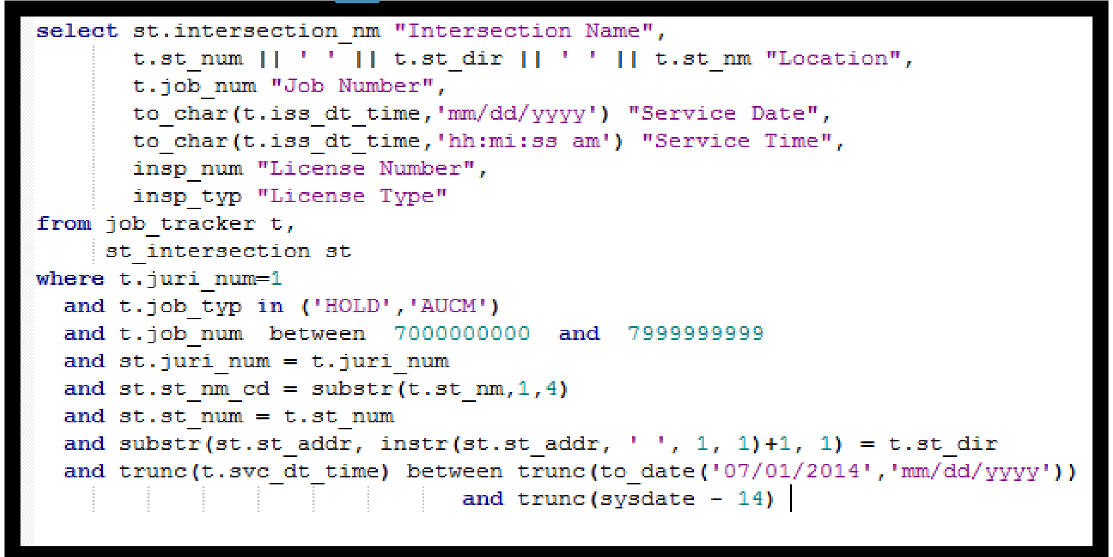
NOTE: `stringsAsFactors = FALSE` can greatly improves performance!

EXAMPLE – COMPLEX QUERY

- For complex queries it is convenient to store the query is a separate file and use readLines to get query



```r
#' Download inspection data

## Edit `queries/inspection_data.sql` to modify the query below.

q <- paste(readLines("queries/inspection_data.sql"), collapse="")
insp <- dbGetQuery(con,
                   statement = q)
```

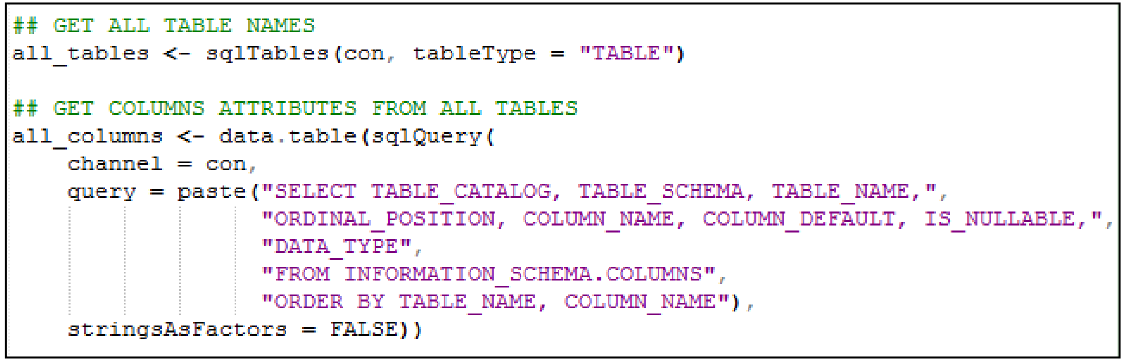


```sql
select st.intersection_nm "Intersection Name",
       t.st_num || ' ' || t.st_dir || ' ' || t.st_nm "Location",
       t.job_num "Job Number",
       to_char(t.iss_dt_time,'mm/dd/yyyy') "Service Date",
       to_char(t.iss_dt_time,'hh:mi:ss am') "Service Time",
       insp_num "License Number",
       insp_typ "License Type"
from job_tracker t,
     st_intersection st
where t.juri_num=1
  and t.job_typ in ('HOLD','AUCM')
  and t.job_num  between  7000000000  and  7999999999
  and st.juri_num = t.juri_num
  and st.st_nm_cd = substr(t.st_nm,1,4)
  and st.st_num = t.st_num
  and substr(st.st_addr, instr(st.st_addr, ' ', 1, 1)+1, 1) = t.st_dir
  and trunc(t.svc_dt_time) between trunc(to_date('07/01/2014','mm/dd/yyyy'))
                                   and trunc(sysdate - 14)
```

EXAMPLE – EXPLORING THE DB

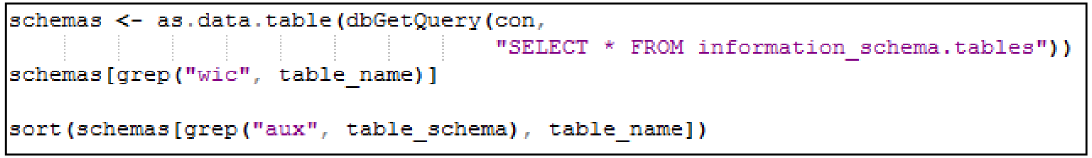
- There are many ways to find out what's in the database
  - Meta commands like "sqlTables" will tell you tables
  - System specific commands will also provide information on the tables
- Everything uses the "connection" object to access the database via the driver

ORACLE SYNTAX



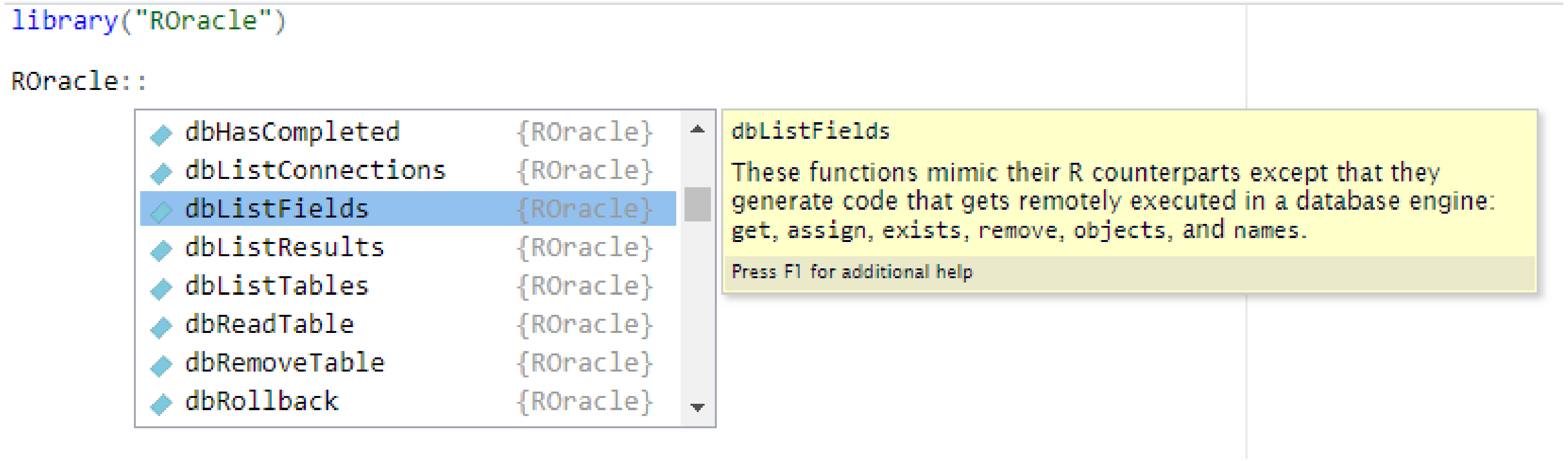
```
## GET ALL TABLE NAMES
all_tables <- sqlTables(con, tableType = "TABLE")

## GET COLUMNS ATTRIBUTES FROM ALL TABLES
all_columns <- data.table(sqlQuery(
    channel = con,
    query = paste("SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,",
                  "ORDINAL_POSITION, COLUMN_NAME, COLUMN_DEFAULT, IS_NULLABLE,",
                  "DATA_TYPE",
                  "FROM INFORMATION_SCHEMA.COLUMNS",
                  "ORDER BY TABLE_NAME, COLUMN_NAME"),
    stringsAsFactors = FALSE))
```

POSTGRES SYNTAX



```
schemas <- as.data.table(dbGetQuery(con,
                                    "SELECT * FROM information_schema.tables"))
schemas[grep("wic", table_name)]

sort(schemas[grep("aux", table_schema), table_name])
```

PACKAGE METHODS



```
library("ROracle")

ROracle::
```

```
dbHasCompleted      {ROracle}
dbListConnections   {ROracle}
dbListFields        {ROracle}
dbListResults       {ROracle}
dbListTables        {ROracle}
dbReadTable         {ROracle}
dbRemoveTable       {ROracle}
dbRollback          {ROracle}
```

dbListFields

These functions mimic their R counterparts except that they generate code that gets remotely executed in a database engine: get, assign, exists, remove, objects, and names.

Press F1 for additional help

- Many packages (RODBC, ROracle, RPostgreSQL) have built in convenience functions that *mimic* SQL functions
- You can use these to read / write tables
- They also have something that "just executes" a query, e.g. RPostgreSQL

CODE ORGANIZATION

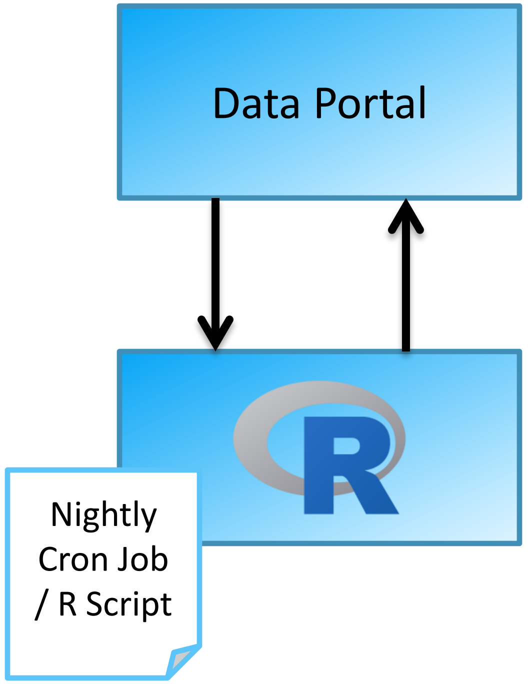
INLINE QUERIES

- Simple strategy, works well if you only do SELECT *
- Gets messy with bigger queries

```
dat <- sqlQuery(channel = con,
                query = paste("select * FROM Example_Insp",
                              "WHERE status = 'COMPLETE'"),
                stringsAsFactors = FALSE)
```

TWO STRATEGIES FOR KEEPING THE CODE PUBLIC

Option 1: Totally Public

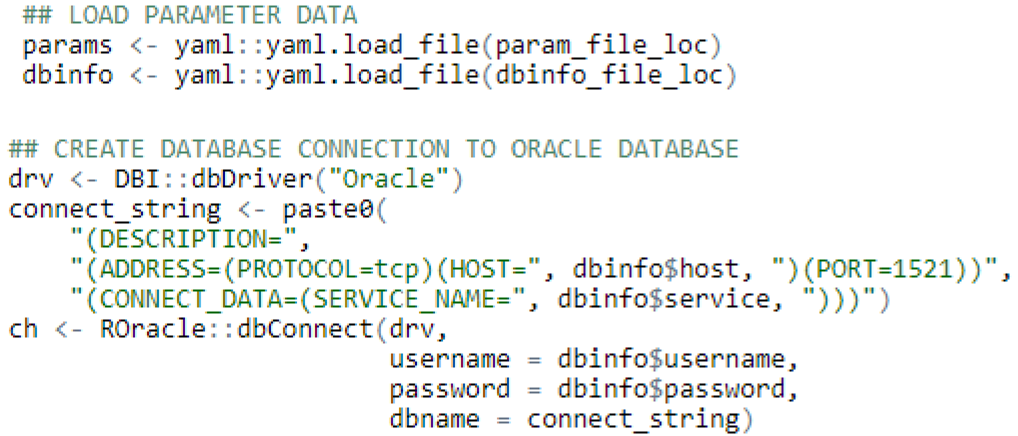


Clear Water

- Uses RSocrata to leverage the Socrata data portal
- Everything stays public
- Nothing to worry about with Postgres or Oracle

Option 2: YAML Files

- Put all the code online
- Keep connection info in plain text in one folder (like ./params)
- Add folder to .gitignore
- Bonus: Use YAML files
  - More readable
  - Named fields
  - Only one dependency

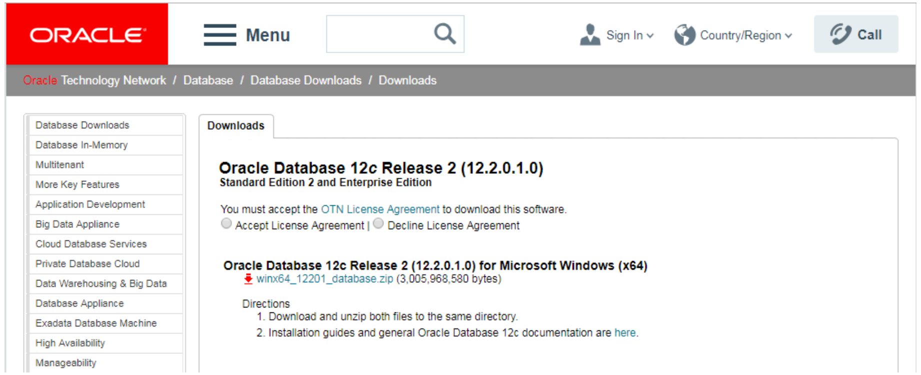


```
 ## LOAD PARAMETER DATA
 params <- yaml::yaml.load_file(param_file_loc)
 dbinfo <- yaml::yaml.load_file(dbinfo_file_loc)

## CREATE DATABASE CONNECTION TO ORACLE DATABASE
drv <- DBI::dbDriver("Oracle")
connect_string <- paste0(
    "(DESCRIPTION=",
    "(ADDRESS=(PROTOCOL=tcp)(HOST=", dbinfo$host, ")(PORT=1521))",
    "(CONNECT_DATA=(SERVICE_NAME=", dbinfo$service, ")))")
ch <- ROracle::dbConnect(drv,
                         username = dbinfo$username,
                         password = dbinfo$password,
                         dbname = connect_string)
```

WINDOWS CLIENT ENVIRONMENT

INSTALL DRIVERS

- Download drivers from Oracle
- Main flavors
  - 11g
  - 12c
- Many driver options, I used trial and error to select correct driver



Oracle Database 12*c* Release 2 (12.2.0.1.0)
Standard Edition 2 and Enterprise Edition

You must accept the OTN License Agreement to download this software.
Accept License Agreement | Decline License Agreement

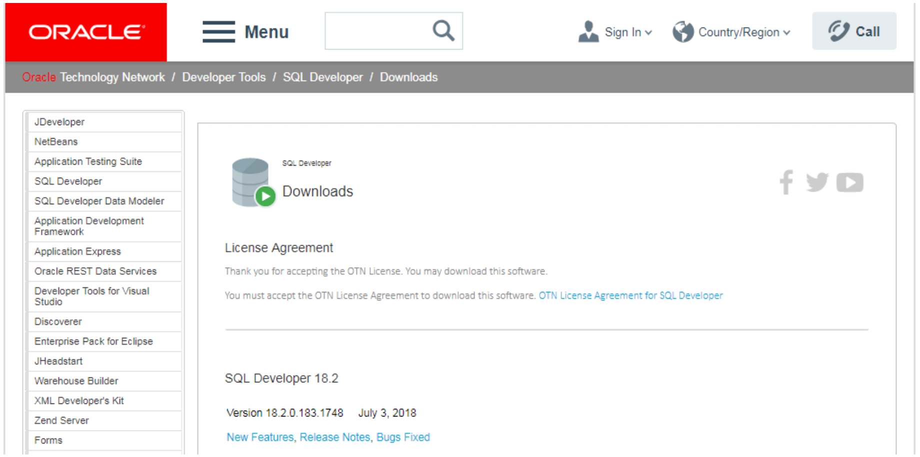
Oracle Database 12c Release 2 (12.2.0.1.0) for Microsoft Windows (x64)
winx64_12201_database.zip (3,005,968,580 bytes)

Directions
1. Download and unzip both files to the same directory.
2. Installation guides and general Oracle Database 12c documentation are here.

INSTALL SQL DEVELOPER

(Optional)

- It’s very helpful to have SQL Developer for checking connections /testing queries


ORACLE
Menu
Sign In
Country/Region
Call
Oracle Technology Network / Developer Tools / SQL Developer / Downloads
JDeveloper
NetBeans
Application Testing Suite
SQL Developer
SQL Developer Data Modeler
Application Development Framework
Application Express
Oracle REST Data Services
Developer Tools for Visual Studio
Discoverer
Enterprise Pack for Eclipse
JHeadstart
Warehouse Builder
XML Developer's Kit
Zend Server
Forms
SQL Developer
Downloads

License Agreement

Thank you for accepting the OTN License. You may download this software.

You must accept the OTN License Agreement to download this software. OTN License Agreement for SQL Developer

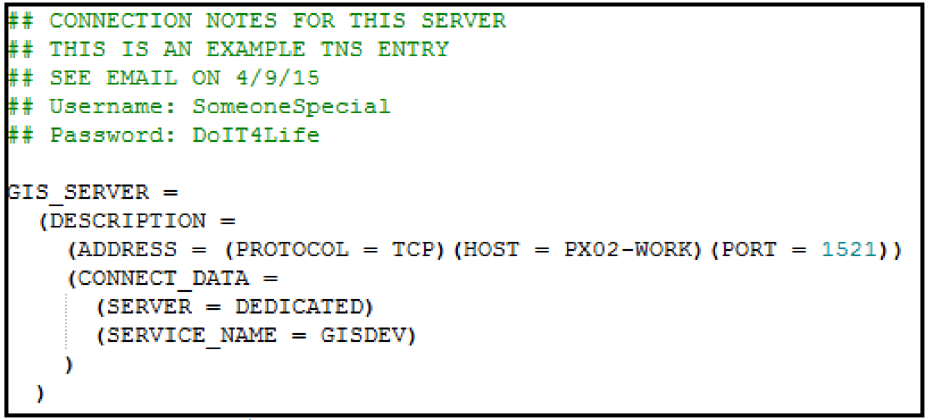
SQL Developer 18.2

Version 18.2.0.183.1748 July 3, 2018

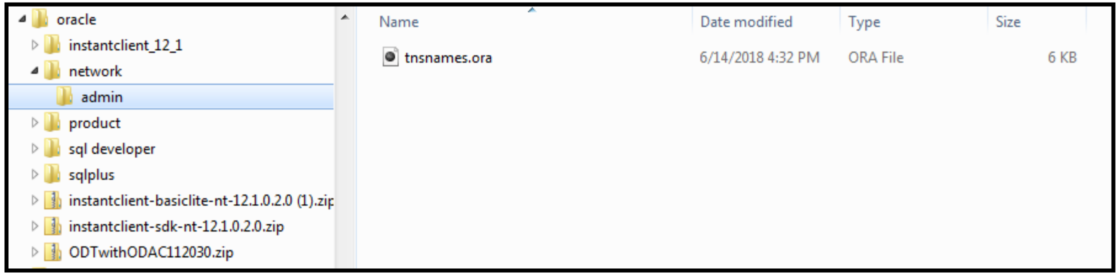
New Features, Release Notes, Bugs Fixed

SET UP TNS NAMES

- The sane location for your Oracle install is c:\oracle
- Add connection info to tnsnames.ora



```
## CONNECTION NOTES FOR THIS SERVER
## THIS IS AN EXAMPLE TNS ENTRY
## SEE EMAIL ON 4/9/15
## Username: SomeoneSpecial
## Password: DoIT4Life

GIS_SERVER =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = PX02-WORK)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = GISDEV)
    )
  )
```



```
oracle
  instantclient_12_1
  network
    admin
  product
  sql developer
  sqlplus
  instantclient-basiclite-nt-12.1.0.2.0 (1).zip
  instantclient-sdk-nt-12.1.0.2.0.zip
  ODTwithODAC112030.zip
```

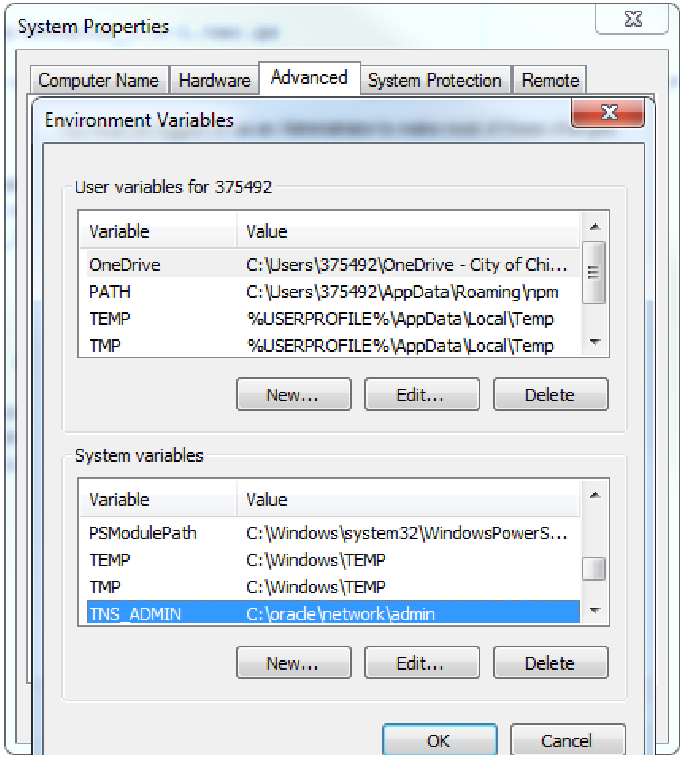
Name	Date modified	Type	Size
tnsnames.ora	6/14/2018 4:32 PM	ORA File	6 KB

ADD TNS_ADMIN

- Set an **environmental variable** for the TNS File

- Needed for
  - SQL Developer
  - ODBC management
  - ROracle package

- At this point you can add TNS connections by name in SQL Developer
  - Look for a green plus sign
  - Choose TNS for connection type
  - Named connections appear in dropdown



User variables for 375492

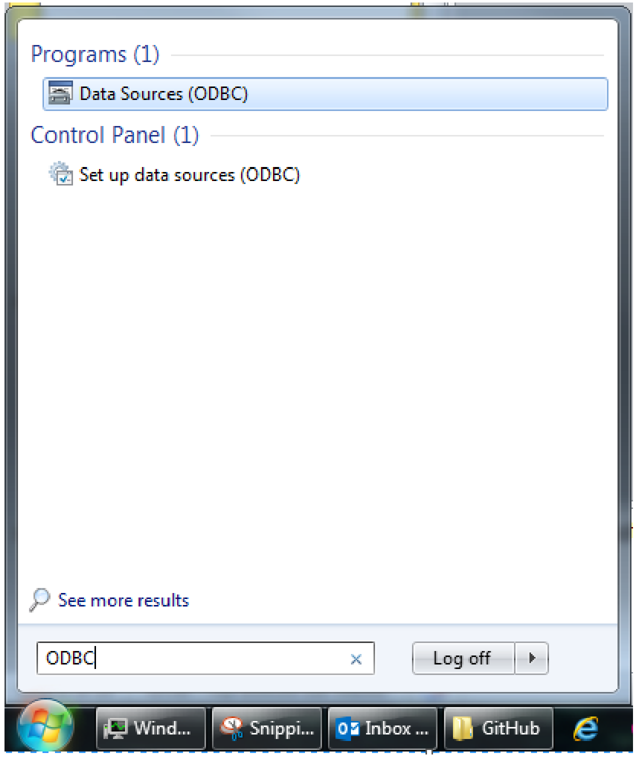
Variable	Value
OneDrive	C:\Users\375492\OneDrive - City of Chi...
PATH	C:\Users\375492\AppData\Roaming\npm
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
PSModulePath	C:\Windows\system32\WindowsPowerS...
TEMP	C:\Windows\TEMP
TMP	C:\Windows\TEMP
TNS_ADMIN	C:\oracle\network\admin

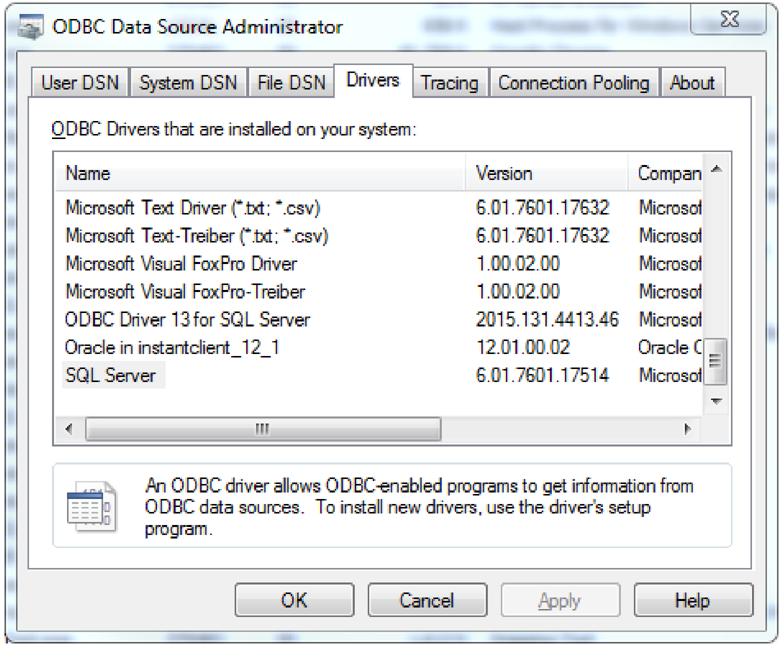
ODBC CONNECTION

- Windows has a "data sources" manager
- Search for ODBC



ODBC CONNECTION

- Default drivers are listed
- Your newly installed driver is listed here



ODBC Data Source Administrator

User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

ODBC Drivers that are installed on your system:

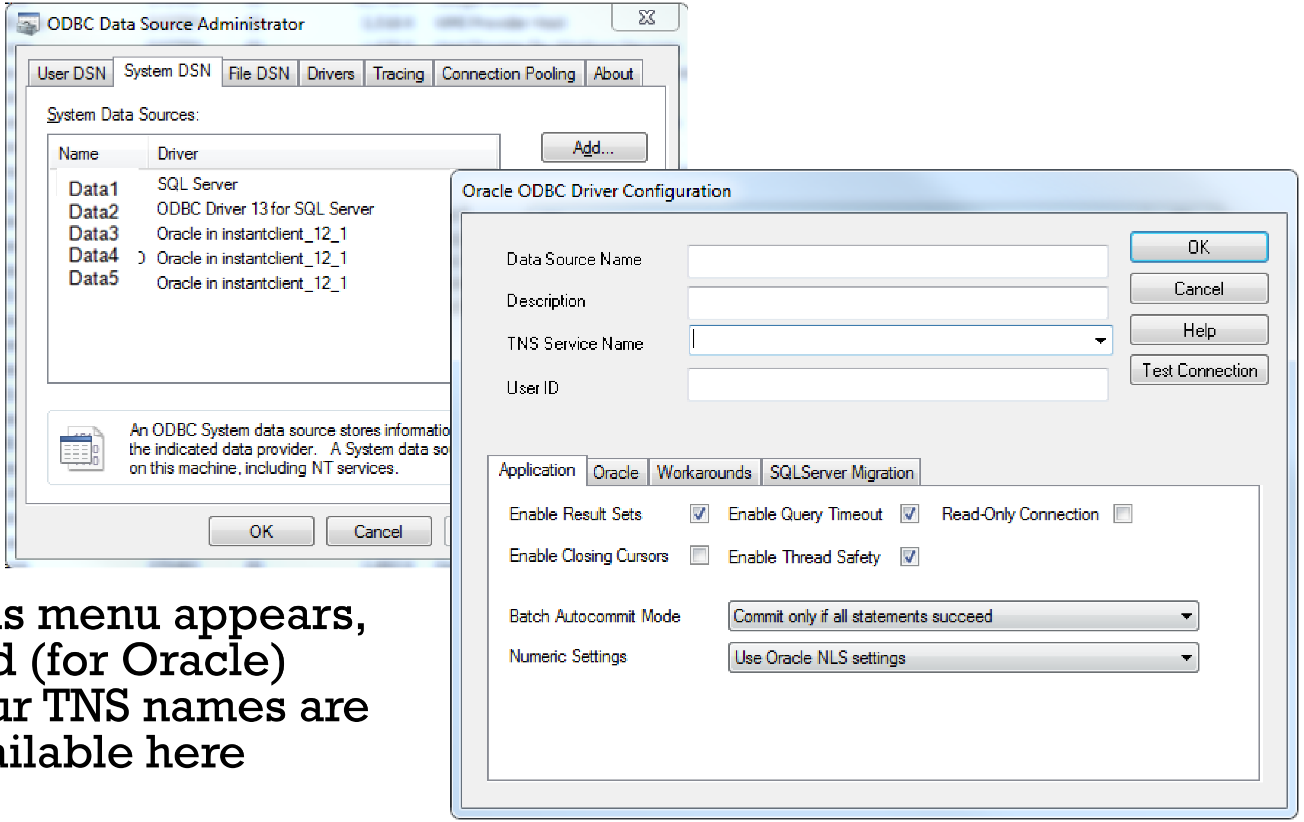
Name	Version	Compan
Microsoft Text Driver (*.txt; *.csv)	6.01.7601.17632	Microsof
Microsoft Text-Treiber (*.txt; *.csv)	6.01.7601.17632	Microsof
Microsoft Visual FoxPro Driver	1.00.02.00	Microsof
Microsoft Visual FoxPro-Treiber	1.00.02.00	Microsof
ODBC Driver 13 for SQL Server	2015.131.4413.46	Microsof
Oracle in instantclient_12_1	12.01.00.02	Oracle C
SQL Server	6.01.7601.17514	Microsof

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

OK | Cancel | Apply | Help

ADD SYSTEM DSN

- This is a list of your data source names
- Click "Add…" and follow the prompts



ODBC Data Source Administrator

User DSN | System DSN | File DSN | Drivers | Tracing | Connection Pooling | About

System Data Sources:

Name	Driver
Data1	SQL Server
Data2	ODBC Driver 13 for SQL Server
Data3	Oracle in instantclient_12_1
Data4	Oracle in instantclient_12_1
Data5	Oracle in instantclient_12_1

Add...

An ODBC System data source stores informatio the indicated data provider. A System data so on this machine, including NT services.

OK | Cancel

Oracle ODBC Driver Configuration

Data Source Name:
Description:
TNS Service Name:
User ID:

OK | Cancel | Help | Test Connection

Application | Oracle | Workarounds | SQLServer Migration

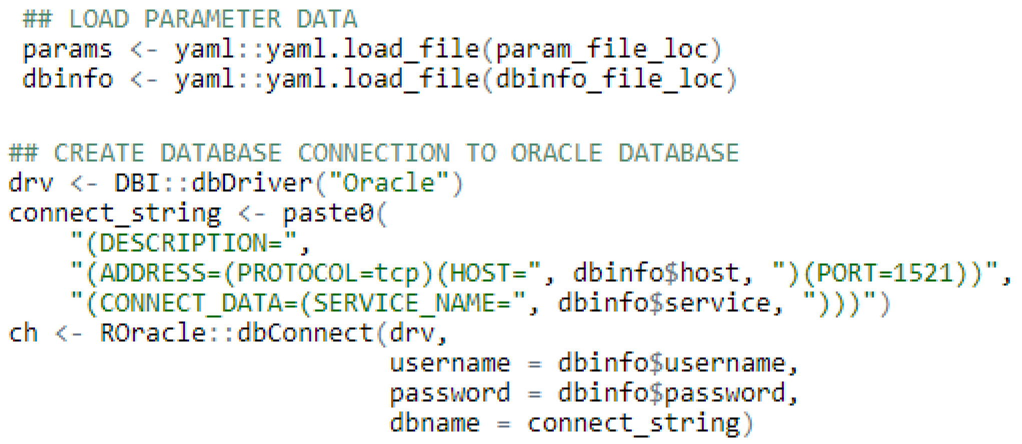
Enable Result Sets ✓ | Enable Query Timeout ✓ | Read-Only Connection ☐
Enable Closing Cursors ☐ | Enable Thread Safety ✓

Batch Autocommit Mode: Commit only if all statements succeed
Numeric Settings: Use Oracle NLS settings

- This menu appears, and (for Oracle) your TNS names are available here

SPECIFY TNS WITHIN R

- With the ROracle package, you can pass in the TNS entry
- Pros:
  - Simpler?
  - Code is more portable
  - SQL Developer is configured by default
- Cons:
  - ROracle is hard to install
  - Simpler to refer to a named entry
  - SQL Developer isn't configured



```
 ## LOAD PARAMETER DATA
 params <- yaml::yaml.load_file(param_file_loc)
 dbinfo <- yaml::yaml.load_file(dbinfo_file_loc)

## CREATE DATABASE CONNECTION TO ORACLE DATABASE
drv <- DBI::dbDriver("Oracle")
connect_string <- paste0(
    "(DESCRIPTION=",
    "(ADDRESS=(PROTOCOL=tcp)(HOST=", dbinfo$host, ")(PORT=1521))",
    "(CONNECT_DATA=(SERVICE_NAME=", dbinfo$service, ")))")
ch <- ROracle::dbConnect(drv,
                         username = dbinfo$username,
                         password = dbinfo$password,
                         dbname = connect_string)
```

ROracle

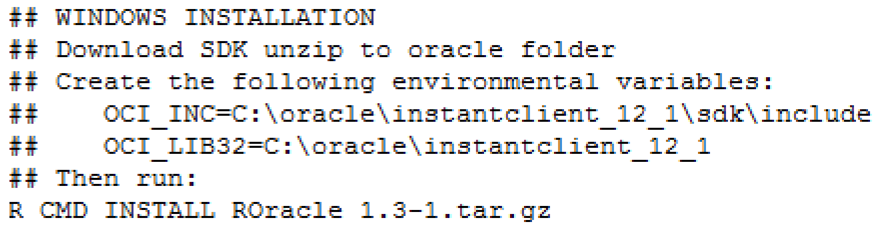
ROracle IN WINDOWS

- Oracle SDK is required
- SDK Note: you just unzip the files to the oracle folder and mingle them with other files that are already there

- More environmental variables are required (see picture on right)

- Must be installed at the command line, not within R*

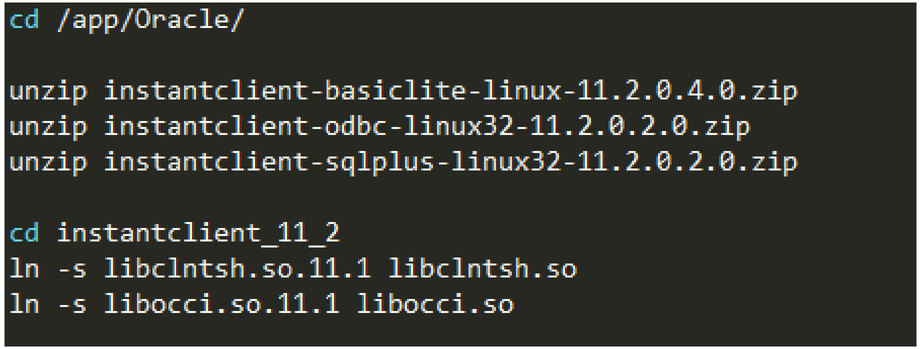
* I’m pretty sure of this, but not positive



```
## WINDOWS INSTALLATION
## Download SDK unzip to oracle folder
## Create the following environmental variables:
##    OCI_INC=C:\oracle\instantclient_12_1\sdk\include
##    OCI_LIB32=C:\oracle\instantclient_12_1
## Then run:
R CMD INSTALL ROracle 1.3-1.tar.gz
```

ROracle IN LINUX

- Install Oracle drivers

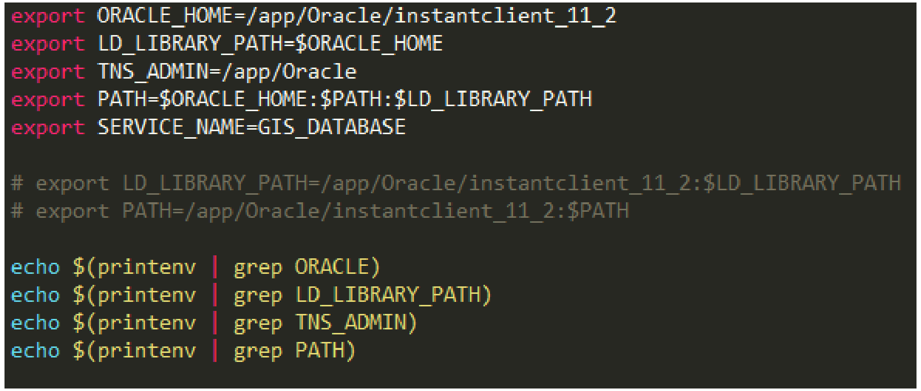


```
cd /app/Oracle/

unzip instantclient-basiclite-linux-11.2.0.4.0.zip
unzip instantclient-odbc-linux32-11.2.0.2.0.zip
unzip instantclient-sqlplus-linux32-11.2.0.2.0.zip

cd instantclient_11_2
ln -s libclntsh.so.11.1 libclntsh.so
ln -s libocci.so.11.1 libocci.so
```

ROracle IN LINUX

- Create a script to set the appropriate paths

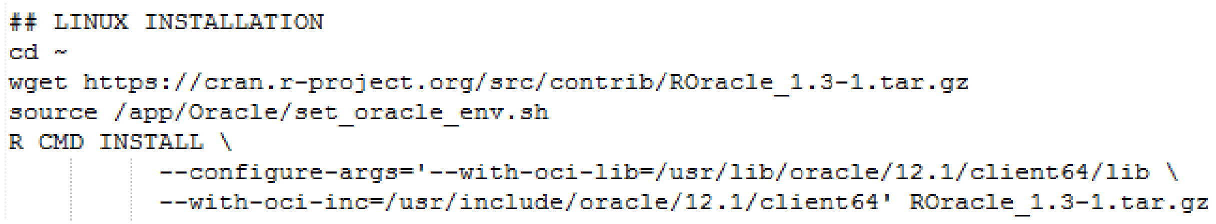
- I added my script to my ~/.bashrc



```
export ORACLE_HOME=/app/Oracle/instantclient_11_2
export LD_LIBRARY_PATH=$ORACLE_HOME
export TNS_ADMIN=/app/Oracle
export PATH=$ORACLE_HOME:$PATH:$LD_LIBRARY_PATH
export SERVICE_NAME=GIS_DATABASE

# export LD_LIBRARY_PATH=/app/Oracle/instantclient_11_2:$LD_LIBRARY_PATH
# export PATH=/app/Oracle/instantclient_11_2:$PATH

echo $(printenv | grep ORACLE)
echo $(printenv | grep LD_LIBRARY_PATH)
echo $(printenv | grep TNS_ADMIN)
echo $(printenv | grep PATH)
```

ROracle IN LINUX

- Install ROracle
- Note: I'm setting the environmental variables with set_oracle_env.sh

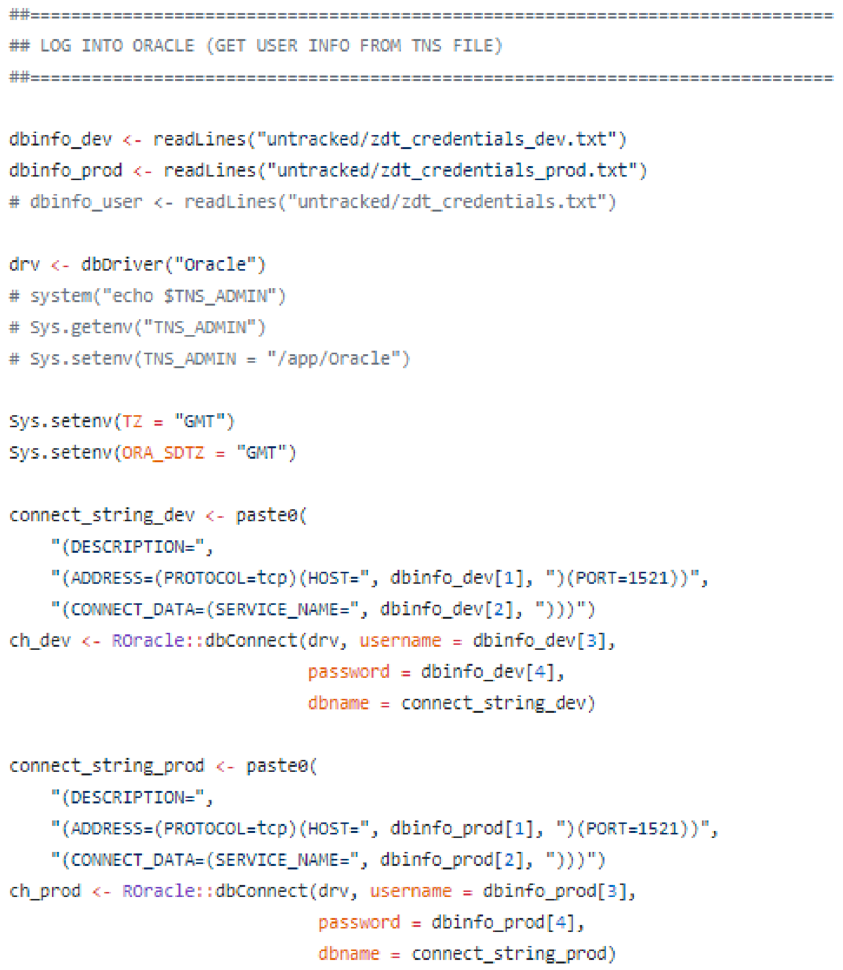


```
## LINUX INSTALLATION
cd ~
wget https://cran.r-project.org/src/contrib/ROracle_1.3-1.tar.gz
source /app/Oracle/set_oracle_env.sh
R CMD INSTALL \
          --configure-args='--with-oci-lib=/usr/lib/oracle/12.1/client64/lib \
          --with-oci-inc=/usr/include/oracle/12.1/client64' ROracle_1.3-1.tar.gz
```

TIME IN ORACLE

TIME ZONES

- The West Nile Prediction repository has full examples (url below)
- Note: you can get and set env variables from within R
- Note: You need to set two variables in order to upload date / times



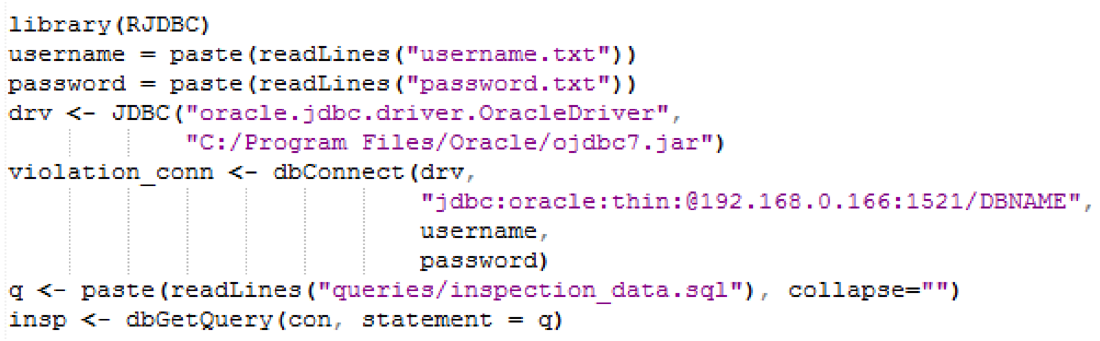
```
##==============================================================================
## LOG INTO ORACLE (GET USER INFO FROM TNS FILE)
##==============================================================================

dbinfo_dev <- readLines("untracked/zdt_credentials_dev.txt")
dbinfo_prod <- readLines("untracked/zdt_credentials_prod.txt")
# dbinfo_user <- readLines("untracked/zdt_credentials.txt")

drv <- dbDriver("Oracle")
# system("echo $TNS_ADMIN")
# Sys.getenv("TNS_ADMIN")
# Sys.setenv(TNS_ADMIN = "/app/Oracle")

Sys.setenv(TZ = "GMT")
Sys.setenv(ORA_SDTZ = "GMT")

connect_string_dev <- paste0(
    "(DESCRIPTION=",
    "(ADDRESS=(PROTOCOL=tcp)(HOST=", dbinfo_dev[1], ")(PORT=1521))",
    "(CONNECT_DATA=(SERVICE_NAME=", dbinfo_dev[2], ")))")
ch_dev <- ROracle::dbConnect(drv, username = dbinfo_dev[3],
                             password = dbinfo_dev[4],
                             dbname = connect_string_dev)

connect_string_prod <- paste0(
    "(DESCRIPTION=",
    "(ADDRESS=(PROTOCOL=tcp)(HOST=", dbinfo_prod[1], ")(PORT=1521))",
    "(CONNECT_DATA=(SERVICE_NAME=", dbinfo_prod[2], ")))")
ch_prod <- ROracle::dbConnect(drv, username = dbinfo_prod[3],
                              password = dbinfo_prod[4],
                              dbname = connect_string_prod)
```

https://github.com/Chicago/west-nile-virus-predictions/

RJDBC

RJDBC – THE PROMISE OF JAVA

- RJDBC is a lot easier
- I’ve had issues in Linux (user error?)
- With RJDBC the setup should be simpler and platform independent
- This simple example (for Windows) should work in Linux



```
library(RJDBC)
username = paste(readLines("username.txt"))
password = paste(readLines("password.txt"))
drv <- JDBC("oracle.jdbc.driver.OracleDriver",
            "C:/Program Files/Oracle/ojdbc7.jar")
violation_conn <- dbConnect(drv,
                            "jdbc:oracle:thin:@192.168.0.166:1521/DBNAME",
                            username,
                            password)
q <- paste(readLines("queries/inspection_data.sql"), collapse="")
insp <- dbGetQuery(con, statement = q)
```

To be continued…

THANK YOU

Contact Info:

Gene Leynes

Data Scientist @ City of Chicago

Email: datascience@cityofchicago.org

Websites:

https://github.com/Chicago/west-nile-virus-predictions