For the OS operations we created a dedicated user (called prosnow) on the PROSNOW .
This readme file assumes every scripts are stored in and directory called from the home directory for this user and this user have sudoer privileges.

PROSNOW Central Data Server configuration

- The PROSNOW machine used to develop the database, API and demonstrator was virtual machine integrated in the INRAE hardware architecture and managed using VMWare
  - Machine harware performance: 8 cores and 60Go RAM
  - OS: Linux prosnow 4.9.0-8-amd64 #1 SMP Debian 4.9.144-3.1 (2019-02-19) x86_64
- The PROSNOW Central Data Server relies on several softwares
  - A PostgreSQL DBMS (10.7 (Debian 10.7-1.pgdg90+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 6.3.0-18+deb9u1) 6.3.0 20170516, 64-bit) with PostGIS (2.5 USE_GEOS=1 USE_PROJ=1 USE_STATS=1) extension and the R bindings using PLR (https://github.com/jconway/plr) with underlying R version 3.4.4
    - PostgreSQL modified parameters:
      - shared_buffers = 8GB # (change requires restart)
      - work_mem = 4GB
      - maintenance_work_mem = 2GB
  - Apache/2.4.25 (Debian)
  - PHP7 server

Database first setup

Database schemas

- "prosnow" schema is used for pre-processing operations (import raw data and process some observations directly pushed to the public schema without using the API)
  Some simple tables are mandatory: resorts, skitracks and snowguns

```
  psql -h localhost -p 5432 -U USER -f create_table_prosnow_schema.sql
prosnow
```

- "public" schema is used by the API and its organization is described in the PROSNOW API repository

Adding a resort

- Import skitracks using files from resort

```
ogr2ogr -f PostgreSQL PG:"dbname='prosnow' host='HOST' port='5432'
user='USER' password='PWD'" /PATH_TO_SPATIAL_DATA/GEODATAFILE -nln
"prosnow.skitracks_import_RESORTNAME" -nlt "MULTIPOLYGON
```

- Fill skitracks table

```
INSERT INTO prosnow.skitracks(resort_id, key, name, the_geom)
SELECT RESORT_ID, ORIGINAL_ID_FIELD, ORIGINAL_NAME_FIELD,
st_force2d(st_transform(wkb_geometry,3035))::geometry(multipolygon,3035)
FROM prosnow.skitracks_import_RESORTNAME;
```

- Import snowmaking shafts using files from resort

```
ogr2ogr -f PostgreSQL PG:"dbname='prosnow' host='HOST' port='5432'
user='USER' password='PWD'" /PATH_TO_SPATIAL_DATA/GEODATAFILE -nln
"prosnow.snowguns_import_RESORTNAME"
```

- Fill snowguns table

```
INSERT INTO prosnow.snowguns(resort_id, key, name, the_geom)
SELECT RESORT_ID, ORIGINAL_ID_FIELD, ORIGINAL_NAME_FIELD,
st_force2d(st_transform(wkb_geometry,3035))
FROM prosnow.snowguns_import_RESORTNAME;
```

Create the PROSNOW information and dedicated functions

Link between Skitracks and snowguns

A snowgun is assigned to a slope based on a nearest neighboor approach. Then to identify the area covered by each snowgun is computed by the PLPGSQL function prosnow. we create concentric buffers around the snowgun point with a step radius of 5 meters until the intersection area between the buffer and the ski slopes geometry reaches 3000m² (as a snowgun is supposed to cover 1/3 ha). The parameters radius step and 3000m² threshold are hard coded in the PLPGSQL function.

- PLPGSQL function creation: the function returns the corresponding buffer radius is called by the SRU creation script for each track_id, gid couple.

```
psql -h localhost -p 5432 -U USER -f function_snowgun_area.sql prosnow
```

Compute SRUs

Python prerequisites

- Setup a virtual environment for python 3.5 in the home directory and activate it to run the scripts below

```
sudo apt-get install python3.5
sudo apt-get install virtualenv
mkdir python35
virtualenv -p /usr/bin/python3.5 python35
source python35/bin/activate
```

- Install the required modules

```
pip install numpy
pip install psycopg2
```

- For the use of psycopg2 in Python script, update connection parameters:
  - Rename conn_param.example to conn_param.py
  - Edit the file and set the variables to the required values
- The install of the gdal module (for osgeo API) might be a bit tricky. The following steps could help:
  - Install the binaries using apt

```
sudo apt-get install gdal-bin
sudo apt-get install libgdal-dev
sudo apt-get install python3-gdal
sudo apt-get install python3.5-dev # Mind the python version !
```

  - Add the required environment variables

```
export CPLUS_INCLUDE_PATH=/usr/include/gdal
export C_INCLUDE_PATH=/usr/include/gdal
```

  - Install the same python module version than the binary one

```
pip install GDAL==$(gdal-config --version) --global-option=build_ext --
global-option="-I/usr/include/gdal"
```

Launch SRU script

- Before running the script some hard coded have to be changed at the end of the script get_track_sector_3.5.py
  - List the resorts for which you want to compute SRUs: line 809, the variable resort_ids is a list of strings from 1 to n elements, each representing the integer id of a resort
  - The variable src_mnt point for the DEM used by PROSNOW and is derived from EU_DEM provided by the Copernicus programm available online here: https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1. PROSNOW uses a unique Virtual Raster Tiles supposed to be store in the eudem3035 folder.

```
gdalbuildvrt /PATH_TO_DOWNLOADED_EXTRACTED_TILES/*.TIF
/PATH_TO_DIRECTORY/eudem3035.vrt
```

- Launch the script
  python get_track_sector_3.5.py
- Go get a coffee now ! If you setup gdal module for python, you really deserve a break:

Case when the SRUs are directly provided by the ski resort

- The SRUs attributes values have to be filled and this is doable using a python script. Before running the script, please, mind the hard coded parameters as for SRU script above
  - The resort_ids is a list of integer, line 112

    python compute_idealized_values.py

- The snowgun buffer size used for the distribution of water consumptions over SRUs, another python script can be used to add this information in the prosnow.snowgun_buf table
    - The resort_ids list of integer, line 81

      python update_buffer_snowguns.py

Next steps: setup API which is a prerequisite for the next steps

Now let's fill the core tables for PROSNOW in the public schema: skiresort, snowguns, skitracks and srus for the considered SKIRESORT_ID

- Add the required function

```
psql psql -h localhost -p 5432 -U USER -f create_function_add_resort.sql
prosnow
```

- Execute the add_resort function

```
SELECT prosnow.add_resort(SKIRESORT_ID)
```

Set default parameters

Once the public schema prepared for PROSNOW data and the SRU and resort have been added, it is time for some additional setup required to operate the demonstrator

- If the prosnow.default_variablemapclasses has not been created yet you'll have to run this SQL script:

```
psql -h localhost -p 5432 -U USER -f create_table_default_values.sql prosnow
```

- Once the table exists and is filled, you'll be able to add the information in the public schema:
    1. Add the default map classes for your SKIRESORT_ID

```
INSERT INTO variablemapclasses (skiresort_id, variable_id, rank,
valmin, valmax, colorhex, red, green, blue, alpha)
SELECT SKIRESORT_ID, variable_id, rank, valmin, valmax, colorhex, red,
green, blue, alpha
FROM prosnow.default_variablemapclasses
```

    2. Add the default threshold for the variables for your SKIRESORT_ID

```
INSERT INTO variablethresholds(variable_id, operator, value)
SELECT SKIRESORT_ID, variable_id, operator, value
FROM prosnow.default_variablethresholds
```

    3. Add default available configurations for your SKIRESORT_ID: by default, to be fully operationnal PROSNOW demonstrator requires configuration 1, 2, 11, and 12 but does not require any configuration with snow production but the final set of available configuration SHOULD be defined accordingly with the resort and not be more that 12

configurations includind the 4 mandatory ones

```
INSERT INTO configuration_skiresort(skiresort_id, configuration_id)
VALUES (SKIRESORT_ID, 1), (SKIRESORT_ID, 2), (SKIRESORT_ID, 11),
(SKIRESORT_ID, 12)
```

4. Add default configuration for your SKIRESORT_ID: note that, here the purpose is only to have an operationnal demonstrator but the default configuration is SRU based and SHOULD defined accordingly with the resort

```
UPDATE srus SET configuration_id = 12
WHERE skiresort_id = SKIRESORT_ID;
```

Deal with TechnoAlpin (TA) Data

SQL stuff

- To import the TA data and spread it over SRUs it is compulsory to link the TA resort code with the PROSNOW resort id (several TA codes can be assigned to a single PROSNOW resort) in the prosnow.match_ta_resort table

```
psql -h localhost -p 5432 -U USER -f create_technoalpin_tables.sql prosnow
```

- The distribution of water consumption over SRUs rely on the weighted (between 0 and 1) relationship between a snowgun and one or more SRU (the total weight for each snowgun is 1) which could be assigned manually or computed based on the SRU / buffer intersection

Get the data from TA FTP server

Using a basic Cron to retrieve data everyday at 6a.m., the import process is fully automated based on spatial information and following some simple rules. It uses a Bash script which is called wc.sh and who is supposed to be in the home directory of the user whose the crontab belong to.

- Only TA resort codes referenced in the prosnow.match_ta_resort will be imported and it is mandatory to fill the "filetype" field because the xml structure differs between Liberty and ATASS softwares.
- A TA shaft is identified using both the TA id and the TA resort code

```
crontab -e
# The crontab line should look like this
# m h  dom mon dow   command
  59 5 * * * .PROSNOW_PATH/Other_scripts/wc.sh > cron.log
```

- The Bash script activate the virtual environment and launch the python script get_wc_fromTA.py (which is also supposed to be in the home directory)
- WARNING To reset spatial information for a TA resort, it is required to delete the corresponding data in the prosnow.ta_shafts

```
DELETE FROM prosnow.ta_shafts
WHERE ta_resort = TA_RESORT
```

Plot charts

To fulfill the demonstrator requirements, two PL/R functions

Interactive SRU chart over time

- Required packages: dygraphs, xts, magrittr
- Edit file create_function_prosnow_new_sru_dyngraph.sql to set the right server url line 19: change 'demonstrator.prosnow.org' by your own server url
- Create function create_function_prosnow_new_sru_dyngraph.sql

```
psql -h localhost -p 5432 -U USER -f
create_function_prosnow_new_sru_dyngraph.sql prosnow
```

- Arguments to call the function

```
SELECT * FROM prosnow.prosnow.new_sru_dyngraph(
    sru, --integer SRU id
    config, --integer snow management configuration id
    var, --character varying DEFAULT 'sd'::character varying PROSNOW var to
be plotted
    thd, --double precision DEFAULT 0.6 Threshold for computing PROSNOW
probabilities
    nbdays --integer DEFAULT 9999 time extent for the plot (number of days)
    )
```

- Output a html file (and the required JS libraries) at the path hard coded at the end of the function (variable "path")

Probability matrix of snow management configuration

- Create function

```
psql -h localhost -p 5432 -U USER -f
create_function_prosnow_sru_probgraph.sql prosnow
```

- Arguments to call the function

```
SELECT prosnow.blob_sru_probgraph(
    sru, --integer SRU id
    var, character varying DEFAULT 'sd'::character varying PROSNOW var to be
plotted
    thd double, --precision DEFAULT 0.6 Threshold for computing var PROSNOW
probabilities
    thd_wbt --double precision DEFAULT '-4'::integer Threshold for computing
wbt var PROSNOW probabilities
    )
```

- Output a blob file at the path hard coded in the variable "file"

Contributing

This code has been developped between 2017 and 2020 by Hugues François (INRAE Grenoble - LESSEM, Université Grenoble Alpes, hugues.francois@inrae.fr) with the support of Frédéric Bray (INRAE Grenoble - LESSEM, Université Grenoble Alpes, frederic.bray@inrae.fr) and Carlo Carmagnola (CNRM-CEN - UMR Météo-France/CNRS, carlo.carmagnola@meteo.fr)

Licence

This code is limited at the prosnow consortium. It will be public in 2023