This file runs in 8 hours using 5 cores at 2-3GhZ with 8GB RAM. %config Completer.use_jedi = False # Disable jedi autocompleter to fix autocomplete issues. This may not be necessary with newer versions

In [1]:
```
# import libraries
import numpy as np
import os
import random as rand
import seaborn as sbn
from matplotlib import pyplot as plt
import pandas as pd
from multiprocess import Pool, cpu_count
```

In [2]:
```
import modelfn
```

Step0:

Input data and define Metric and Fit Objective Functions

uncomment this block to run the simulation anew. - This is not necessary to reproduce results, prerun results are provided. def fitmeta(ff): """Input fitflag options:'0','1','2','CA','NC','DP','DPNC','CADP','CANC'. Output is a best-fit parameters dataframe""" # -- Import Libraries -- from fitdata import fitdata import pandas as pd # define a function to Normalize the data by loading control and by the drug-free readout # define a function to Normalize the data by loading control and by the drug-free readout def datanorm(data): """Inputs data for a list of drugs organized by [[pMEKdata],[pERKdata],[MEKdata]]. The pMEK and pERK data for each drug are normalized by MEK (loading control) and then each of the two rows==normalized by first element (drug=0).""" normdata0=[[row[0][iel]/row[-1][iel] for iel in range(len(row[0]))] for row in data] normdata=[[row[iel]/row[0] for iel in range(len(row)) if iel > 0] for row in normdata0] return normdata drugnames= ['AZ','TAK','LY','SB','GDC','DAB','AZ-VEM','VEM','PB'] drugvals=[0, 0.01, 0.03, 0.1, 0.3, 1, 3, 10] # micro-molar ## DATA==INPUT for a list of drugs arranged as [[pMEK],[pERK],[MEK]] data for each drug with each element corresponding to above concentrations. rawinpdata=[[[25, 55, 37, 40, 27, 17, 13, 10], [73, 119, 105, 92, 83, 50,37, 10], [34, 41, 29, 33, 32, 37, 37, 39]], [[22, 38, 49, 51, 45, 41, 25, 11], [90, 112, 139, 133, 150, 110, 58, 4], [27, 31,32, 33, 30, 31, 32, 35]], [[26, 41, 42, 39, 21, 12, 7,4], [105, 182, 159, 182, 97, 63, 13, 0.062], [59, 57, 51, 59, 53, 50, 50, 51]],[[17, 79, 83, 72, 49, 18, 5, 2], [89, 185, 182, 187, 166, 103, 15, 0.5], [32, 30, 33, 33, 32, 30, 30, 29]],[[7, 42, 63, 80, 73, 55, 23, 11], [71, 161, 176, 162, 176, 165, 118, 33], [43, 45, 40, 44, 44, 47, 47, 53]],[[25, 38, 64, 77, 84, 72, 72, 34], [110, 132, 158, 133, 161, 134, 160, 118], [51, 56, 61, 57, 55, 45, 51, 46]],[[14, 24, 31, 56, 59, 66, 73, 63], [76, 103, 124, 139, 146, 137, 153, 154], [61, 56, 55, 56, 52, 45, 55, 55]],[[26, 31, 37, 48, 48, 60, 71, 90], [113, 121, 127, 133, 133, 155, 156, 146], [68, 76, 91, 91, 91, 92, 91, 96]],[[38, 28, 34, 27, 33, 30, 35, 37], [100, 88, 79, 98, 102, 104, 98, 88], [44, 36, 42, 38, 38, 34, 36, 34]]] normMEK=datanorm(rawinpdata) drugvalsnorm=drugvals[1:] RAFval=0.04 #uM Kdimval=0.1 #uM KAval=0.0001 # Fixes CA mechanism. Ignored when CA is specified in fitflag and KA is therefore varied. normtyp=1 return fitdata(ydata=normMEK,drugvalues=drugvalsnorm,drugnames=drugnames,optionsinp= {'fitflag':ff,'RAFval':RAFval,'Kdimval':Kdimval,'KA':KAval,'normtype':normtyp})%%time # uncomment this block to run the simulation anew. - This is not necessary to reproduce results, prerun results are provided. Nruns=1500 # Total number of runs NeachRun=100 # store in file after this many runs niter=int(Nruns/NeachRun) # For the reasons of python importing system, restart the kernel each time a new fitflag is used - otherwise older libraries could remain in the memory. ff='2' AlgoInit=[ff]*NeachRun # Parallel process resglobal=[] for iter1 in range(niter): if __name__=='__main__': npool=min([len(AlgoInit),5]) po=Pool(npool) resl=list(po.map(fitmeta,AlgoInit)) po.close() po.join() resglobal=resglobal+resl print(iter1+1,'of',niter) # since the run keeps taking too much time, save intermeidate results so far in a file and keep updating. it1=0 for idf in resglobal: idf['irun']=it1 it1+=1 dfres=pd.concat([elem for elem in resglobal if len(elem)>0]) foutname='SKMEL2_Karoulia_ff'+ff+'_normtp1_1Dec21.gz' dfres.to_csv(foutname,sep='\t',index=False)

In [3]:
```
filelist=['SKMEL2_Karoulia_ff2_normtp1_1Dec21.gz','SKMEL2_Karoulia_ffCADP_normtp1_1Dec21.gz','S
modelslist=[ifile.split('_')[-3].split('ff')[1] for ifile in filelist]
```

In [4]:
```
dic_res10p=dict()
dic_dfreslist=dict()
it1=0
for ifile in filelist:
    dfres=pd.read_csv(ifile,sep='\t')
    imodel=modelslist[it1]
```

```
    dic_dfreslist[imodel]=dfres
    minerr=min(dfres['fitmetric'])
    max10p=minerr*1.1 # Since the best fit is not unique, a small interval is chosen around it
#     max10p=dfres.fitmetric.quantile(0.1)
    dic_res10p[imodel]=dfres[dfres['fitmetric']<=max10p].sort_values(by='fitmetric')
    it1+=1
    del dfres
#     del minerr
    del max10p
```

In [5]:
```
setlen=lambda x:len(set(x))
```

In [6]:
```
setlen(dic_dfreslist['2'].irun),setlen(dic_res10p['2'].irun)
```

Out[6]:
```
(1500, 35)
```

In [7]:
```
dic_fitmetric={ikey:[dic_res10p[ikey].drop_duplicates('irun')['fitmetric'].quantile(0.025),dic_
df_fitmetric=pd.DataFrame(dic_fitmetric.values(),columns=['CL2.5','Mean','CL97.5'],index=models
```

In [8]:
```
for ikey in modelslist:
    dic_res10p[ikey]['Model']=ikey
```

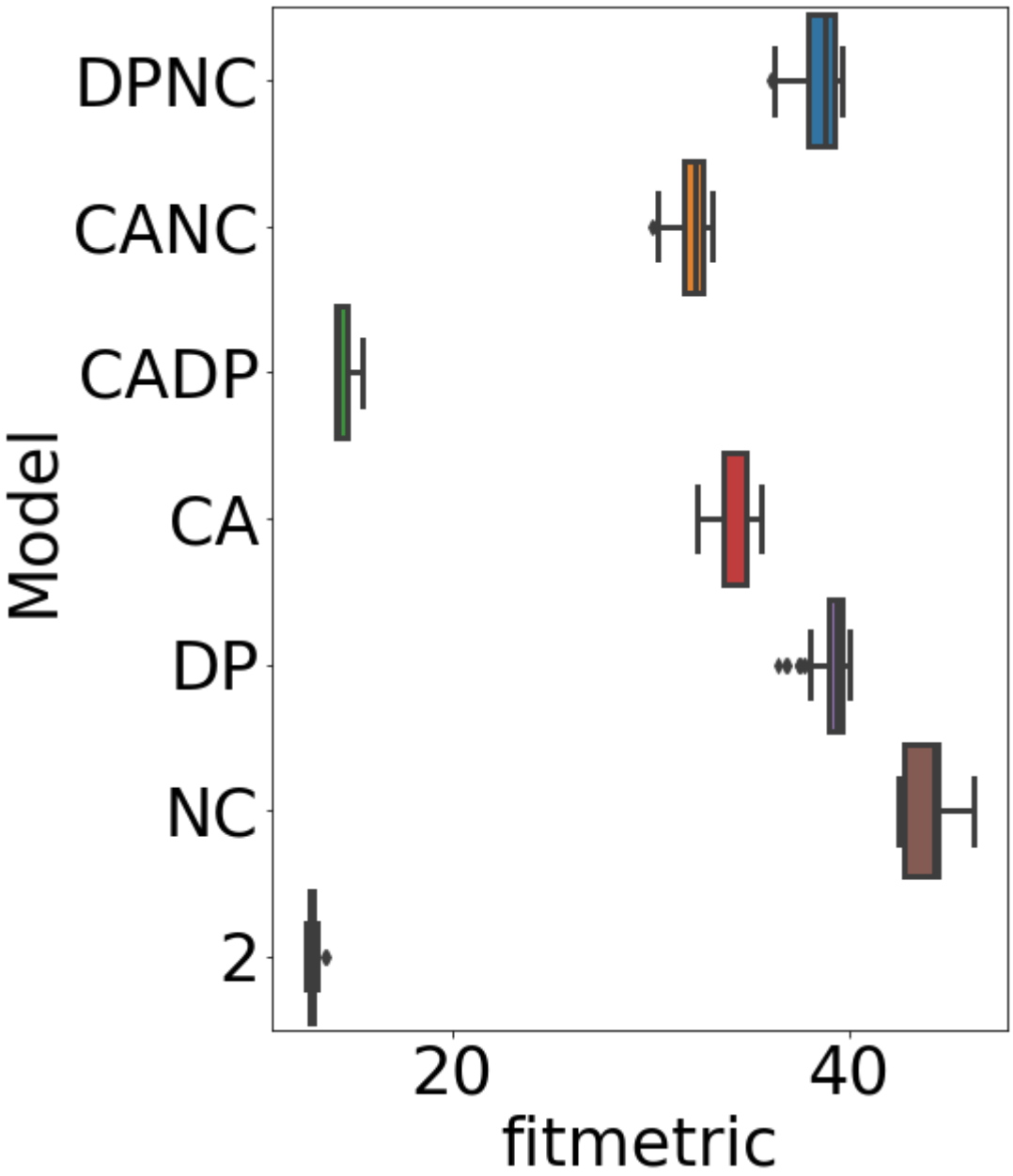
In [9]:
```
dfFM=pd.concat(dic_res10p.values()).drop_duplicates('irun')
```

In [10]:
```
set(dfFM.Model)
```

Out[10]:
```
{'2', 'CA', 'CADP', 'CANC', 'DP', 'DPNC', 'NC'}
```

Figure 3D

In [11]:
```
plt.rc('axes', labelsize=35)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=35)    # fontsize of the tick labels
plt.rc('ytick', labelsize=35)    # fontsize of the tick labels
plt.subplots(figsize=(7,10))
plot1=sbn.boxplot(y='Model',x='fitmetric',data=dfFM,linewidth=3,width=0.9,order=['DPNC', 'CANC'
plt.show()
```


DPNC
CANC
CADP
CA
DP
NC
2
Model
20
40
fitmetric

/