



**crunchy data**

# How Green Was My Valley

Spatial Analytics with PostgreSQL, PostGIS, R, and PL/R

**Joe Conway**

**[joe.conway@crunchydata.com](mailto:joe.conway@crunchydata.com)**

**[mail@joeconway.com](mailto:mail@joeconway.com)**

Crunchy Data

November 14, 2019

# Agenda

## Spatial Analytics

- Overview
- Ingesting Data
- Analytics

Full details of this talk available at:

<https://info.crunchydata.com/blog/spatial-analytics-with-postgres-postgis-plr>

# Datasets

- Administrative shape data
- Geocoded data (not covered here)
- MOD13A1 NDVI
  - Normalized Difference Vegetation Index
  - global, spatial raster data

[https://lpdaac.usgs.gov/dataset\\_discovery/modis/modis\\_products\\_table/mod13a1](https://lpdaac.usgs.gov/dataset_discovery/modis/modis_products_table/mod13a1)

# Components

- PostgreSQL
  - The world's most advanced open source database.
- PostGIS
  - A spatial database extender for PostgreSQL. Adds support for geographic objects allowing location queries to be run in SQL.
- R
  - An open source language and environment for statistical computing and graphics.
- PL/R
  - R Procedural Language Handler for PostgreSQL. Enables user-defined SQL functions to be written in the R language. Actively developed since early 2003.

<https://www.r-project.org>  
<http://www.postgis.net>  
<http://www.joeconway.com/plr>

# PostgreSQL and R Setup

- Install desired versions of PostgreSQL, PostGIS, R, and PL/R
- Create the database and install Postgres extensions
- Install variety of required/interesting R packages
- Be sure to install R packages as `root` or `postgres`

# RPostgreSQL Compatibility

- Allows prototyping using R, move to PL/R for production
- Queries performed in current database
- Driver/connection parameters ignored; dbDriver, dbConnect, dbDisconnect, and dbUnloadDriver are no-ops

# RPostgreSQL Compatibility Example

- PostgreSQL access from R

```
get_ndvi_mean<-function(rastfile) {  
  library(cairoDevice); library(RGtk2); library(sp); library(raster); library(RPostgreSQL)  
  
  ndvi <- brick(rastfile)  
  ndvi_mn <- cellStats(ndvi, stat = 'mean')  
  
  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")  
  sql.str <- "SELECT lyr_date FROM ndvi_dates(get_robj('ndvi_dt'))"  
  ndvi_dt <- dbGetQuery(conn, sql.str)  
  
  return(data.frame(ndvi_dt, ndvi_mn))  
}
```

# RPostgreSQL Compatibility Example

- Same function from PL/R

```
CREATE OR REPLACE FUNCTION get_ndvi_mean
  (rastfile text, OUT lyr_date date, OUT lyr_mean float8)
RETURNS setof RECORD AS $$
  library(cairoDevice); library(RGtk2); library(sp); library(raster); library(RPostgreSQL)

  ndvi <- brick(rastfile)
  ndvi_mn <- cellStats(ndvi, stat = 'mean')

  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")
  sql.str <- "SELECT lyr_date FROM ndvi_dates(get_robj('ndvi_dt'))"
  ndvi_dt <- dbGetQuery(conn, sql.str)

  return(data.frame(ndvi_dt, ndvi_mn))
$$ LANGUAGE 'plr' STRICT;
```

# Create Raw Image from SQL

```
CREATE OR REPLACE FUNCTION plot_ndvi_trend(rastfile text)
RETURNS bytea AS $$
  library(cairoDevice); library(RGtk2); library(sp); library(raster)
  pixmap <- gdkPixmapNew(w=1024, h=768, depth=24); asCairoDevice(pixmap)

  ndvi <- brick(rastfile)
  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")
  sql.str <- "SELECT lyr_date FROM ndvi_dates(get_robj('ndvi_dt'))"
  ndvi_dt <- dbGetQuery(conn, sql.str)
  ndvi_mn <- cellStats(ndvi, stat = 'mean')
  idx <- order(ndvi_dt)
  Data <- data.frame(ndvi_dt$lyr_date[idx], ndvi_mn[idx])
  plot(Data, type = "l", xlab = "Time", ylab = "NDVI")

  plot_pixbuf <- gdkPixbufGetFromDrawable(NULL, pixmap, pixmap$getColormap(),
                                          0, 0, 0, 0, 500, 500)
  gdkPixbufSaveToBufferv(plot_pixbuf, "png", character(0), character(0))$buffer
$$ LANGUAGE 'plr' STRICT;
```

# Use of Raw Image from SQL

- Need screen buffer on typical server:

```
Xvfb :1 -screen 0 1024x768x24
export DISPLAY=:1.0
```

- Calling it from PHP

```
<?php
$dbconn = pg_connect("...");
$rs = pg_query( $dbconn,
    "SELECT plr_get_raw(plot_ndvi_trend('/usr/local/bda/modis/ndvi_cooked/ndvi.tif'))");
$hexpic = pg_fetch_array($rs);
$cleandata = pg_unescape_bytea($hexpic[0]);

header("Content-Type: image/png");
header("Last-Modified: " .
    date("r", filectime($_SERVER['SCRIPT_FILENAME'])));
header("Content-Length: " . strlen($cleandata));
echo $cleandata;
?>
```

# Source and Ingest USA Counties

- Use `getData()` (`rgdal` package) to get shape layer for an administrative area
  - GADM: database of global administrative boundaries  
→ <http://www.gadm.org>
- Create a PostGIS table and store it there

# Create USA Counties Layer Table

```
CREATE OR REPLACE FUNCTION create_admin_layer(country text, level int, tablename text)
RETURNS text AS $$
  library(rgdal); library(RPostgreSQL)

  # download the requested administrative shapes and create the Postgres table
  dsn="PG:user='postgres' dbname='gis' host='localhost'"
  shapes = getData('GADM', country = country, level = level)
  writeOGR(shapes, dsn, tablename, "PostgreSQL")
  sql.str <- sprintf("CREATE INDEX %s_idx1 ON %s(name_1,name_2)", tablename, tablename)
  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")
  dbGetQuery(conn, sql.str)

  return("OK")
$$ LANGUAGE plr;

DROP TABLE IF EXISTS counties;
SELECT create_admin_layer(country := 'USA', level := 2, tablename := 'counties');
```

# Extract and Plot San Diego County

```
-- Note: might need to run "Xvfb :1 -screen 0 1024x768x24"
CREATE OR REPLACE FUNCTION plot_admin_layer(layername text, name1 text, name2 text)
RETURNS bytea AS $$
  library(rgeos); library(cairoDevice); library(RGtk2); library(RPostgreSQL)

  pixmap <- gdkPixmapNew(w=1024, h=768, depth=24); asCairoDevice(pixmap)

  proj_str <- "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")
  sql.str <- sprintf("SELECT st_astext(wkb_geometry) AS geom
                    FROM %s WHERE name_1 = '%s' AND name_2 = '%s'", layername, name1, name2)
  plot(readWKT(dbGetQuery(conn, sql.str), p4s=proj_str))

  plot_pixbuf <- gdkPixbufGetFromDrawable(NULL, pixmap, pixmap$getColormap(),
                                          0, 0, 0, 0, 500, 500)
  buffer <- gdkPixbufSaveToBufferv(plot_pixbuf, "png", character(0), character(0))$buffer
  return(buffer)
$$ LANGUAGE 'plr' STRICT;
```

# San Diego County Administrative Boundaries

```
SELECT plr_get_raw(plot_admin_layer('counties', 'California', 'San Diego'));
```



# San Diego County Administrative Boundaries - Reprojected

```
p4s <- "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"  
mp4s <- "+proj=laea +lat_0=45 +lon_0=-100 +x_0=0 +y_0=0 +a=6370997 +b=6370997 +units=m +no_defs"  
plot(spTransform(readWKT(dbGetQuery(conn, sql.str), p4s=p4s), CRS(mp4s)))
```



# Preprocessing

- Review downloaded files, eliminate anomalies (9 out of 1166)
- Run preprocessing function
  - Load area of interest (Aoi) administrative boundaries
  - Reproject Aoi to same projection as MOD13A1 raster data
  - Loop through the downloaded files
    - Unzip File of Interest (Foi)
    - Load NDVI, QA, and Acquisition rasters
    - Crop rasters to Aoi (trim matrix to Aoi extent)
    - Mask rasters to Aoi (mark cells outside Aoi as NA)
    - Use QA raster to mark cells with questionable data in NDVI raster as NA
    - Use Acquisition raster to calculate mean date (YYYY.pppp) of NDVI data
  - Capture NDVI rasters and Dates

# Preprocessing Function

```
# loop through all the MODIS files and preprocess
CREATE OR REPLACE FUNCTION process_raw_ndvi
  (mywd text, layername text, name1 text, name2 text)
RETURNS bytea AS $$

  [...too long to cover...]

  return(data.frame(ndvi_i, ndvi_dt))
$$ LANGUAGE 'plr' STRICT;
```

# Preprocessing Function

```
CREATE TABLE robjects(objname text primary key, obj bytea);

INSERT INTO robjects
SELECT 'ndvi_dt', process_raw_ndvi('/usr/local/bda/modis',
                                   'counties',
                                   'California',
                                   'San Diego');

--Time: 8010273.572 ms
```

# Plotting NDVI

```
CREATE OR REPLACE FUNCTION plot_ndvi(rastfile text, bricklayer int, plotfile text,  
                                     poilayer text, aoilayer text, aoiname1 text, aoiname2 text)  
RETURNS text AS $$  
  library(sp); library(raster); library(rgeos); library(rgdal); library(RPostgreSQL)  
  Sys.setenv("DISPLAY"=":0.0")  
  
  # required in R, noop in PL/R  
  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")  
  
  # CRS of the MODIS raster  
  mp4str <- "+proj=laea +lat_0=45 +lon_0=-100 +x_0=0  
            +y_0=0 +a=6370997 +b=6370997 +units=m +no_defs"  
  
  # plot to file if destination filename given  
  if (plotfile != "") png(plotfile, width = 1024, height = 768)  
  
  [...]
```

# Plotting NDVI

[...]

```
# retrieve boundaries for requested admin area
p4str <- "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
sql.str <- sprintf("SELECT st_astext(wkb_geometry) AS geom
                  FROM %s WHERE name_1 = '%s' and name_2 = '%s'",
                  aolayer, aoiname1, aoiname2)
sdc <- readWKT(dbGetQuery(conn, sql.str), p4s=p4str)

# reproject the boundary to the CRS of the MODIS raster and add to plot
plot(spTransform(sdc, CRS(mp4str)))

# read and plot requested raster layer
ndvi <- brick(rastfile)
ndvi_layer <- ndvi[[bricklayer]]
plot(ndvi_layer, add = TRUE)
```

[...]

# Plotting NDVI

[...]

```
# retrieve points of interest
```

```
dsn <- "PG:user='postgres' dbname='gis' host='localhost'"
```

```
poi <- readOGR(dsn=dsn, poilayer, stringsAsFactors = FALSE)
```

```
# reproject the poi to the CRS of the MODIS raster
```

```
poi <- spTransform(poi, CRS(mp4str))
```

```
# add poi to plot
```

```
plot(poi, col = "red", pch = 16, add = TRUE)
```

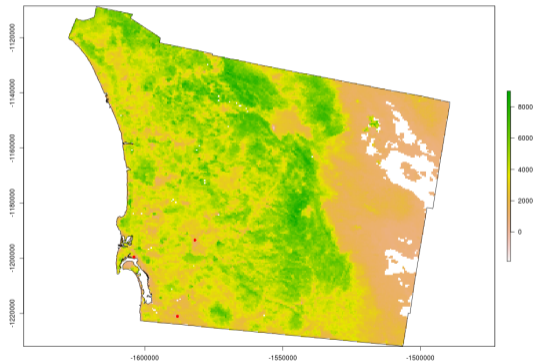
```
if (plotfile != "") dev.off()
```

```
return("OK")
```

```
$$ LANGUAGE 'plr' STRICT;
```

# Plotting NDVI

```
SELECT plot_ndvi('/usr/local/bda/modis/ndvi_cooked/ndvi.tif', 42, '/tmp/airports2.png',  
                'airports', 'counties', 'California', 'San Diego');
```



# NDVI - Average over Time

```
CREATE OR REPLACE FUNCTION get_ndvi_mean
  (rastfile text, OUT lyr_date date, OUT lyr_mean float8)
RETURNS setof RECORD AS $$
  library(sp); library(raster); library(RPostgreSQL)

  ndvi <- brick(rastfile)
  ndvi_mn <- cellStats(ndvi, stat = 'mean')

  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")
  sql.str <- "SELECT lyr_date FROM ndvi_dates(get_robj('ndvi_dt'))"
  ndvi_dt <- dbGetQuery(conn, sql.str)

  return(data.frame(ndvi_dt, ndvi_mn))
$$ LANGUAGE 'plr' STRICT;
```

# NDVI - Average over Time

```
SELECT lyr_date, lyr_mean
FROM get_ndvi_mean('/usr/local/bda/modis/ndvi_cooked/ndvi.tif') LIMIT 3;
```

lyr_date	lyr_mean
2000-02-29	3307.52445819025
2000-03-09	3358.51562339221
2000-03-11	3372.62659753491

(3 rows)

# NDVI - Average over Time

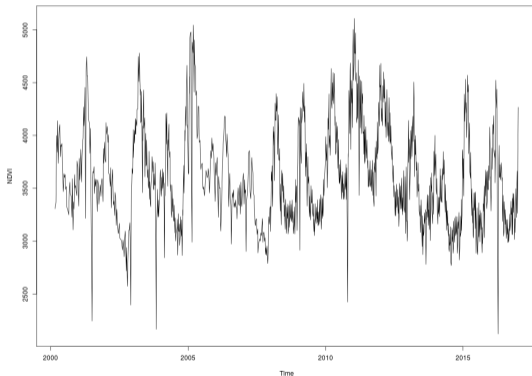
```
CREATE OR REPLACE FUNCTION plot_ndvi_trend(rastfile text)
RETURNS bytea AS $$
  library(cairoDevice); library(RGtk2); library(sp); library(raster); library(RPostgreSQL)
  pixmap <- gdkPixmapNew(w=1024, h=768, depth=24); asCairoDevice(pixmap)

  ndvi <- brick(rastfile)
  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")
  sql.str <- "SELECT lyr_date FROM ndvi_dates(get_robj('ndvi_dt'))"
  ndvi_dt <- dbGetQuery(conn, sql.str)
  ndvi_mn <- cellStats(ndvi, stat = 'mean')
  idx <- order(ndvi_dt)
  Data <- data.frame(ndvi_dt$lyr_date[idx], ndvi_mn[idx])
  plot(Data, type = "l", xlab = "Time", ylab = "NDVI")

  plot_pixbuf <- gdkPixbufGetFromDrawable(NULL, pixmap, pixmap$getColormap(),
                                          0, 0, 0, 0, 500, 500)
  gdkPixbufSaveToBufferv(plot_pixbuf, "png", character(0), character(0))$buffer
$$ LANGUAGE 'plr' STRICT;
```

# NDVI - Average over Time

```
SELECT plr_get_raw(plot_ndvi_trend('/usr/local/bda/modis/ndvi_cooked/ndvi.tif'));
```



# NDVI - Visualize Average by Month

```
# Pure R code here
plot_ndvi_year <- function(rastfile, layeryr, plotfile)
{

  library(sp); library(raster); library(RPostgreSQL)
  Sys.setenv("DISPLAY"=":0.0")

  # required in R, noop in PL/R
  conn <- dbConnect(dbDriver("PostgreSQL"), user="postgres", dbname="gis", host="localhost")

  # plot to file if destination filename given
  if (plotfile != "") png(plotfile, width = 1024, height = 768)

  sql.str <- "SELECT lyr_date FROM ndvi_dates(get_robj('ndvi_dt'))"
  ndvi_dt <- dbGetQuery(conn, sql.str)

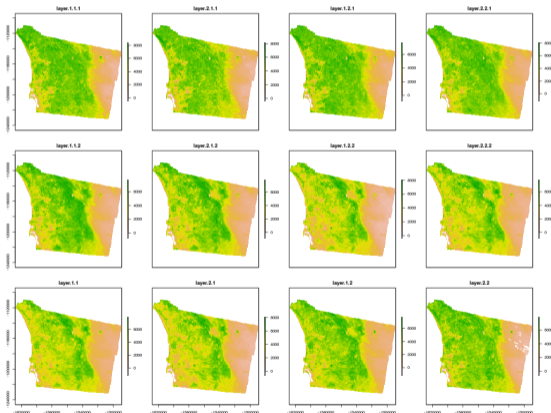
  [...]
```

# NDVI - Visualize Average by Month

```
[...]  
  
# read and plot requested raster layer  
ndvi <- brick(rastfile)  
ndvi_rast <- raster()  
for (i in 1:12)  
{  
  fmt_str <- paste(layeryr, sprintf("%02d", i), sep = "-")  
  ndvi_layer <- mean(ndvi[[which(format(ndvi_dt, "%Y-%m") == fmt_str)]], na.rm = TRUE)  
  ndvi_rast <- addLayer(ndvi_rast, ndvi_layer)  
}  
  
plot(ndvi_rast)  
  
if (plotfile != "") dev.off()  
}
```

# NDVI - Visualize Average by Month

```
plot_ndvi_year("/usr/local/bda/modis/ndvi_cooked/ndvi.tif", 2011, "/tmp/ndvi-2011.png")
```



# Questions?

Thank You!

mail@joeconway.com

joe@crunchydata.com

@josepheconway