Demonstrate Package

```
# The try-catch is here because the cluster's R image lacks git.
repo_name <- "unknown"
tryCatch({
        repo_name <- system("git config --get remote.origin.url", intern = TRUE)
}, error = function(x) {}
)

# The R on the cluster is also an older version that won't
# execute rampdata::data_configuration() in the preamble above.
# You can define it yourself, if that's the case.
params2 = list(data_config = rampdata::data_configuration())
```

From git@github.com:dd-harp/rampdata.git on Mon Mar 30 13:08:14 2020, generated by adolgert.

Overview

This notebook demonstrates the `rampdata` package. It defines the data configuration in a header. This data configuration can then help find data within code.

```
cat(params$data_config$LOCALDATA)
```

```
## /home/adolgert/data
```

Local Path

The `local_path` command translates a keyword list into a path that's relative to the data directory on your computer.

```
shapefile <- list(location = "uganda", stage = "input", path = "UBOS/outline.shp")
rampdata::local_path(shapefile, params$data_config)
```

```
## /home/adolgert/data/inputs/locations/uganda/UBOS/outline.shp
```

```
edited_shapefile <- list(stage = "working", path = "UBOS/outline_fixed.shp",
                         project = "rampdata")
rampdata::local_path(edited_shapefile, params$data_config)
```

```
## /home/adolgert/data/projects/rampdata/users/adolgert/UBOS/outline_fixed.shp
```

Data configuration

That local path uses a configuration object that's initialized in the metadata section of this notebook.

```
params:
  data_config: !r rampdata::data_configuration()
```

When the notebook is built, R runs the `data_configuration` function in order to find the location of the data on your machine. It reads this configuration information from a `data.ini` file. Look at the help for `data_configuration` for an example of the `data.ini` and what to put in it.

We use the params because it is a portable way to set parameters. For instance, if you want to render a document from the command line, you can set them from there.

```
R -e "rmarkdown::render('ramp_example.Rmd', params = list(data_config = '.'))"
```

This method works for running single functions, too, so instead of running a script, we run a function from the package, through R.

```
R -e "gisdemog::generate_population()"
```