# VectorPoint - Programmer's Guide

**2018-03-14**

**Table of Contents**

# Introduction

This guide supports the paper on VectorPoint (VP) epidemiological surveillance software. VP is a suite for spatial surveillance written in the R programming language using the open-source *shiny* library[1].

The components of VP are:

- Database
- R/Shiny UI
- Risk prediction model

## Terminology

- Site or premises: a place (often a residence) with latitude and longitude
- UNICODE: an identifier for a site

[1] Version 1.0.5 © 2017 Rstudio, Inc. http://shinyapps.io

# Main functionality

## Field data collection

In this mode, workers in the field

1. Log in into the software
2. Navigate and explore the risk map
3. Select a site and enter inspection results

## Simulated search

This mode of operation is similar to field data mode, except that any inspection data is marked as "TEST_DATA" in the database. This mode is designed for testing and training purposes, as well as for observation of app usage.

## Read-only operation

Users could be configured as read-only users, in which case any attempt to enter data will be refused.

## Working with Risk maps

Risk maps are tables in the database. They are created separately from the main app, and could be computing using a variety of models.

A risk map need to be uploaded to the database into any table.

## Configuration

Configuration of the app is performed by modifying:

- SEARCH_USERS table
- Database configuration of *global.R*

# Database structure

The database can be any MySQL database or compatible.

APP_INSPECTIONS:

The table for storing the results of inspections. Each row reports on a single inspection of a site. The row records the date, the results of the inspection, and the site identifier (UNICODE). A can have one or more inspections by different users.

Some of the key columns in this table are:

| **Column** | **Meaning** | **Type** | **Example** |
|---|---|---|---|
| UNICODE | A unique ID for each site | Mandatory string | 10.12.44.11 |
| USER_NAME | The inspector | Automatically generated | johnD531 |
| GROUP_NAME | The inspector's group | Automatically generated | inspectors2017 |
| STATUS_INSPECCION | Categorical status code* | Entered by inspector | ENT |
| TEST_DATA | 1 (simulation) 0 (true field inspection) | Binary | 1 |
| DATETIME | Time in UTC | Automatically generated | 2017-01-11 19:57:56 |
| PREDICTED_PROBAB | In [0,1] the probability according to the model at the time of inspection | Automatically generated | 0.123 |

*STATUS_INSPECCION is usually one of the following:

- I or inspection = inspection completed,
- ENT or entrevista = interview of the occupants without inspection,
- C=closed (inspection not possible),
- R = occupant refused inspection,
- V = inspector should return (volver) to this site,
- LP = lot public,
- LV = lote vacant,
- D = uninhabited (deshabitada)

Many of the data fields differentiate between entomological findings in the house (intradomestic) and around the house (peri-domestic).

Sites recorded as “V” will be listed when the user clicks on the “Information” button in the floating menu.

SEARCH_USERS:
A table linking a user to a unique risk group. The user_id is confidential and should include a numeric code that would make guessing difficult. The user_id is used in the field by users to access the system. The same person could have multiple user IDs in order to view different datasets.

MODEL_OUTPUT tables:
One or more tables containing the data to be visualized.

It can have any legal table name, such as MODEL_RESULT_2017_12_01_AREA_2.

It is allowed and recommended to combine several models into a database VIEW.

These are the columns in this table:

| **Column** | **Meaning** | **Type** | **Example** |
|---|---|---|---|
| UNICODE | A unique ID for each site | Mandatory string | 10.12.44.11 |
| probability | In [0,1] the risk | Mandatory double | 0.34 |
| uncertainty | In [0,1] the model uncertainty | Mandatory double | 0.1 |
| locCode | Area where the site is located | Mandatory string | 10.12.44 |
| latitude / longitude | latitude and longitude in degrees | Mandatory double | -16.231, -71.332 |
| time | Years since the last inspection | Optional numeric | 4 |
| recentinspec | Date of most recent known inspection (when not in VectorPoint) | Optional date | 2014-02-03 |

CACHE:

We provide two functions to cache data, for cases there the app is disconnected from the cloud.

1. Browser's cache: When the app is disconnected, the app is able to store surveillance fields in the browser's cache.
2. Database CACHE table: A table for caching between log-ins. It is keyed by USER_NAME. The data format is either TEXT or BLOB. It was previously used for caching inspection results in cases of disconnection, but is no longer used.

# Limitations and Future Work

UNICODE

The app makes a heavy use of the UNICODE location system to identify sites. In the future, and in other surveillance zones, sites could be identified using any arbitrary system. Similarly, the locCode neighborhood identifier should be arbitrary, rather than being derived from the UNICODE.

Spanish language

The User Interface (UI) uses Spanish, but this should be made configurable.

Caching and connectivity

The app is dependent on access to the cloud shinyapps.io and the database. If a session is interrupted during data entry, the app resets. A partial solution relies on the user manually sending the data to cache - it is then shown when the user logs in again. It is desirable for the app to gracefully wait for the connection to be restored, and this functionality might become available in future versions of Shiny.

GPS and location

When the app is running, location information is obtained from the browser and drawn on the map. The accuracy of this method is not always high, and might be affected by interference and poor smartphone hardware. The user must, therefore, verify his location using visual cues and the street layout.