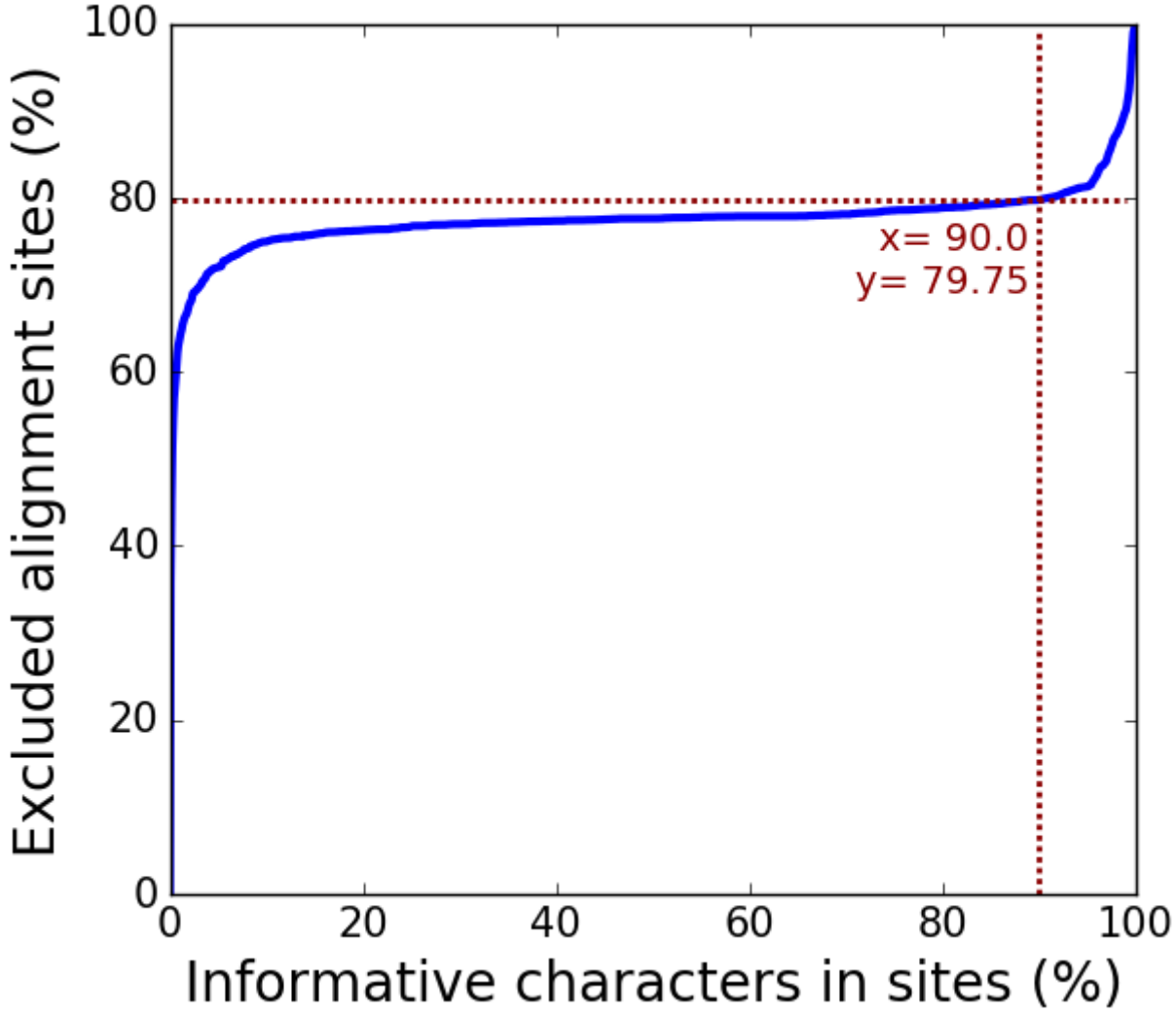
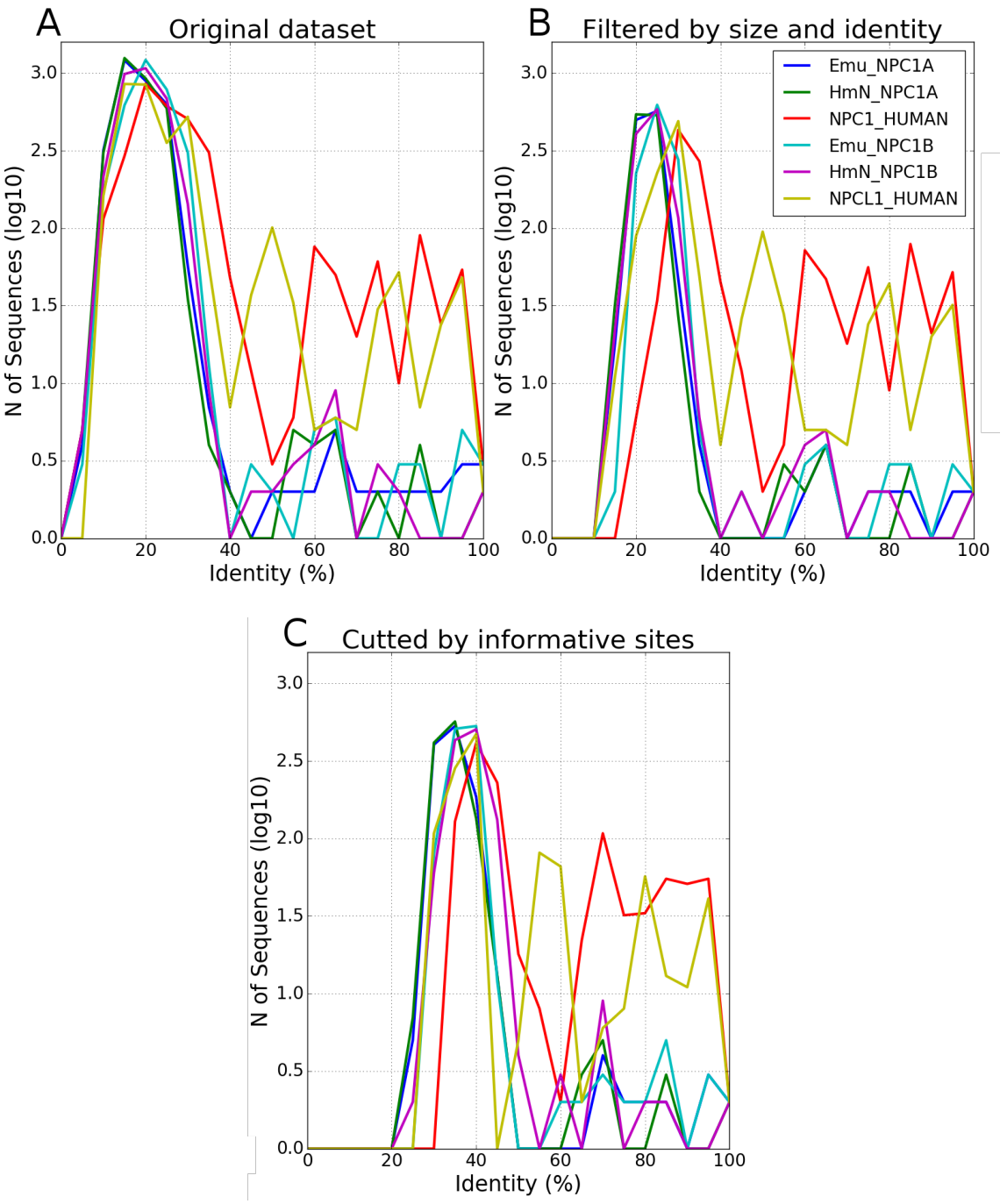
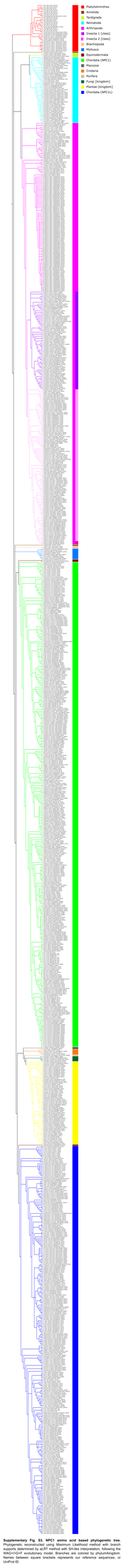


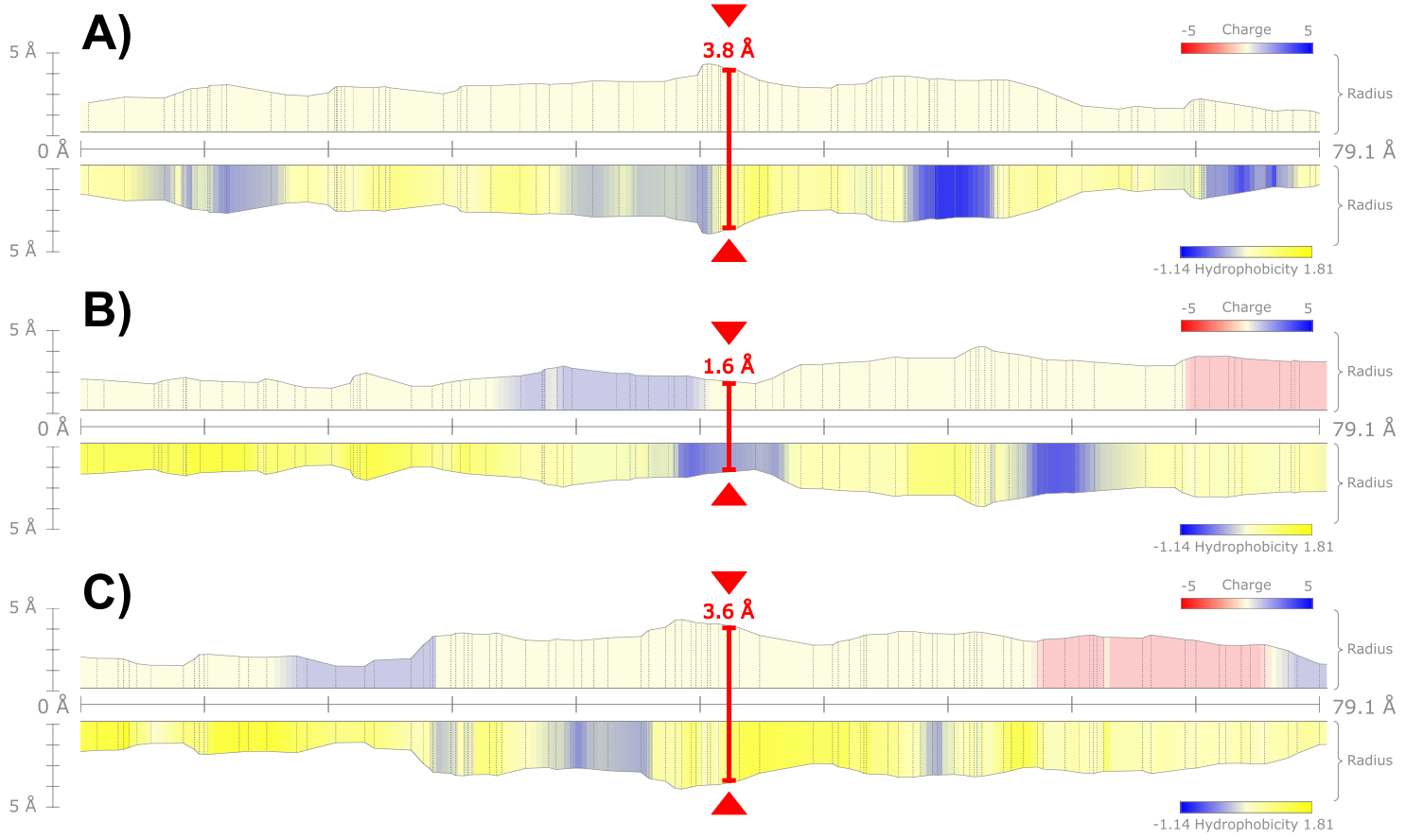
Supplementary Fig. S1. Identification of non-informative sites. The graphic represents the number of sites that would be excluded given a cutoof of different percentages of informative characters in sites. An exclusion of sites with less than 10% of informative characters, would result in excluding 74.94% of the sites from the alignment. However, when excluding sites with less than 90% informative characters, the exclusion of sites only rose to 79.75% of the alignment. This shows us that our alignment presented a majority of sites with very little information compared to sites that concentrated most of the relevant information for our subsequent analyses. Script used in this analysis is available in Supplementary Data S1.



Supplementary Fig. S2. Comparison of the representativeness of the original HMMER-generated dataset, the filtered subset used for the initial alignment and resulting final aligned dataset after editing by exclusion of non-informative sites. A identity matrix of all sequences in those datasets were calculated to each reference sequence (*E. multilocularis* NPC1A and NPC1B, *H. microstoma* NPC1A and NPC1B, and Human NPC1 and NPC1L1). The number of sequences for each identity value was plotted. This generates identity patterns to each reference sequence, in each analysed dataset. Comparisons of those patterns suggest that after the filterring of the dataset, and after the exclusion of non-informative sites of the alignment, the retained sites maintained the representativeness of the initial dataset, demonstrating that the final protein alignment remains appropriate for further evolutionary analyses. Script used in this analysis is available in Supplementary Data S2.



Supplemenary Fig. S3. NPC1 amino acid based phylogenetic tree. Phylogenetic reconstructed using Maximum Likelihood method with branch supports determined by aLRT method with SH-like interpretation, following the WAG+H+F evolutionary model. Branches are colored by phylum/kingdom. Names between square brackets represents our reference sequences, or UniProt ID.



Supplementary Fig. S3. Measures of the radius of the cholesterol delivering tunnel in analyzed models. MOLEonline was used to analyze the tunnel in the *Hs*NPC1L1 (A), *Em*NPC1A (B) and *Em*NPC1B (C) protein models. Bars denoted the radius of the tunnel along its length. The predicted interruption in EmNPC1A is indicated in the three analysis by red arrows, with the radius measure of this region indicated, as well. Colors of the bars indicated the charge and hydrophobicity of aminoacids along the tunnel, accordingly to the legends.

Supplementary Data S1. Autoral python script to the identification of non-informative sites in the alignment.

```
from sys import argv, exit
import matplotlib.pyplot as plt
fig, axs = plt.subplots(1, 1)

if len(argv) != 3:
  print '\nUsage: ' + argv[0] + ' <align.fas> Cutoff%' #/del_sitios.txt'
  exit()

arq1in = open(argv[1], 'r')
arq1 = arq1in.read()
arq1in.close()

iscutoff = '?'
try:
  cutoff = float(argv[2])
  iscutoff = True
except ValueError:
  iscutoff = False

#_____________Fasta Input_____________________________________
lista = []
ids = []
seqs = []
lista = arq1[1:].split('\n>')
for i in lista:
  temp_seq = i.split('\n')
  ids.append(temp_seq[0])
  seqs.append(''.join(temp_seq[1:]).replace('\n', ''))
del(lista)
#_____________________________________________________________


siteinf = [0]*len(seqs[0])

for i in seqs:
  for j in range(len(siteinf)):
    if i[j] != '-':
      siteinf[j] = siteinf[j]+1
distseq = []

site_n = 0.0
site_t = 0.0
for i in range(len(seqs)):
  site_n = siteinf.count(i)
  site_t = site_t + site_n
  distseq.append(site_t *100 / len(seqs[0]))

realcut = cutoff * len(seqs) /100 # cutoff
delsites = []
for i in range(len(siteinf)):
  if siteinf[i] >= realcut:
    continue
  else: delsites.append(i)

deletedp = len(delsites) *100.0/len(seqs[0])

insites = list(set(range(len(seqs[0]))) - set(delsites)) #list of sites that must be excluded
for i in seqs:
  finalseq.append('')
  for j in range(len(i)):
    if j in insites:
      finalseq[-1] += i[j]
#_____________________________________________________________

#_____________MatPlot_____________________________________
axs.set_ylim(0, 100)
axs.set_xlim(0, 100)
axs.set_xlabel('Informative characters in sites (%)', size=20)
axs.set_ylabel('Excluded alignment sites (%)', size=20)

percents = []
for i in range(len(seqs)):
  percents.append(i*100.0/len(seqs))

axs.annotate('x= '+ str(round(cutoff , 2)) +'\ny= '+ str(round(deletedp, 2)), color = 'darkred', size=14, weight='normal',
            xy=(cutoff, deletedp), xycoords='data',
            xytext=(-5, -10), textcoords='offset pixels',
            horizontalalignment='right',
            verticalalignment='top')

axs.yaxis.set_tick_params(labelsize=15)
axs.xaxis.set_tick_params(labelsize=15)

axs.plot(percents, distseq, linewidth=3)
axs.plot([cutoff]*100, range(100), linewidth=2, dashes=[2,2], color='darkred')
axs.plot(range(100), [deletedp]*100, linewidth=2, dashes=[2,2], color='darkred')
plt.show()
#_____________________________________________________________

#_____________Fasta Output_____________________________________
arqf = open(''.join(argv[1].split('.')[0:-1])  + '_cut'+ str(cutoff) +'.fas', 'w')
for i in range(len(ids)):
  arqf.write(    '>' + ids[i] + '\n' + finalseq[i] + '\n'   )
arqf.close()
#_____________________________________________________________
```

Supplementary Data S2. Autoral python script to the analysis of representativeness of the resulting final aligned dataset in comparison to the original HMMER-generated dataset and the filtered subset used for the initial alignment

```
import numpy as np
import matplotlib.pyplot as plt
from math import log as lg

ids = ['Emu_NPC1A', 'HmN_NPC1A', 'NPC1_HUMAN', 'Emu_NPC1B', 'HmN_NPC1B', 'NPCL1_HUMAN']
matrix = []

arqin = open('npc1-matrix.csv', 'r') ##INPUT##  IDENTITY MATRIX
arq = arqin.readlines()
arqin.close()
for i in ids:
 for j in arq:
  if j.split(' ')[0] == i: matrix.append(j)
del(arq)

arqin = open('filtered_matrix_nocopy_final.csv', 'r') ##INPUT## IDENTITY MATRIX FITLTRED
arq = arqin.readlines()
arqin.close()
for i in ids:
 for j in arq:
  if j.split(' ')[0] == i:
   while '  ' in j: j = j.replace('  ', ' ')
   matrix.append(j)
del(arq)

arqin = open('matrixcut90.csv', 'r') ##INPUT## IDENTITY MATRIX OF EXCLUDED ALIGNMENT (CUTOFF 90%)
arq = arqin.readlines()
arqin.close()
for i in ids:
 for j in arq:
  if j.split(' ')[0] == i:
   while '  ' in j: j = j.replace('  ', ' ')
   matrix.append(j)
del(arq)


i00=1
i05=1
i10=1
i15=1
i20=1
i25=1
i30=1
i35=1
i40=1
i45=1
i50=1
i55=1
i60=1
i65=1
i70=1
i75=1
i80=1
i85=1
i90=1
i95=1
i100=1

fig, axs = plt.subplots(1, 3)
t = np.arange(0, 101, 5)
a= 0
b= 0
c = 0
for i in matrix:
 for j in i.replace('\n', '').split(' ')[1:]:
  if float(j)>=100: i100+=1
  elif float(j)>=95: i95+=1
  elif float(j)>=90: i90+=1
  elif float(j)>=85: i85+=1
  elif float(j)>=80: i80+=1
  elif float(j)>=75: i75+=1
  elif float(j)>=70: i70+=1
  elif float(j)>=65: i65+=1
  elif float(j)>=60: i60+=1
  elif float(j)>=55: i55+=1
  elif float(j)>=50: i50+=1
  elif float(j)>=45: i45+=1
  elif float(j)>=40: i40+=1
  elif float(j)>=35: i35+=1
  elif float(j)>=30: i30+=1
  elif float(j)>=25: i25+=1
  elif float(j)>=20: i20+=1
  elif float(j)>=15: i15+=1
  elif float(j)>=10: i10+=1
  elif float(j)>=05: i05+=1
  elif float(j)>=00: i00+=1

 ins = (lg(i00,10), lg(i05,10), lg(i10,10), lg(i15,10), lg(i20,10), lg(i25,10), lg(i30,10), lg(i35,10), lg(i40,10), lg(i45,10),
lg(i50,10), lg(i55,10), lg(i60,10), lg(i65,10), lg(i70,10), lg(i75,10), lg(i80,10), lg(i85,10), lg(i90,10), lg(i95,10),
lg(i100,10))
 axs[b].plot(t, ins, label= i.split(' ')[0], linewidth=3)
 axs[b].set_xlabel('Identity (%)', size=25)
 axs[b].set_ylabel('N of Sequences (log10)', size=25)
 axs[b].grid(True)
 axs[b].set_ylim(0, 3.2)
 axs[b].yaxis.set_tick_params(which='major', labelsize=20)
 axs[b].xaxis.set_tick_params(which='major', labelsize=20)
 c = c+1
 if b==0 and c%(len(matrix)/3)==0: b=1
 elif b==1 and c%(len(matrix)/3)==0: b=2
 i00=1
 i05=1
 i10=1
 i15=1
 i20=1
 i25=1
 i30=1
 i35=1
 i40=1
 i45=1
 i50=1
 i55=1
 i60=1
 i65=1
 i70=1
 i75=1
 i80=1
 i85=1
 i90=1
 i95=1
 i100=1

axs[0].set_title('Original dataset', size=30)
axs[1].set_title('Filtered by size and identity', size=30)
axs[1].legend(loc='upper right', fontsize=20)
axs[2].set_title('Cutted by informative sites', size=30)

fig.tight_layout()
plt.show()
```