Riskmap Pipeline

Here I describe the process of moving from xGenESeSS models to heatmaps. I will include some example files as we go along.

Starting out: We start out (for the terror dataset) with 230 xgen models. Example: see file '0model.json'

Step 1: Apply cynet. We apply cynet to each xgen model to get cynet log files. We use the function map_events_parallel for this. See attached cynet.py file.

```
models_files = glob.glob(glob_path)
models_files = [m.split('.')[0] for m in models_files]

CYNET_PATH = os.path.dirname(sys.modules['cynet'].__file__) + '/bin/cynet'
FLEXROC_PATH = os.path.dirname(sys.modules['cynet'].__file__) + '/bin/flexroc'

args = []
for model in models_files:
    for num in model_nums:
        LOG_PATH = model + 'use_{}models.log'.format(num)
        outfile = model + 'use_{}models.csv'.format(num)
        args.append([model, num, horizon, DATA_PATH, RUNLEN, VARNAME, RESSUFIX, \
        CYNET_PATH, FLEXROC_PATH,
        LOG_PATH, #Here onwards are the run parameters of the pipeline.
        PARTITION,
        DATA_TYPE,
        FLEXWIDTH,
        FLEX_TAIL_LEN,
        POSITIVE_CLASS_COLUMN,
        EVENTCOL,
        tpr_threshold,
        fpr_threshold,
        coord_col,
        day_col,
        NEGATIVE_CLASS_COLUMN,
        header,
        positive_str,
        outfile])
Parallel(n_jobs = cores, verbose = 1, backend = 'threading')\
(map(delayed(single_map), args))
```

Snippet from map_events_parallel

Note that the above model_nums are: [10,15,20,25]. This function calls the function single_map in parallel which calls cynet and flexroc.

```
for varname in VARNAME:
    stored_model=FILE+'_sel_'+str(uuid.uuid4())+'.json'

    M=uNetworkModels(FILE + '.json')
    M.setVarname()
    M.augmentDistance()

    if varname is not 'ALL':
        M.select(var='src_var',equal=varname,inplace=True)

    M.select(var='delay',inplace=True,low=Horizon)
    M.select(var='distance',n=model_nums,store=stored_model,reverse=False,inplace=True)

    if M.models:
        simulation = simulateModel(stored_model, DATA_PATH, RUNLEN, CYNET_PATH=CYNET_PATH,FLEXROC_PATH=FLEXROC_PATH)
        simulation.single_cynet(LOG_PATH=LOG_PATH, DATA_TYPE=DATA_TYPE, PARTITION=PARTITION)
        simulation.parse_cynet(LOG_PATH,
                               tpr=tpr_threshold,
                               fpr=fpr_threshold,
                               FLEXWIDTH=FLEXWIDTH,
                               FLEX_TAIL_LEN=FLEX_TAIL_LEN,
                               coord_col=coord_col,
                               day_col=day_col,
                               EVENTCOL=EVENTCOL,
                               NEGATIVE_CLASS_COLUMN=NEGATIVE_CLASS_COLUMN,
                               POSITIVE_CLASS_COLUMN=POSITIVE_CLASS_COLUMN,
                               header=header,
                               positive_str=positive_str,
                               outfile=outfile)
```

Snippet from single_cynet function.

It is very important to note that VARNAME is set to [‘ALL’]. We have always done it this way. VARNAME is a variable that I have not changed, ever.

single_cynet(*args) applies cynet once.

Now since we have 4 model nums (10,15,20,25) and 1 VARNAME (‘ALL’) there are only **FOUR** log files created per model. We started with 230 xgen models, thus we have 920 cynet log files.
Examples: 0modeluse_10models.log, 0modeluse_15models.log, 0modeluse_20models, 0modeluse_25models.log.

0modeluse_15models.log denotes cynet applied to the 0th xgen model with model nums set to 15.

Step 2: This steps starts with the log files being produced by cynet. The parse_cynet function calls flexroc on the logfiles, and obtains the threshold needed. Currently we have have flexroc getting the threshold such that tpr=0.85.

Step 3: Parse cynet then parses the cynet logfile into a csv. It then applies the threshold to the last column (the positive events column) to get a 'predictions' column. Then outputs these csvs.

Check: Thus far we have 230 xgen model.json files, 920 cynet log files, and 920 cynet csv files. Example of csvs: 0modeluse_10models.csv, 0modeluse_15models.csv, 0modeluse_20models.csv, 0modeluse_25models.csv

Step 4: Combine the csvs into one file. This is very simple. I took all the _20models.csv files for this combine. Hence These are only the cynet log files with model number set to 20. Created is the file '20models.csv'. See example.

I note that in this file, there variables present are these three: 'Armed_Assault-Hostage_Taking_Barricade_Incident-Hijacking-Assassination-Hostage_Taking_Kidnapping_', 'VAR', 'Bombing_Explosion-Facility_Infrastructure_Attack'

Step 5: Generate heatmaps. We use the utilities in the file carto.py for this. I will note that the types argument allows us to select for the variables above. In the current maps, I set the types to a list of all three of the above. See picture below. Dataframecsv is '20models.csv' produced in the last step. This will create one riskmap for one day.

```
def draw_riskmap(day, dataframecsv, types,title,radius,outfile,detail=0.25, grace=1,day_col='day',
                 actual_event_col='actual_event',variable_col='variable',predictin_col='predictions',lon_col = 'lon',lat_col='lat'):
    '''
    Draws a riskmap for the selected day.
    '''
    days = list(range(day - grace, day + grace + 1))

    df = pd.read_csv(dataframecsv)
    df = df[df[day_col].isin(days)]
    df = df[df[variable_col].isin(types)]

    preddf = df[df[predictin_col] == 1]

    true_df = df[df[day_col] == day]
    true_df = true_df[true_df[actual_event_col] == 1]

    true_lon = list(true_df[lon_col])
    true_lat = list(true_df[lat_col])

    lons = list(preddf[lon_col])
    lats = list(preddf[lat_col])

    lon_min = min(lons)
    lon_max = max(lons)
    lat_min = min(lats)
    lat_max = max(lats)

    lon_grid=np.arange(lon_min-radius,lon_max+radius,detail)
    lat_grid=np.arange(lat_min-radius,lat_max+radius,detail)
    lon_mesh,lat_mesh=np.meshgrid(lon_grid,lat_grid)
```