EPG2 - User Guide

Phil Symonds, Jon Taylor, Phill Biddulph

May 4, 2016

EPG2 allows for the individual or batch generation of EnergyPlus .idf files. If you get stuck, please email: p.symonds@ucl.ac.uk. The code is being continually improved and developed so please check the EPG2 Dropbox folder for updates.

1 Installation

EPG2 requires Python version 3.4 with the following packages installed:

- scipy
- numpy
- matplotlib
- pyDOE

To get EPG2, copy all the files over from the EPG2 Dropbox folder to a sensible location on your computer. You may need to edit the epg2.py file such that the paths at the top point to the EnergyPlus folder and the folder where EPG2 is installed.

1.1 Windows

On Windows you can download a version of Python that come with most of the above packages pre-installed e.g. Anaconda or Spyder. You will then just have to manually install any of the additional packages from above that are not already included.

You will also have to add the path of EPG2 to the python path. You can do this by editing and adding the setup_python.py (as is in the EPG2 Dropbox folder) file to .ipython/profile_default/startup/

1.2 Mac/Linux

Python comes pre-installed on Unix systems. You can use pip to get the latest version of the packages you require. You just need to open a terminal and then do:

```
pip3 install scipy
pip3 install numpy
pip3 install matplotlib
pip3 install --upgrade pyDOE
```

2 Running EPG2

There are three ways in which EPG2 can be run:

1. Standard: Uses a <script>.csv file to produce idfs defined in the csv file
2. Factorial Design: uses a <script>.txt file to run over different variants
3. Latin Hypercube: input parameters are randomised according to pre-defined distributions and ranges

For all of these methods we start by opening a terminal. On Windows you need to open the Anaconda console from the start menu. On a MAC/Linux system you have the standard terminal and you need to change directory (`cd`) to where the epg2.py file is located.

In the terminal, you can then use the command:

```
ipython3 --pylab
```

which starts an interactive python session. Then you need to import the epg2 module:

```
import epg2
```

The next line you enter depends on how you want to run the code e.g. 1, 2 or 3 from above.

1. For standard running using the .csv file as input you do the following:

```
epg2.unpick_csv(indir='Debug_runs',csvfile='script.csv',batfile=False,updatecsv=True)
```

where you can change some of the options. `indir` is the location of your project where you want the .idf files to be produced and also where your .csv file that you are reading is located. `csvfile` is the name of your csv file (more details on what variant to enter into the csv file are in section 3.1). Set `batfile` to `True` if you want to save scripts than can later be run on Legion. Change `updatecsv` to `False` if you don't want the csv file to be updated. A full example of how to do this is given in section 3.1.

2. For factorial type mass production of .idf files do:

```
epg2.unpick_script(indir='Debug_runs',script='script.txt',build=True,batfile=False)
```

where this time the *script.txt* needs to be specified as that in your project (`indir`) folder where you want your .idf files to be produced. Running the above command produces a .csv file which has information on the runs that were produced. It is possible to produce the .csv file without building the .idf files by changing `build` to `False`.

3. To run a latin hypercube design do:

```
epg2.run_hypercube(indir='Debug_runs',build=True,batfile=True)
```

No script or csv file is required for this method. You will have to dive into the code (epg2.py, lines 41-236) to change the number of .idfs to produce and the ranges and distributions of input parameters. The options are the same as for the previous two methods. Running this produces a .csv file with the built forms that have been produced.

2.1 Drawing a built form

EPG2 also has the option to produce a drawing of a built form by doing:

```
epg2.draw_building(buildname='Detached',xcut=3,ycut=3,zcut=5.5,shadingname='Blank')
```

where `buildname` is the name of the built form (this must be one of the built forms in the data_files/build_props.csv). What is drawn depends on `xcut`, `ycut` and `zcut`. These values should be chosen such that the intersect with the level of the building that you want drawn (e.g. `zcut=1` would show the rooms on the ground floor). If you want the shading of the building to be drawn you can set `shadingname` to something (this must be a one of the types in data_files/shade_props.csv).

2.2 Find out the u-value of a fabric

EPG2 also allows you to quickly find out the u-value of a fabric by doing:

```
epg2.calc_fab_u(fab_name='0.45_External_Wall')
```

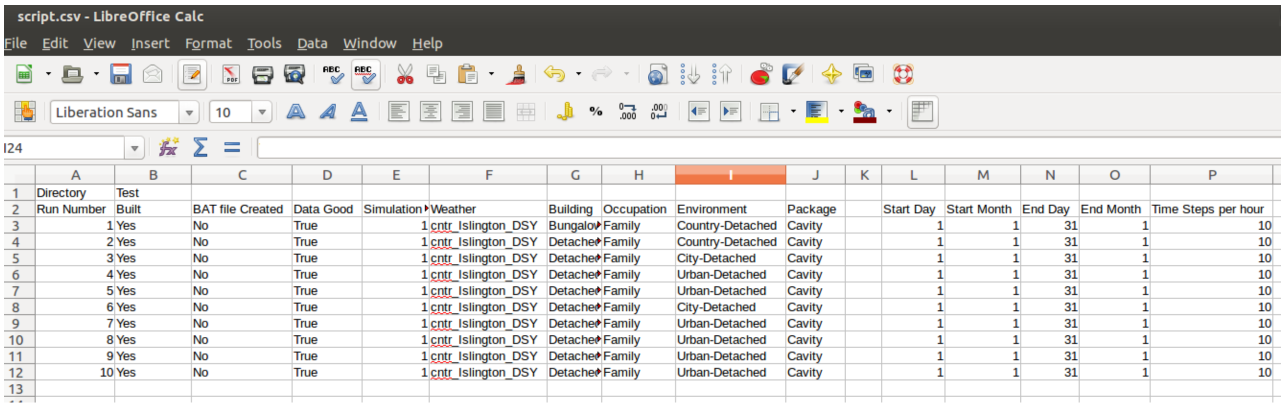
where the `fab_name` that you choose must be one from the data_files/fab_props.csv file.

3 Examples

3.1 Example 1: How to create a built form using a <*script*>*.csv*

This section gives a step-by-step example of how you can create an .idf file for Semi-detached house with pensioner occupants using the standard procedure (1).

1. First of all, create a new directory within the EPG2 folder called "Test".
2. Copy the script.csv from the Debug_Runs folder to the "Test" folder that you just created.
3. Open the Test/script.csv file in excel. You will notice that there are quite alot of options to choose from in the columns of this file (see Figures 1 and 2).



script.csv - LibreOffice Calc
File Edit View Insert Format Tools Data Window Help
Liberation Sans
10
I24

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Directory	Test														
2	Run Number	Built	BAT file Created	Data Good	Simulation	Weather	Building	Occupation	Environment	Package		Start Day	Start Month	End Day	End Month	Time Steps per hour
3	1	Yes	No	True	1	cntr_Islington_DSY	Bungalow	Family	Country-Detached	Cavity		1	1	31	1	10
4	2	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	Country-Detached	Cavity		1	1	31	1	10
5	3	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	City-Detached	Cavity		1	1	31	1	10
6	4	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	Urban-Detached	Cavity		1	1	31	1	10
7	5	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	Urban-Detached	Cavity		1	1	31	1	10
8	6	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	City-Detached	Cavity		1	1	31	1	10
9	7	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	Urban-Detached	Cavity		1	1	31	1	10
10	8	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	Urban-Detached	Cavity		1	1	31	1	10
11	9	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	Urban-Detached	Cavity		1	1	31	1	10
12	10	Yes	No	True	1	cntr_Islington_DSY	Detache	Family	Urban-Detached	Cavity		1	1	31	1	10
13																

Figure 1: Columns A to P of a csv file.

Here I will run through the A-Z of the various columns:

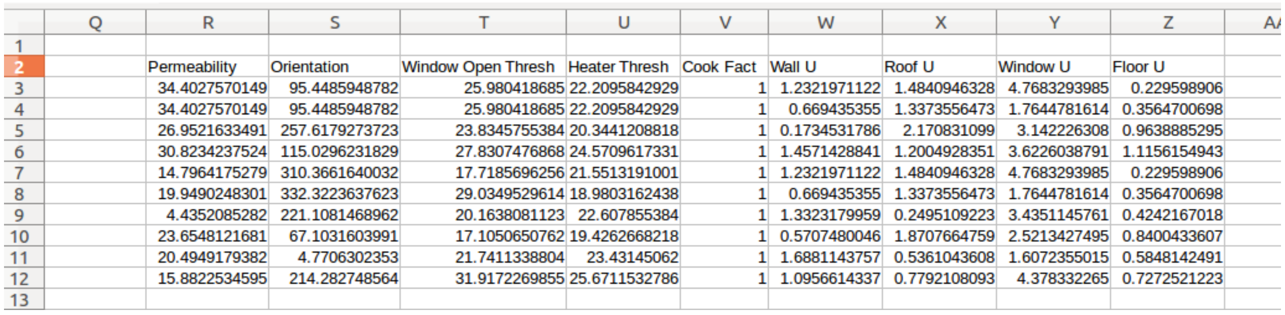
(A) *Run Number* - acts as a unique identifier for the idf being built

(B) *Built* - is whether or not you want the .idf file to be created. must be either Yes or No

(C) *BAT file Created* - is whether or not you want to produce scripts which can be run on Legion. Is again Yes or No

(D) *Data Good* - will get updated to True if EPG2 has worked

(E) *Simulation Level* - options are 1 for Default or 3 for HAMT. Using HAMT takes a lot longer when running EnergyPlus



	Q	R	S	T	U	V	W	X	Y	Z	AA
1											
2		Permeability	Orientation	Window Open Thresh	Heater Thresh	Cook Fact	Wall U	Roof U	Window U	Floor U	
3		34.4027570149	95.4485948782	25.980418685	22.2095842929	1	1.2321971122	1.4840946328	4.7683293985	0.229598906	
4		34.4027570149	95.4485948782	25.980418685	22.2095842929	1	0.669435355	1.3373556473	1.7644781614	0.3564700698	
5		26.9521633491	257.6179273723	23.8345755384	20.3441208818	1	0.1734531786	2.170831099	3.142226308	0.9638885295	
6		30.8234237524	115.0296231829	27.8307476868	24.5709617331	1	1.4571428841	1.2004928351	3.6226038791	1.1156154943	
7		14.7964175279	310.3661640032	17.7185696256	21.5513191001	1	1.2321971122	1.4840946328	4.7683293985	0.229598906	
8		19.9490248301	332.3223637623	29.0349529614	18.9803162438	1	0.669435355	1.3373556473	1.7644781614	0.3564700698	
9		4.4352085282	221.1081468962	20.1638081123	22.607855384	1	1.3323179959	0.2495109223	3.4351145761	0.4242167018	
10		23.6548121681	67.1031603991	17.1050650762	19.4262668218	1	0.5707480046	1.8707664759	2.5213427495	0.8400433607	
11		20.4949179382	4.7706302353	21.7411338804	23.43145062	1	1.6881143757	0.5361043608	1.6072355015	0.5848142491	
12		15.8822534595	214.282748564	31.9172269855	25.6711532786	1	1.0956614337	0.7792108093	4.378332265	0.7272521223	
13											

Figure 2: Columns R to Z of a csv file.

(F) *Weather* - Weather file being used

(G) *Building* - built form which must be chosen from the first column of data_files/build_props.csv (more on this in section 4)

(H) *Occupation* - the occupancy type which must be chosen from the first column of data_files/occupation_props.csv (more on this in section 4)

(I) *Environment* - this column is where the wind exposure (e.g. City, Urban or Country) and shading (e.g. Detached, Semi, Terrace) is defined. The options available are in the env_props.csv

(J) *Package* - is where the properties of the built form are chosen in terms of the windows walls, floor etc...

(K) Leave blank

(L) *Start Day* - simulation start day

(M) *Start Month* - simulation start month

(N) *End Day* - simulation end day

(O) *End Month* - simulation end month

(P) *Time Steps per hour* - number of time steps per hour to simulate

(Q) Leave blank

(R) *Permeability* - permeability of the building in m^3/m^2h @ 50 pa

(S) *Orientation* - East of North in degrees

(T) Window Open Thresh - Temperature at which windows are opened in the summer. Note: This only works if the Window_temperature schedule is being called in occupation_props.csv taken from the house_scheds.csv file. The temperature (column L in house_scheds.csv) must also be set to 20 (perhaps I should change this - feedback welcome)

(U) *Heater Thresh* - Thermostat set temperature during the winter

(V) *Cook Fact* - Represents the a factor to scale the amount of PM2.5 produced in cooking. Note: Only used if a PM2.5 cooking schedule is being called in the occupation_props.csv

(W) *Wall U* - U-value of external walls. Note - Only works if the CAVITY_RandomU or SOLID_BRICK_RandomU fabrics (in fab_props.csv) is called in the pack_props.csv file. Make sure value is in suitable range. If set to 0 the default for the fabric is used. Otherwise the thickness of the air gap is varied to match the chosen u-value. The same is true for the other u-value columns X-Z.

(X) *Roof U* - U-value of the roof (Attic insulation) - Note: Only works if the LoftFloor_RandomU and LoftCeiling_RandomU fabrics (in fab_props.csv) are called in the pack_props.csv file

(Y) *Window U* - U-value of the windows - Note: Only works if the Window_RandomU fabric (in fab_props.csv) is called in the pack_props.csv file

(Z) *Floor U* - U-value of the floor - Note: Only works if the FLOOR_RandomU fabric (in fab_props.csv) is called in the pack_props.csv file

(AA) *Glaz Fract* - Glazing fraction. Overrides the values in build_props.csv unless set to 0

(AB) *Floor Height* - Overrides the ceiling height values in build_props.csv unless set to 0

(AC) *Area Fact(Area)* - scales floor area up/down by factor provided. Uses default from build_props.csv if set to 0

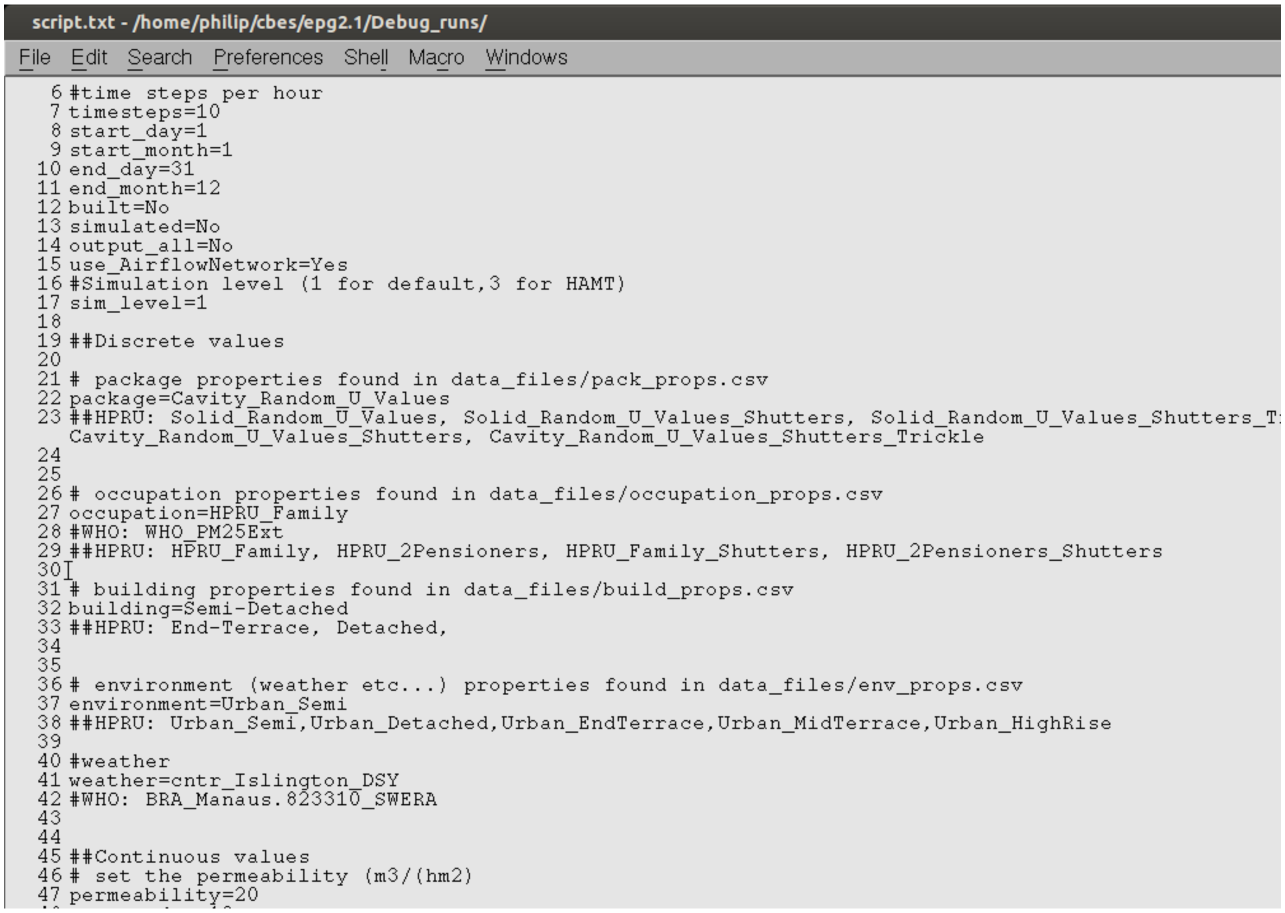
(AD) *Gains Fact* - scales the wattage of the equipment in equpment_props.csv. Uses defaults if set to 0

4. In row 3 change the "Building" to "Semi" (one of the pre-made built forms in the build_props.csv).
5. Change the "Occupation" to "Pensioners" (one of the pre-made occupancies in the occ_props.csv)
6. Change the "Enivironment" to "Urban-Semi" (one of the pre-made environments in the env_props.csv).
7. Finally, in order for an idf to be produced you need to change "Built" to "No" in row 3.
8. You can then go ahead and run EPG2 using the standard method described in 2. This will produce an file called EPG2_1.idf in the "Test" folder

3.2 Example 2: How to create a set of idfs using a factorial design with a <*script*>*.txt*

This section gives a step-by-step guide on how to produce idfs using a factorial method where one idf is created for each building or occupancy variant.

1. First of all, create a new directory within the EPG2 folder called "Factorial_Test" and copy the script.txt file from the Debug_Runs folder into it
2. Open the script.txt using your favourite text editor. It will look like Figure 3. In it's current form it is set up to produce one .idf file


script.txt - /home/philip/cbes/epg2.1/Debug_runs/
File Edit Search Preferences Shell Macro Windows

```
 6 #time steps per hour
 7 timesteps=10
 8 start_day=1
 9 start_month=1
10 end_day=31
11 end_month=12
12 built=No
13 simulated=No
14 output_all=No
15 use_AirflowNetwork=Yes
16 #Simulation level (1 for default,3 for HAMT)
17 sim_level=1
18
19 ##Discrete values
20
21 # package properties found in data_files/pack_props.csv
22 package=Cavity_Random_U_Values
23 ##HPRU: Solid_Random_U_Values, Solid_Random_U_Values_Shutters, Solid_Random_U_Values_Shutters_T
   Cavity_Random_U_Values_Shutters, Cavity_Random_U_Values_Shutters_Trickle
24
25
26 # occupation properties found in data_files/occupation_props.csv
27 occupation=HPRU_Family
28 #WHO: WHO_PM25Ext
29 ##HPRU: HPRU_Family, HPRU_2Pensioners, HPRU_Family_Shutters, HPRU_2Pensioners_Shutters
30
31 # building properties found in data_files/build_props.csv
32 building=Semi-Detached
33 ##HPRU: End-Terrace, Detached,
34
35
36 # environment (weather etc...) properties found in data_files/env_props.csv
37 environment=Urban_Semi
38 ##HPRU: Urban_Semi,Urban_Detached,Urban_EndTerrace,Urban_MidTerrace,Urban_HighRise
39
40 #weather
41 weather=cntr_Islington_DSY
42 #WHO: BRA_Manaus.823310_SWERA
43
44
45 ##Continuous values
46 # set the permeability (m3/(hm2)
47 permeability=20
```

Figure 3: The script.txt file.

3. To enable it to produce multiple idf files running over different categorical variants you need to append them to a line, e.g. change lines 27 and 32 to:

```
occupation=Family,Pensioners
building=Semi,End-Terrace,Detached
```

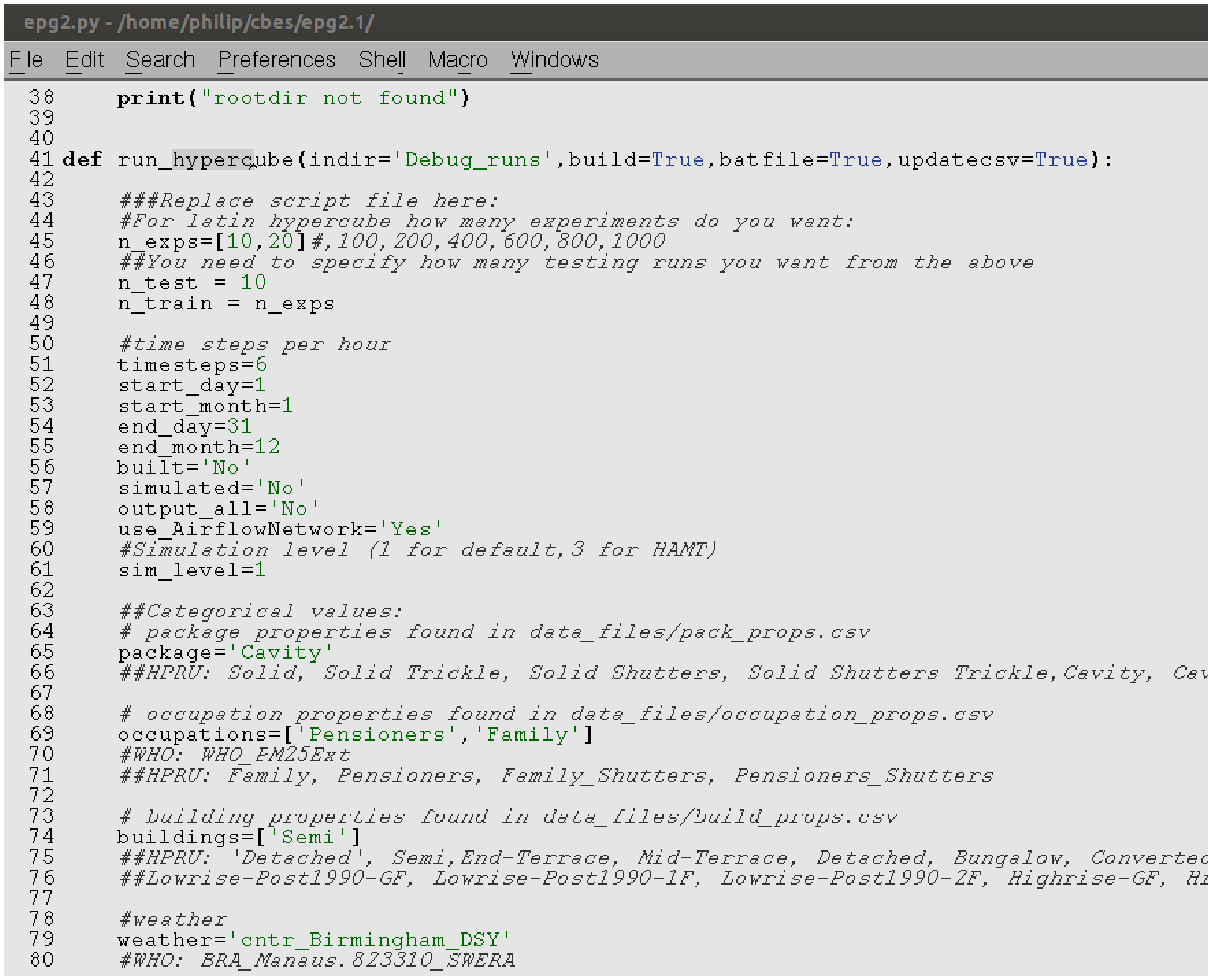
This can be done for any of the variants in the script. Note - you need to make sure that the buildings and occupancies are defined in build_props.csv and occupation_props.csv, respectively.

4. You can then run using method 2 described in section 2. This will produce 6 idf files as well as a script.csv file containing the details of these runs in your “Factorial_Test” folder.

3.3 Example 3: How to create a set of idfs using a latin hypercube design

This is for advanced users only and you will need to edit some of the EPG2 code in order for it suite your specific needs.

1. Open the epg2.py file using your favorite text editor and navigate to the `run_hypercube` function. 2. You can see that the data that was input using methods 1 and 2 are defined as variables within the code in Figure 4. This can be edited by the user. You will need to set the number of latin hypercube experiments you want in line 45. There are currently two numbers. One for training and one for testing a metamodel.


epg2.py - /home/philip/cbes/epg2.1/
File Edit Search Preferences Shell Macro Windows

```
38     print("rootdir not found")
39
40
41 def run_hypercube(indir='Debug_runs',build=True,batfile=True,updatecsv=True):
42
43     ###Replace script file here:
44     #For latin hypercube how many experiments do you want:
45     n_exps=[10,20]#,100,200,400,600,800,1000
46     ##You need to specify how many testing runs you want from the above
47     n_test = 10
48     n_train = n_exps
49
50     #time steps per hour
51     timesteps=6
52     start_day=1
53     start_month=1
54     end_day=31
55     end_month=12
56     built='No'
57     simulated='No'
58     output_all='No'
59     use_AirflowNetwork='Yes'
60     #Simulation level (1 for default,3 for HAMT)
61     sim_level=1
62
63     ##Categorical values:
64     # package properties found in data_files/pack_props.csv
65     package='Cavity'
66     ##HPRU: Solid, Solid-Trickle, Solid-Shutters, Solid-Shutters-Trickle,Cavity, Cav
67
68     # occupation properties found in data_files/occupation_props.csv
69     occupations=['Pensioners','Family']
70     #WHO: WHO_PM25Ext
71     ##HPRU: Family, Pensioners, Family_Shutters, Pensioners_Shutters
72
73     # building properties found in data_files/build_props.csv
74     buildings=['Semi']
75     ##HPRU: 'Detached', Semi,End-Terrace, Mid-Terrace, Detached, Bungalow, Convertec
76     ##Lowrise-Post1990-GF, Lowrise-Post1990-1F, Lowrise-Post1990-2F, Highrise-GF, Hi
77
78     #weather
79     weather='cntr_Birmingham_DSY'
80     #WHO: BRA_Manaus.823310_SWERA
```

Figure 4: The the `run_hypercube` method in the epg2.py file.

3. At lines 104 to 107 there are a set of `for` loops over the number of experiments, the building type and the occupancy types defined earlier in the code (these can also be changed).

4. Additional for loops can be added over other variants if required. Just copy what has been done for buildings and occupants.

4 Data files

The .csv files in the data_files folder are a key component of EPG2 and are what are used to define the properties of the building, it's environment and the occupants within the building. The current files contain all of the built forms required for the AWESOME and HPRU work. Custom built forms and schedules can be added by the user. This section describes a bit about how this can be done. The files are set up such that there are some dependencies between files and the names of somethings within files are required to match those in other files. An example of this is that fabrics within the fab_props file are used in pack_props. Figure 5 shows the dependency structure of the files.

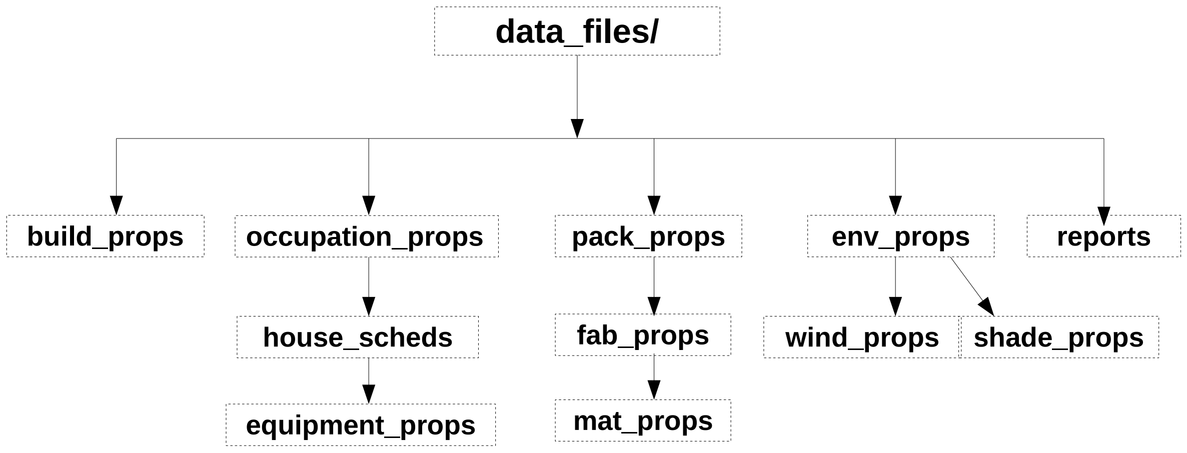


Figure 5: The structure of the EPG2 data files.

The following briefly describes what information these files contain:

build_props - defines the properties of the building in terms of geometry and structure and whether or not they have partition walls or joined zones

occupation_props - defines which occupants are present and calls on occupant behaviour such as heating, lighting and window opening in house_scheds

house_scheds - contains the schedules of the occupants in terms of heating, lighting, equipment, metabolism etc...

equipment_props - information on the equipment being used by the occupants e.g their wattage (not edited very often)

pack_props - has information on the building such as the walls, window, floor, roof. It calls on things defined in fab_props

fab_props - defines the materials that make up the walls, windows, floors and the thickness of these materials. The materials must be defined in mat_props

mat_props - defines properties of the materials such conductivity, latent heat etc. . .

env_props - defines the shading environment (from shade_props) and whether the house is in an urban, city, or country location

shade_props - defines the solar shading of the building in terms of the geometry of shading objects

wind_props - contains wind coefficients by incident angle to the building (not edited much - perhaps needs updating)

reports - defines what is output when EnergyPlus is run in terms of variables and zones