2020-04-21

Group 3:

Examines excess number of matings in female lifetime (~4 instead of ~2.5)

Cyan: Same as magenta from group 2 (uniform, new reservoir BV estimation, external pressure from reservoir as function of farm offspring)

Red: number of mating events per timestep = min(available males, females) - Pois(\lambda=2)

Magenta: Same as red but with age preference squared

Made no difference to the number of matings (or anything else). Probably because reducing the number of matings by Pois(\lambda=2) is insignificant in the grand scheme of things and so exaggerating the age preference will not have much of an affect.

-> Running group 3 with Pois(\lambda=round(min(available males, females)/10))

Also been attending Covid19 meeting, GAPostdocs meeting and getting data exercises done for MOOC1 of micromasters (to be completed by the end of the month).

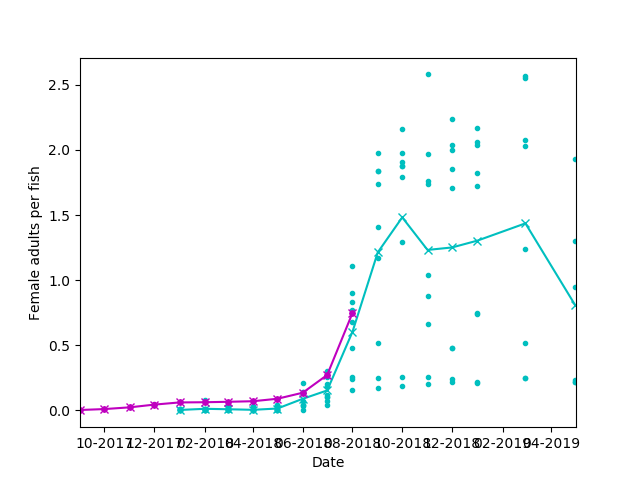
Actions: 1) Complete MOOC exercises

2) use outcome of group 3 to run group 4 and treatment scenarios

3) attend covid19 meetings as and when requested

4) work on paper draft

2020-04-07



Cyan: Loch Fyne data

Magenta: simulated data

Max memory used 25Gb

\*Hurrah! -Jess\*

2020-04-01

Memory issues caused by large increase in lice but more efficient code is always beneficial.

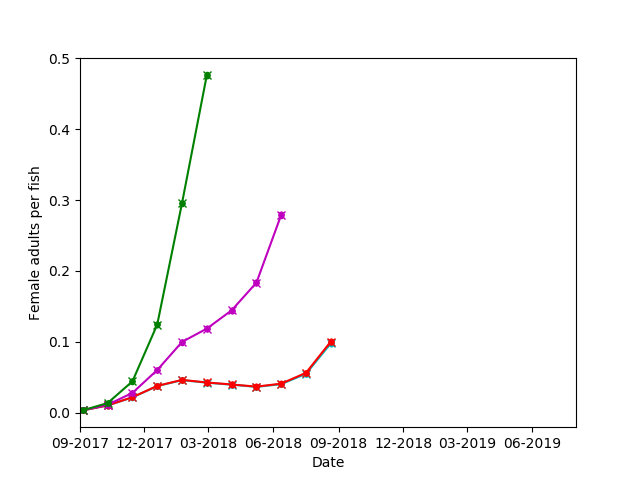
Group 2:

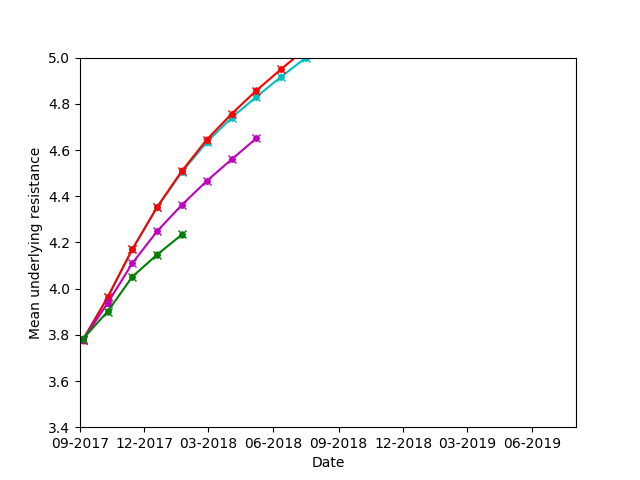
Cyan: Same as magenta from group 1 (preEPIC with minor updates and uniform values for farms)

Red: Equivalent of cyan but with estimation of reservoir BVs as weighted mean from farms, reservoir and influx. Also, the external pressure on farms is constant I.e. ax + b with a=0

Magenta: Equivalent of red but with external pressure on farms from reservoir as a linear function of the number of offspring created on the farms with a = 0.003

Green: a=0.01





Groups 3 and 4 have been resubmitted with a=0.001.

2020-03-24

Notes from meeting:

BV issues resolved – Hurrah!

(Proposed) Actions:

* Focus task: Paper drafting
* JE contact Ant about Memory leak debugging

2020-03-20

Recorded talk sea lice for second year undergrads

Got more memory issues at the following lines:

df = df\_list[fc].loc[df\_list[fc].stage==i].copy()

offspring = pd.concat(offs\_lst)

df\_list[fc].loc[df\_list[fc].avail>0, 'avail'] = df\_list[fc].loc[df\_list[fc].avail>0, 'avail'] + tau

df\_list[fc] = df\_list[fc].append(plankt\_cage, ignore\_index=True)

Also looked at memory usage of entire code after running it for 36 simulated days:

Line # Mem usage Increment Line Contents

================================================

62 81.719 MiB 0.000 MiB @profile

63 def func1():

64

65 81.719 MiB 0.000 MiB v\_path = sys.argv[2]

66 81.719 MiB 0.000 MiB file\_path = "./outputs" + v\_path

67

68 81.719 MiB 0.000 MiB sys.path.append(file\_path)

69

70 81.719 MiB 0.000 MiB file\_in = sys.argv[1]

71 82.641 MiB 0.922 MiB inpt=\_\_import\_\_(file\_in, globals(), locals(), ['\*'])

72

73 82.641 MiB 0.000 MiB v\_file = str(sys.argv[3])

74

75 #np.random.seed(545836870)

76

77 #Functions-------------------------------------------------------------------------------------

78

79 #Update resistance distribution and sample from it

80 535.180 MiB 452.539 MiB def resistEMB(prp\_ext, frms\_muEMB, res\_muEMB, length=1):

81 535.180 MiB 0.000 MiB EMB\_out = []

82 535.180 MiB 0.000 MiB est\_muEMB = prp\_ext\*(inpt.prop\_influx\*inpt.f\_muEMB + (1-inpt.prop\_influx)\*res\_muEMB) + (1-prp\_ext)\*frms\_muEMB

83 535.180 MiB 0.000 MiB for i in range(length):

84 535.180 MiB 0.000 MiB EMB\_out.extend([np.random.normal(est\_muEMB, inpt.f\_sigEMB)])

85 535.180 MiB 0.000 MiB return EMB\_out

86

87 #Fish background mortality rate, decreasing as in Soares et al 2011

88 535.180 MiB 0.000 MiB def fb\_mort(jours):

89 535.180 MiB 0.000 MiB return 0.00057#(1000 + (jours - 700)\*\*2)/490000000

90

91 #prob of developping after D days in a stage as given by Aldrin et al 2017

92 535.180 MiB 0.000 MiB def devTimeAldrin(del\_p, del\_m10, del\_s, temp\_c, D):

93 535.180 MiB 0.000 MiB unbounded = log(2)\*del\_s\*D\*\*(del\_s-1)\*(del\_m10\*10\*\*del\_p/temp\_c\*\*del\_p)\*\*(-del\_s)

94 535.180 MiB 0.000 MiB unbounded[unbounded==0] = 10\*\*(-30)

95 535.180 MiB 0.000 MiB unbounded[unbounded>1] = 1

96 535.180 MiB 0.000 MiB return unbounded.astype('float64')

97

98 #average dev days using devTimeAldrin, not used in model but handy to have

99 #5deg: 5.2,-,67.5,2

100 #10deg: 3.9,-,24,5.3

101 #15deg: 3.3,-,13.1,9.4

102 82.641 MiB -452.539 MiB def aveDevDays(del\_p, del\_m10, del\_s, temp\_c):

103 return 100\*devTimeAldrin(del\_p, del\_m10, del\_s, temp\_c, 100)\

104 -0.001\*sum([devTimeAldrin(del\_p, del\_m10, del\_s, temp\_c, i)

105 for i in np.arange(0,100.001,0.001)])

106

107 82.641 MiB 0.000 MiB def eudist(pointA,pointB):

108 return sqrt((pointA[0]-pointB[0])\*\*2 + (pointA[1]-pointB[1])\*\*2)

109

110

111 #----------------------------------------------------------------------------------------------

112 #----------------------------------------------------------------------------------------------

113

114

115 #Input Data------------------------------------------------------------------------------------

116 82.641 MiB 0.000 MiB EMBmort = 0.9

117

118 82.641 MiB 0.000 MiB hrs\_travel = inpt.E\_days

119 82.641 MiB 0.000 MiB prop\_arrive = inpt.prob\_arrive

120

121

122 82.641 MiB 0.000 MiB mort\_rates = np.array([0.17, 0.22, 0.008, 0.05, 0.02, 0.06]) # L1,L2,L3,L4,L5f,L5m

123 82.641 MiB 0.000 MiB eggs = inpt.eggs

124 82.641 MiB 0.000 MiB d\_hatching = inpt.d\_hatching #8#[9,10,11,9,8,6,4,4,4,4,5,7]

125

126 82.641 MiB 0.000 MiB tau = 1 ###############################################

127

128 #Initial Values--------------------------------------------------------------------------------

129 82.641 MiB 0.000 MiB cur\_date = inpt.start\_date

130

131 82.641 MiB 0.000 MiB fcID = ['f'+str(i)+'c'+str(j) for i in range(1,inpt.nfarms+1) for j in range(1,inpt.ncages[i-1]+1)]

132 193.578 MiB 110.938 MiB fsh = [list(range(1,inpt.fishf[i]+1)) for i in range(inpt.nfarms) for j in range(inpt.ncages[i])]

133 193.578 MiB 0.000 MiB all\_fish = dict(zip(fcID,fsh))

134

135 #Lice population in farms

136 193.922 MiB 0.344 MiB df\_list = [pd.DataFrame(columns=['Farm','Cage','Fish','MF','stage','stage\_age','avail',\

137 'mate\_resistanceT1','resistanceT1','date','arrival','nmates']) \

138 193.922 MiB 0.000 MiB for i in range(inpt.nfarms) for j in range(inpt.ncages[i])]

139 193.922 MiB 0.000 MiB offspring = pd.DataFrame(columns=df\_list[0].columns)

140

141 193.922 MiB 0.000 MiB lifemates = []

142 193.922 MiB 0.000 MiB offs\_len = 0

143 193.922 MiB 0.000 MiB prevOffs\_len = 0

144 193.922 MiB 0.000 MiB env\_sigEMB = 1.0

145 193.922 MiB 0.000 MiB farms\_muEMB = [inpt.f\_muEMB]\*inpt.nfarms

146 193.922 MiB 0.000 MiB farms\_sigEMB = [inpt.f\_sigEMB]\*inpt.nfarms

147 193.922 MiB 0.000 MiB prev\_muEMB = farms\_muEMB.copy()

148 193.922 MiB 0.000 MiB prev\_sigEMB = farms\_sigEMB.copy()

149 193.922 MiB 0.000 MiB res\_muEMB = inpt.f\_muEMB

150 193.922 MiB 0.000 MiB pres\_muEMB = inpt.f\_muEMB

151 193.922 MiB 0.000 MiB prop\_ext = 1

152 193.922 MiB 0.000 MiB plankt\_resist = []

153 #----------------------------------------------------------------------------------------------

154 #----------------------------------------------------------------------------------------------

155 #--------------------------Simulation----------------------------------------------------------

156 #----------------------------------------------------------------------------------------------

157 #----------------------------------------------------------------------------------------------

158 193.922 MiB 0.000 MiB directory = os.path.dirname(file\_path)

159 193.922 MiB 0.000 MiB try:

160 193.922 MiB 0.000 MiB if not os.path.exists(directory):

161 os.makedirs(directory)

162 except OSError:

163 print('Error creating directory')

164 193.922 MiB 0.000 MiB file1 = open(file\_path + 'lice\_counts' + v\_file + '.txt','a+')

165 193.922 MiB 0.000 MiB file2 = open(file\_path + 'resistanceBVs' + v\_file + '.csv','a+')

166 193.922 MiB 0.000 MiB print('farm', 'cur\_date', 'muEMB', 'sigEMB', 'prop\_ext', file=file2, sep=',', flush=True)

167 #prev\_time = time.time()

168 #prev\_femaleAL = [0]\*(inpt.nfarms-1)

169 193.973 MiB 0.051 MiB delta\_treat = [0]\*(inpt.nfarms)

170 533.254 MiB 339.281 MiB while cur\_date <= inpt.end\_date:

171

172 498.105 MiB -35.148 MiB cur\_date = cur\_date + dt.timedelta(days=tau)

173 498.105 MiB 0.000 MiB t = (cur\_date - inpt.start\_date).days

174

175 # if (t%7)==0:

176 # print((time.time()-prev\_time)/60, ' date ' + str(cur\_date), file=file, flush=True)

177 # prev\_time = time.time()

178

179

180 #add new offspring to cages

181 498.105 MiB 0.000 MiB offs\_len = offs\_len + len(offspring.index)

182 498.105 MiB 0.000 MiB k = -1

183 503.977 MiB 5.871 MiB for f in range(1,inpt.nfarms+1):

184 503.977 MiB 0.000 MiB for c in range(1,inpt.ncages[f-1]+1):

185 503.977 MiB 0.000 MiB k = k + 1

186 503.977 MiB 0.000 MiB if len(offspring.index)>0:

187 503.977 MiB 0.000 MiB df\_list[k] = df\_list[k].append(offspring[(offspring['Farm']==f) & (offspring['Cage']==c)].copy(), ignore\_index=True)

188 #offspring = offspring.drop(offspring[(offspring['Farm']==f) & (offspring['Cage']==c)].index)

189 498.105 MiB -5.871 MiB offspring = pd.DataFrame(columns=df\_list[0].columns)

190

191 498.105 MiB 0.000 MiB if (t%35)==0:

192 466.367 MiB -31.738 MiB res\_muEMB = pres\_muEMB

193 466.367 MiB 0.000 MiB pres\_muEMB = np.mean(plankt\_resist)

194 466.367 MiB 0.000 MiB plankt\_resist = []

195 466.367 MiB 0.000 MiB prop\_ext = (sum(inpt.ncages)\*eval(inpt.ext\_pressure))/(sum(inpt.ncages)\*eval(inpt.ext\_pressure) + offs\_len/35)

196 466.367 MiB 0.000 MiB prevOffs\_len = offs\_len/35

197 466.367 MiB 0.000 MiB offs\_len = 0

198 466.367 MiB 0.000 MiB if len(lifemates)>0:

199 466.371 MiB 0.004 MiB print(cur\_date, len(lifemates), np.nanmean(lifemates),flush=True)

200 466.371 MiB 0.000 MiB lifemates=[]

201

202 #------------------------------------------------------------------------------------------

203 #Events during tau in cage-----------------------------------------------------------------

204 #------------------------------------------------------------------------------------------

205 498.105 MiB 31.734 MiB fc = -1

206 533.254 MiB 35.148 MiB for farm in range(1, inpt.nfarms+1):

207 527.238 MiB -6.016 MiB if cur\_date.day==1:

208 433.121 MiB -94.117 MiB femaleAL = np.array([],dtype=float)

209

210 #Estimate distribution params for external pressure that originated from farms

211 527.238 MiB 94.117 MiB if ((t%35)==0) & (cur\_date >= inpt.farm\_start[farm-1] + dt.timedelta(days=35)):

212 474.648 MiB -52.590 MiB farms\_muEMB[farm-1] = prev\_muEMB[farm-1]

213 474.648 MiB 0.000 MiB farms\_sigEMB[farm-1] = prev\_sigEMB[farm-1]

214 474.648 MiB 0.000 MiB resistanceT1 = []

215 474.648 MiB 0.000 MiB if farm==1:

216 466.914 MiB -7.734 MiB for i in range(inpt.ncages[farm-1]):

217 466.914 MiB 0.000 MiB resistanceT1.extend(df\_list[i].resistanceT1)

218 else:

219 474.648 MiB 7.734 MiB for i in range(sum(inpt.ncages[0:farm-1]),sum(inpt.ncages[0:farm])):

220 474.648 MiB 0.000 MiB resistanceT1.extend(df\_list[i].resistanceT1)

221 474.648 MiB 0.000 MiB prev\_muEMB[farm-1] = np.nanmean(resistanceT1)

222 474.648 MiB 0.000 MiB prev\_sigEMB[farm-1] = np.nanstd(resistanceT1)

223 474.648 MiB 0.000 MiB print(farm, cur\_date, prev\_muEMB[farm-1], prev\_sigEMB[farm-1], prop\_ext, file=file2, sep=',', flush=True)

224

225 535.180 MiB 60.531 MiB for cage in range(1, inpt.ncages[farm-1]+1):

226 535.180 MiB 0.000 MiB fc = fc + 1

227

228 535.180 MiB 0.000 MiB NSbool = eval(inpt.NSbool\_str)

229 535.180 MiB 0.000 MiB if NSbool==True:

230

231 535.180 MiB 0.000 MiB if cur\_date.day==1:

232 433.629 MiB -101.551 MiB femaleAL = np.append(femaleAL, sum((df\_list[fc].stage==5)/len(all\_fish['f'+str(farm)+'c'+str(cage)])))

233

234 535.180 MiB 101.551 MiB temp\_now = inpt.temp\_f(cur\_date.month, inpt.xy\_array[farm-1][1])

235

236 535.180 MiB 0.000 MiB if not df\_list[fc].empty:

237 535.180 MiB 0.000 MiB df\_list[fc].date = cur\_date

238 535.180 MiB 0.000 MiB df\_list[fc].stage\_age = df\_list[fc].stage\_age + tau

239 535.180 MiB 0.000 MiB df\_list[fc].arrival = df\_list[fc].arrival - tau

240 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].avail>0, 'avail'] = df\_list[fc].loc[df\_list[fc].avail>0, 'avail'] + tau

241 535.180 MiB 0.000 MiB df\_list[fc].loc[(df\_list[fc].MF=='M') & (df\_list[fc].avail>4), 'avail'] = 0

242 535.180 MiB 0.000 MiB df\_list[fc].loc[(df\_list[fc].MF=='F') & (df\_list[fc].avail>d\_hatching(temp\_now)), 'avail'] = 0

243 535.180 MiB 0.000 MiB df\_list[fc].loc[(df\_list[fc].MF=='F') & (df\_list[fc].avail>d\_hatching(temp\_now)), 'mate\_resistanceT1'] = None

244

245 #new planktonic stages arriving from wildlife reservoir

246 535.180 MiB 0.000 MiB nplankt = eval(inpt.ext\_pressure)\*tau

247 535.180 MiB 0.000 MiB plankt\_cage = pd.DataFrame(columns=df\_list[fc].columns)

248 535.180 MiB 0.000 MiB plankt\_cage['MF'] = np.random.choice(['F','M'],nplankt)

249 535.180 MiB 0.000 MiB plankt\_cage['stage'] = 2

250 535.180 MiB 0.000 MiB plankt\_cage['Farm'] = farm

251 535.180 MiB 0.000 MiB plankt\_cage['Cage'] = cage

252 535.180 MiB 0.000 MiB p=stats.poisson.pmf(range(15),3)

253 535.180 MiB 0.000 MiB p = p/np.sum(p) #probs need to add up to one

254 535.180 MiB 0.000 MiB plankt\_cage['stage\_age'] = np.random.choice(range(15),nplankt,p=p)

255 535.180 MiB 0.000 MiB plankt\_cage['avail'] = 0

256 535.180 MiB 0.000 MiB plankt\_cage['resistanceT1'] = resistEMB(prop\_ext, farms\_muEMB[farm-1], res\_muEMB, nplankt)

257 535.180 MiB 0.000 MiB plankt\_cage['date'] = cur\_date

258 535.180 MiB 0.000 MiB plankt\_cage['avail'] = 0

259 535.180 MiB 0.000 MiB plankt\_cage['arrival'] = 0

260

261 535.180 MiB 0.000 MiB if ((t+1)%35)==0:

262 466.305 MiB -68.875 MiB plankt\_resist.extend(plankt\_cage.resistanceT1)

263

264 535.180 MiB 68.875 MiB df\_list[fc] = df\_list[fc].append(plankt\_cage, ignore\_index=True)

265 535.180 MiB 0.000 MiB del plankt\_cage

266 535.180 MiB 0.000 MiB dead\_fish = set([])

267

268 #Background mortality events-------------------------------------------------------

269 535.180 MiB 0.000 MiB inds\_stage = np.array([sum(df\_list[fc]['stage']==i) for i in range(1,7)])

270 535.180 MiB 0.000 MiB Emort = np.multiply(mort\_rates, inds\_stage)

271 535.180 MiB 0.000 MiB mort\_ents = np.random.poisson(Emort)

272 535.180 MiB 0.000 MiB mort\_ents=[min(mort\_ents[i],inds\_stage[i]) for i in range(len(mort\_ents))]

273 535.180 MiB 0.000 MiB mort\_inds = []

274 535.180 MiB 0.000 MiB for i in range(1,7):

275 535.180 MiB 0.000 MiB df = df\_list[fc].loc[df\_list[fc].stage==i].copy()

276 535.180 MiB 0.000 MiB if not df.empty:

277 535.180 MiB 0.000 MiB if np.sum(df.stage\_age)>0:

278 535.180 MiB 0.000 MiB p = (df.stage\_age+1)/np.sum(df.stage\_age+1)

279 535.180 MiB 0.000 MiB p = pd.to\_numeric(p)

280 535.180 MiB 0.000 MiB values = np.random.choice(df.index, mort\_ents[i-1], p=p, replace=False).tolist()

281 else:

282 values = np.random.choice(df.index, mort\_ents[i-1], replace=False).tolist()

283 535.180 MiB 0.000 MiB mort\_inds.extend(values)

284 535.180 MiB 0.000 MiB if i==5:

285 535.180 MiB 0.000 MiB lifemates.extend(df.nmates[values])

286 535.180 MiB 0.000 MiB del df

287

288 #Treatment mortality events------------------------------------------------------

289 535.180 MiB 0.000 MiB EMBsus = [1 if df\_list[fc].stage[i]>2 else 0 for i in range(len(df\_list[fc].index))]

290 535.180 MiB 0.000 MiB if eval(inpt.bool\_treat):

291 535.180 MiB 0.000 MiB if cage==1:

292 527.238 MiB -7.941 MiB print(farm, cur\_date, flush=True)

293 535.180 MiB 7.941 MiB phenoEMB = df\_list[fc].resistanceT1 + np.random.normal(0,env\_sigEMB,len(df\_list[fc].resistanceT1)) #add environmental deviation

294 535.180 MiB 0.000 MiB phenoEMB = 1/(1 + np.exp(phenoEMB)) #1-resistance

295 535.180 MiB 0.000 MiB phenoEMB = phenoEMB\*EMBsus #remove stages that aren't susceptible to EMB

296 535.180 MiB 0.000 MiB ETmort = sum(phenoEMB)\*EMBmort

297 535.180 MiB 0.000 MiB if ETmort>0:

298 535.180 MiB 0.000 MiB Tmort\_ents = np.random.poisson(ETmort)

299 535.180 MiB 0.000 MiB Tmort\_ents = min(Tmort\_ents,len(df\_list[fc].resistanceT1))

300 535.180 MiB 0.000 MiB p = (phenoEMB)/np.sum(phenoEMB)

301 535.180 MiB 0.000 MiB mort\_inds.extend(np.random.choice(df\_list[fc].index, Tmort\_ents, p=p, replace=False).tolist())

302 535.180 MiB 0.000 MiB mort\_inds = list(set(mort\_inds))

303

304 #Development events----------------------------------------------------------------

305

306 535.180 MiB 0.000 MiB if inds\_stage[0]>0:

307 535.180 MiB 0.000 MiB L1toL2 = devTimeAldrin(0.401,8.814,18.869,temp\_now, df\_list[fc].loc[df\_list[fc].stage==1,'stage\_age'].values)

308 535.180 MiB 0.000 MiB L1toL2\_ents = np.random.poisson(sum(L1toL2))

309 535.180 MiB 0.000 MiB L1toL2\_ents = min(L1toL2\_ents, inds\_stage[0])

310 535.180 MiB 0.000 MiB L1toL2\_inds = np.random.choice(df\_list[fc].loc[df\_list[fc].stage==1].index, \

311 535.180 MiB 0.000 MiB L1toL2\_ents, p=L1toL2/np.sum(L1toL2), replace=False)

312 else:

313 295.637 MiB -239.543 MiB L1toL2\_inds = []

314

315 535.180 MiB 239.543 MiB if inds\_stage[2]>0:

316 535.180 MiB 0.000 MiB L3toL4 = devTimeAldrin(1.305,18.934,7.945,temp\_now,df\_list[fc].loc[df\_list[fc].stage==3,'stage\_age'].values)

317 535.180 MiB 0.000 MiB L3toL4\_ents = np.random.poisson(sum(L3toL4))

318 535.180 MiB 0.000 MiB L3toL4\_ents = min(L3toL4\_ents, inds\_stage[2])

319 535.180 MiB 0.000 MiB L3toL4\_inds = np.random.choice(df\_list[fc].loc[df\_list[fc]['stage']==3].index, \

320 535.180 MiB 0.000 MiB L3toL4\_ents, p=L3toL4/np.sum(L3toL4), replace=False)

321 else:

322 218.262 MiB -316.918 MiB L3toL4\_inds = []

323

324 535.180 MiB 316.918 MiB if inds\_stage[3]>0:

325 535.180 MiB 0.000 MiB L4toL5 = devTimeAldrin(0.866,10.742,1.643,temp\_now,df\_list[fc].loc[df\_list[fc].stage==4,'stage\_age'].values)

326 535.180 MiB 0.000 MiB L4toL5\_ents = np.random.poisson(sum(L4toL5))

327 535.180 MiB 0.000 MiB L4toL5\_ents = min(L4toL5\_ents, inds\_stage[3])

328 535.180 MiB 0.000 MiB L4toL5\_inds = np.random.choice(df\_list[fc].loc[df\_list[fc]['stage']==4].index, \

329 535.180 MiB 0.000 MiB L4toL5\_ents, p=L4toL5/np.sum(L4toL5), replace=False)

330 else:

331 275.000 MiB -260.180 MiB L4toL5\_inds = []

332

333 #Fish growth and death-------------------------------------------------------------

334 535.180 MiB 260.180 MiB wt = 10000/(1+exp(-0.01\*(t-475)))

335 535.180 MiB 0.000 MiB fish\_fc = np.array(df\_list[fc][df\_list[fc].stage>3].Fish.unique().tolist()) #fish with lice

336 535.180 MiB 0.000 MiB fish\_fc = fish\_fc[~np.isnan(fish\_fc)]

337 535.180 MiB 0.000 MiB adlicepg = np.array(df\_list[fc][df\_list[fc].stage>3].groupby('Fish').stage.count())/wt

338 535.180 MiB 0.000 MiB Plideath = 1/(1+np.exp(-19\*(adlicepg-0.63)))

339 535.180 MiB 0.000 MiB nfish = len(all\_fish['f'+str(farm)+'c'+str(cage)])

340 535.180 MiB 0.000 MiB Ebf\_death = fb\_mort(t)\*tau\*(nfish)

341 535.180 MiB 0.000 MiB Elf\_death = np.sum(Plideath)\*tau

342 535.180 MiB 0.000 MiB bfd\_ents = np.random.poisson(Ebf\_death)

343 535.180 MiB 0.000 MiB lfd\_ents = np.random.poisson(Elf\_death)

344

345 535.180 MiB 0.000 MiB if fish\_fc.size>0:

346 535.180 MiB 0.000 MiB dead\_fish.update(np.random.choice(fish\_fc,

347 535.180 MiB 0.000 MiB lfd\_ents, p=Plideath/np.sum(Plideath), replace=False).tolist())

348 535.180 MiB 0.000 MiB if len(dead\_fish)>0:

349 527.238 MiB -7.941 MiB all\_fish['f'+str(farm)+'c'+str(cage)] = [i for i in all\_fish['f'+str(farm)+'c'+str(cage)] if i not in dead\_fish]

350 535.180 MiB 7.941 MiB dead\_fish.update(np.random.choice(all\_fish['f'+str(farm)+'c'+str(cage)], bfd\_ents, replace=False).tolist())

351 535.180 MiB 0.000 MiB if len(dead\_fish)>0:

352 535.180 MiB 0.000 MiB all\_fish['f'+str(farm)+'c'+str(cage)] = [i for i in all\_fish['f'+str(farm)+'c'+str(cage)] if i not in dead\_fish]

353

354 #Infection events------------------------------------------------------------------

355 535.180 MiB 0.000 MiB cop\_cage = sum((df\_list[fc].stage==2) & (df\_list[fc].arrival<=df\_list[fc].stage\_age))

356 535.180 MiB 0.000 MiB if cop\_cage>0:

357 535.180 MiB 0.000 MiB eta\_aldrin = -2.576 + log(nfish) + 0.082\*(log(wt)-0.55)

358 535.180 MiB 0.000 MiB Einf = (exp(eta\_aldrin)/(1+exp(eta\_aldrin)))\*tau\*cop\_cage

359 535.180 MiB 0.000 MiB inf\_ents = np.random.poisson(Einf)

360 535.180 MiB 0.000 MiB inf\_ents = min(inf\_ents,cop\_cage)

361 535.180 MiB 0.000 MiB inf\_inds = np.random.choice(df\_list[fc].loc[(df\_list[fc].stage==2) & (df\_list[fc].arrival<=df\_list[fc].stage\_age)].index, inf\_ents, replace=False)

362 else:

363 inf\_inds = []

364

365 #Mating events---------------------------------------------------------------------

366

367 #who is mating

368 535.180 MiB 0.000 MiB females = df\_list[fc].loc[(df\_list[fc].stage==5) & (df\_list[fc].avail==0)].index

369 535.180 MiB 0.000 MiB males = df\_list[fc].loc[(df\_list[fc].stage==6) & (df\_list[fc].avail==0)].index

370 535.180 MiB 0.000 MiB nmating = min(sum(df\_list[fc].index.isin(females)),\

371 535.180 MiB 0.000 MiB sum(df\_list[fc].index.isin(males)))

372 535.180 MiB 0.000 MiB nmating = nmating - np.random.poisson(2)

373 535.180 MiB 0.000 MiB if nmating>0:

374 535.180 MiB 0.000 MiB sires = np.random.choice(males, nmating, replace=False)

375 535.180 MiB 0.000 MiB p\_dams = 1 - (df\_list[fc].loc[df\_list[fc].index.isin(females),'stage\_age']\*\*2/

376 535.180 MiB 0.000 MiB (np.sum(df\_list[fc].loc[df\_list[fc].index.isin(females),'stage\_age']\*\*2)+1))

377 535.180 MiB 0.000 MiB dams = np.random.choice(females, nmating, p=np.array(p\_dams/np.sum(p\_dams)).tolist(), replace=False)

378 else:

379 318.258 MiB -216.922 MiB sires = []

380 318.258 MiB 0.000 MiB dams = []

381 535.180 MiB 216.922 MiB df\_list[fc].loc[df\_list[fc].index.isin(dams),'avail'] = 1

382 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(sires),'avail'] = 1

383 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(dams),'nmates'] = df\_list[fc].loc[df\_list[fc].index.isin(dams),'nmates'].values + 1

384 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(sires),'nmates'] = df\_list[fc].loc[df\_list[fc].index.isin(sires),'nmates'].values + 1

385 #Add genotype of sire to dam info

386 df\_list[fc].loc[df\_list[fc].index.isin(dams),'mate\_resistanceT1'] = \

387 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(sires),'resistanceT1'].values

388

389

390 #create offspring

391 535.180 MiB 0.000 MiB bv\_lst = []

392 535.180 MiB 0.000 MiB eggs\_now = int(round(eggs\*tau/d\_hatching(temp\_now)))

393 535.180 MiB 0.000 MiB for i in dams:

394 underlying = 0.5\*df\_list[fc].resistanceT1[i]\

395 535.180 MiB 0.000 MiB + 0.5\*df\_list[fc].mate\_resistanceT1[i]+ \

396 535.180 MiB 0.000 MiB np.random.normal(0, farms\_sigEMB[farm-1], eggs\_now+250)/np.sqrt(2)

397 535.180 MiB 0.000 MiB bv\_lst.extend(underlying)

398 535.180 MiB 0.000 MiB new\_offs = len(dams)\*eggs\_now

399 535.180 MiB 0.000 MiB num = 0

400 535.180 MiB 0.000 MiB offs\_lst = [offspring]

401 535.180 MiB 0.000 MiB for f in range(1,inpt.nfarms+1):

402 535.180 MiB 0.000 MiB arrivals = np.random.poisson(prop\_arrive[f-1][farm-1]\*new\_offs)

403 535.180 MiB 0.000 MiB if arrivals>0:

404 535.180 MiB 0.000 MiB num = num + 1

405 535.180 MiB 0.000 MiB offs = pd.DataFrame(columns=df\_list[fc].columns)

406 535.180 MiB 0.000 MiB offs['MF'] = np.random.choice(['F','M'], arrivals)

407 535.180 MiB 0.000 MiB offs['Farm'] = f

408 535.180 MiB 0.000 MiB offs['Cage'] = np.random.choice(range(1,inpt.ncages[f-1]+1), arrivals)

409 535.180 MiB 0.000 MiB offs['stage'] = np.repeat(1, arrivals)

410 535.180 MiB 0.000 MiB offs['stage\_age'] = np.repeat(0, arrivals)

411 535.180 MiB 0.000 MiB if len(bv\_lst)<arrivals:

412 randams = np.random.choice(dams,arrivals-len(bv\_lst))

413 for i in randams:

414 underlying = 0.5\*df\_list[fc].resistanceT1[i]\

415 + 0.5\*df\_list[fc].mate\_resistanceT1[i]+ \

416 np.random.normal(0, farms\_sigEMB[farm-1], 1)/np.sqrt(2)

417 bv\_lst.extend(underlying)

418 535.180 MiB 0.000 MiB ran\_bvs = np.random.choice(len(bv\_lst),arrivals,replace=False)

419 535.180 MiB 0.000 MiB offs['resistanceT1'] = [bv\_lst[i] for i in ran\_bvs]

420 535.180 MiB 0.000 MiB for i in sorted(ran\_bvs, reverse=True):

421 535.180 MiB 0.000 MiB del bv\_lst[i]

422 535.180 MiB 0.000 MiB Earrival = [hrs\_travel[i-1][farm-1] for i in offs.Farm]

423 535.180 MiB 0.000 MiB offs['arrival'] = np.random.poisson(Earrival)

424 535.180 MiB 0.000 MiB offs['avail'] = 0

425 535.180 MiB 0.000 MiB offs['date'] = cur\_date

426 535.180 MiB 0.000 MiB offs['nmates'] = 0

427 535.180 MiB 0.000 MiB offs\_lst.append(offs)

428 535.180 MiB 0.000 MiB del offs

429 535.180 MiB 0.000 MiB offspring = pd.concat(offs\_lst)

430

431

432 #Update cage info------------------------------------------------------------------

433 #----------------------------------------------------------------------------------

434 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(L1toL2\_inds),'stage'] = 2

435 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(L1toL2\_inds),'stage\_age'] = 0

436 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(L3toL4\_inds),'stage'] = 4

437 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(L3toL4\_inds),'stage\_age'] = 0

438 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(L4toL5\_inds) & (df\_list[fc].MF=='F'),'stage'] = 5

439 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(L4toL5\_inds) & (df\_list[fc].MF=='M'),'stage'] = 6

440 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(L4toL5\_inds),'stage\_age'] = 0

441

442 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(inf\_inds), 'stage'] = 3

443 535.180 MiB 0.000 MiB df\_list[fc].loc[df\_list[fc].index.isin(inf\_inds),'stage\_age'] = 0

444 df\_list[fc].loc[df\_list[fc].index.isin(inf\_inds),'Fish'] = \

445 535.180 MiB 0.000 MiB np.random.choice(all\_fish['f'+str(farm)+'c'+str(cage)], len(inf\_inds))

446

447

448 #remove dead individuals

449 535.180 MiB 0.000 MiB df\_list[fc] = df\_list[fc].drop(mort\_inds)

450 535.180 MiB 0.000 MiB df\_list[fc] = df\_list[fc].drop(df\_list[fc].loc[df\_list[fc].Fish.isin(dead\_fish)].index)

451

452

453 #df\_list[fc].to\_csv(file\_path + 'lice\_df.csv', mode='a')

454

455 533.254 MiB -1.926 MiB if cur\_date.day==1:

456 433.121 MiB -100.133 MiB print(cur\_date, femaleAL.mean(), femaleAL.std(), file=file1, flush=True)

457 433.121 MiB 0.000 MiB femaleAL = np.array([],dtype=float)

458

459 533.254 MiB 100.133 MiB file1.close()

460 533.254 MiB 0.000 MiB file2.close()

461 533.254 MiB 0.000 MiB return farms\_muEMB

Made a minor change to the external pressure (updated on git) and set it to run, currently queued.

I don’t want to do a major overhaul of the code while I still have a temperature.

2020-03-12

Diff between group 2 cyan (<) and red (>):

77c77

< def resistEMB(prp\_ext, frms\_muEMB, frms\_sigEMB, length=1):

---

> def resistEMB(prp\_ext, frms\_muEMB, res\_muEMB, length=1):

78a79

> est\_muEMB = inpt.prop\_influx\*inpt.f\_muEMB + (1-inpt.prop\_influx)\*(prp\_ext\*res\_muEMB + (1-prp\_ext)\*frms\_muEMB)

80,84c81

< r = np.random.uniform(0,1,1)

< if r<prp\_ext:

< EMB\_out.extend([np.random.normal(inpt.f\_muEMB, inpt.f\_sigEMB)])

< else:

< EMB\_out.extend([np.random.normal(frms\_muEMB, frms\_sigEMB)])

---

> EMB\_out.extend([np.random.normal(est\_muEMB, inpt.f\_sigEMB)])

147a145,146

> res\_muEMB = inpt.f\_muEMB

> pres\_muEMB = inpt.f\_muEMB

148a148

> plankt\_resist = []

187a188,190

> res\_muEMB = pres\_muEMB

> pres\_muEMB = np.mean(plankt\_resist)

> plankt\_resist = []

191c194

< print(cur\_date, len(lifemates), np.array(lifemates).mean(),flush=True)

---

> print(cur\_date, len(lifemates), np.nanmean(lifemates),flush=True)

248c251

< plankt\_cage['resistanceT1'] = resistEMB(prop\_ext, farms\_muEMB[farm-1], farms\_sigEMB[farm-1], nplankt)

---

> plankt\_cage['resistanceT1'] = resistEMB(prop\_ext, farms\_muEMB[farm-1], res\_muEMB, nplankt)

251a255,257

>

> if ((t+1)%35)==0:

> plankt\_resist.extend(plankt\_cage.resistanceT1)

Inputs:

inpt.prop\_influx = 0.33

prop\_ext = (sum(inpt.ncages)\*inpt.ext\_pressure)/(sum(inpt.ncages)\*inpt.ext\_pressure + offs\_len/35) #proportion of daily planktonic lice arriving from reservoir over all planktonic lice arriving per day

f\_muEMB = 3.5

f\_sigEMB = 0.7

Cyan:

overall mu\_emb = 0.33\*3.5 + (1-0.33)\*frms\_muEMB

Red:

est\_muEMB = 0.33\*3.5 + (1-0.33)\*prp\_ext\*res\_muEMB + (1-0.33)\*(1-prp\_ext)\*frms\_muEMB

With 3.5 <= res\_muEMB <= frms\_muEMB

So that would be less and really instead it should be:

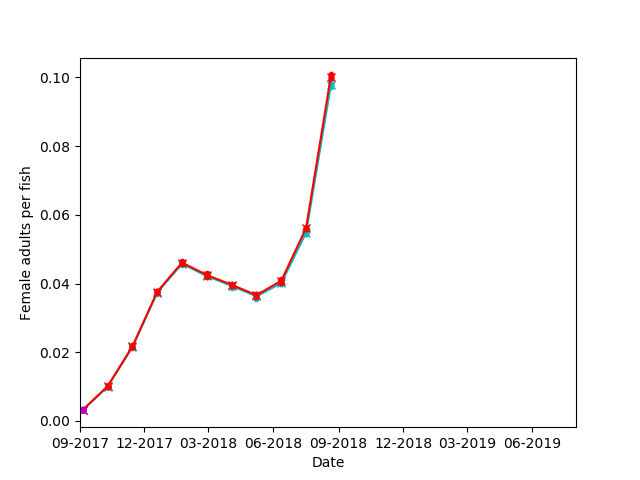
est\_muEMB = prp\_ext\*(0.33\*3.5 + (1-0.33)\*res\_muEMB) + (1-prp\_ext)\*frms\_muEMB

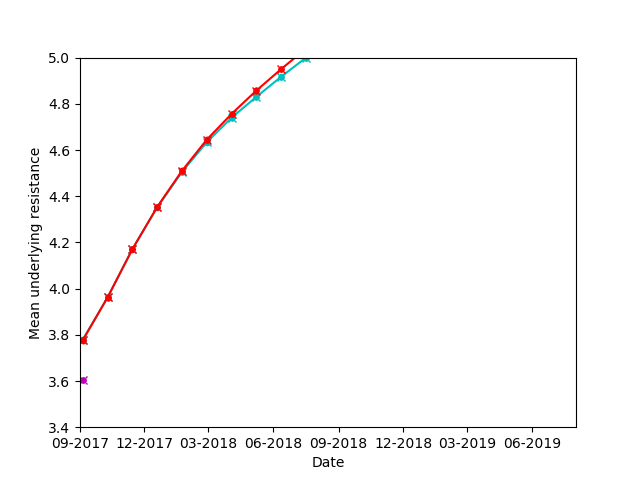
Models have been updated and resubmitted.

Group 2:

Cyan: Same as magenta from group 1 (preEPIC with minor updates and uniform values for farms)

Red: Equivalent of cyan but with estimation of reservoir BVs as weighted mean from farms, reservoir and influx

(Magenta: Equivalent of red but with external pressure on farms from reservoir as a linear function of the number of offspring created on the farms) 



Also dataframe appending has been replaced with less onerous concatenation (see github).

2020-03-10

Notes from meeting:

Debby talked through updates so far, including regression approach, memory management, current outputs. Findings included more cages having a dampening effect early in epidemic – possibly due to diluting effect of many fish. Some uncertainty about how breeding values are calculated, and therefore order of teal/red lines on figure below. - Debby to investigate this offline and report next time.

Tasks agreed as below (copied here):

Actions:

1. Currently queued on Eddie and awaiting outputs from:

Group 3:

Examines excess number of matings in female lifetime (~4 instead of ~2.5)

Cyan: Same as magenta from group 2 (uniform, new reservoir BV estimation, external pressure from reservoir as function of farm offspring)

Red: number of mating events per timestep = min(available males, females) - Pois(\lambda=2)

Magenta: Same as red but with age preference squared

Group 4:

Cyan: Equivalent to magenta from group 3 but with variables from Loch Fyne

Red: Loch Fyne SSPO lice counts (values in fig 6)

1. Use most appropriate code according to outputs from 1 to run low-middle-high value for:
   1. Number of cages
   2. External pressure
   3. Heritability
2. At the same time flesh out the paper intro and outline
3. Added task: investigate/clarification of BV calculations

Other activity: slide preparation for undergrad talk

Next meeting: 10 am March 24

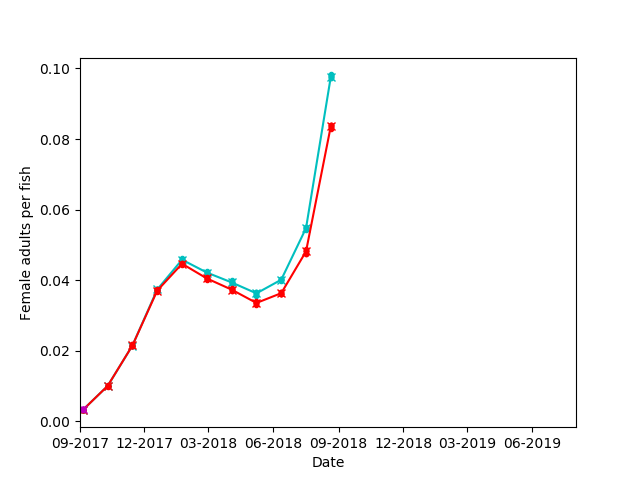
2020-03-10

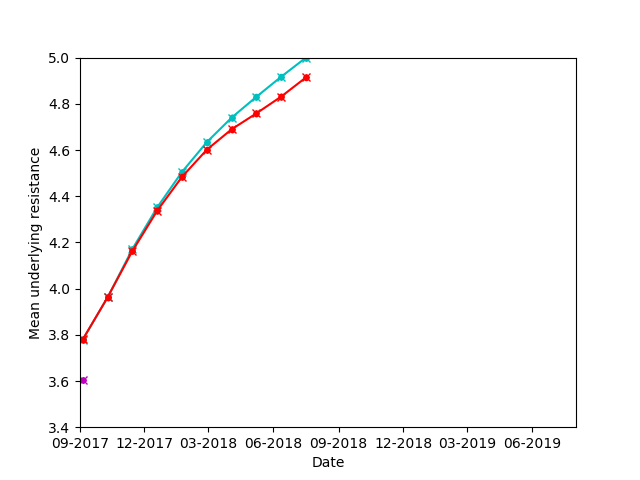
Group 2:

Cyan: Same as magenta from group 1 (preEPIC with minor updates and uniform values for farms)

Red: Equivalent of cyan but with estimation of reservoir BVs as weighted mean from farms, reservoir and influx

(Magenta: Equivalent of red but with external pressure on farms from reservoir as a linear function of the number of offspring created on the farms) queued – the one dot is from a previous iteration





Reassuring to see: The difference between cyan and red is how the BVs are estimated which results in lower resistance BVs as expected and lower resistance -> lower counts.

Actions:

1. Currently queued on Eddie and awaiting outputs from:

Group 3:

Examines excess number of matings in female lifetime (~4 instead of ~2.5)

Cyan: Same as magenta from group 2 (uniform, new reservoir BV estimation, external pressure from reservoir as function of farm offspring)

Red: number of mating events per timestep = min(available males, females) - Pois(\lambda=2)

Magenta: Same as red but with age preference squared

Group 4:

Cyan: Equivalent to magenta from group 3 but with variables from Loch Fyne

Red: Loch Fyne SSPO lice counts (values in fig 6)

1. Use most appropriate code according to outputs from 1 to run low-middle-high value for:
   1. Number of cages
   2. External pressure
   3. Heritability
2. At the same time flesh out the paper intro and outline

Other activity: slide preparation for undergrad talk

Potential hurdle: Eddie queuing time

* + With offspring deletion: (group 1 magenta) ran for 72hrs, outputs to 29/08/2018, maxvmem 25-27.5G
  + Without: group 1 red after 69 hrs, outputs to 26/08/2018, maxvmem 15-17G
    - Group 2 red after 69 hrs, outputs to 31/08/2018, maxvmem ~20G

2020-03-05

Check outlying farm

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| colour | count | Rank count | Model estimate | Sum in | Rank in | Sum out | Rank out | self | Rank self |
| blue x | 0.111808 | 1 | 0.103728 | 1.34E-01 | 4 | 0.15256 | 3 | 4.35E-02 | 2 |
| black x | 0.063067 | 2 | 0.070401 | 1.40E-01 | 2 | 0.253116 | 2 | 6.46E-03 | 7 |
| green x | 0.062512 | 3 | 0.060277 | 1.34E-01 | 3 | 0.32346 | 1 | 1.34E-03 | 9 |
| magenta x | 0.060299 | 4 | 0.054958 | 1.49E-01 | 1 | 0.086735 | 4 | 5.15E-02 | 1 |
| yellow x | 0.050748 | 5 | 0.050638 | 1.16E-01 | 6 | 0.0695 | 6 | 8.36E-03 | 5 |
| cyan x | 0.050394 | 6 | 0.05147 | 7.67E-02 | 9 | 0.081135 | 5 | 9.14E-03 | 4 |
| blue . | 0.048803 | 7 | 0.057883 | 1.20E-01 | 5 | 0.037867 | 7 | 7.82E-03 | 6 |
| green . | 0.043771 | 8 | 0.0492 | 1.04E-01 | 7 | 0.026059 | 9 | 2.87E-02 | 3 |
| red x | 0.034751 | 9 | 0.025681 | 8.96E-02 | 8 | 0.032286 | 8 | 5.29E-03 | 8 |

Ordered model coefficients for standardized data-frame for lice counts in June 2018:

|  |  |
| --- | --- |
| Self-infection | 1.999 |
| Sum proportion arriving | 0.335 |
| Intercept | 0 |
| Number of cages | -0.734 |
| Sum proportion leaving | -1.712 |

Adjusted R-squared: 0.9032

The only surprising thing here is that the lice counts go down with the number of cages. It does make sense to some extent because the input for the proportion of particles arriving at a site does not explicitly take into account the fact that more cages might cause more ”stickiness”. So more cages means that the number of copepodids arriving at a farm from another farm is divided over more cages. However, the number of available fish, which would be greater with the increased number of cages, increases the likelihood of a copepodid attaching to a fish and moving to the chalimus stage. Moreover, each cage has an equal number of copepodids arriving from the wildlife reservoir each day.

This effect should be less important for earlier lice-counts when the ratio of farm to farm infections to external pressure is smaller.

Same in January erratic before that. Look at start date

Ordered model coefficients for standardized data-frame:

|  |  |
| --- | --- |
| Self-infection | 0.641 |
| Sum proportion arriving | 0.580 |
| Intercept | 0 |
| Week stocking starts | -0.089 |
| Weeks until fully stocked | -0.535 |
| Sum proportion leaving | -0.648 |
| Number of cages | -0.736 |
|  |  |

Adjusted R-squared: 0.9612

Adding the week by which all the cages in the farm are stocked explains more of the variation but doesn’t really change anything otherwise.

Removing the number of cages greatly reduces R-squared (0.4801) the only major change is that the abs. value of the coefficient for the week at which stocking starts increases.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| cages | 6 | 4 | 8 | 12 | 9 | 9 | 8 | 9 | 9 |
| Stock time | 6 | 4 | 4 | 4 | 9 | 9 | 4 | 9 | 9 |
| Stock start | 9 | 4 | 9 | 9 | 9 | 0 | 4 | 9 | 4 |

In January 2018:

Ordered model coefficients for standardized data-frame:

|  |  |
| --- | --- |
| Sum proportion arriving | 0.661 |
| Self-infection | 0.404 |
| Intercept | 0 |
| Week stocking starts | -0.189 |
| Sum proportion leaving | -0.449 |
| Weeks until fully stocked | -0.576 |
| Number of cages | -0.712 |
|  |  |

Adjusted R-squared: 0.9805

In December 2017:

Ordered model coefficients for standardized data-frame:

|  |  |
| --- | --- |
| Sum proportion arriving | 0.165 |
| Self-infection | 0.070 |
| Intercept | 0 |
| Sum proportion leaving | -0.070 |
| Number of cages | -0.340 |
| Weeks until fully stocked | -0.652 |
| Week stocking starts | -0.705 |
|  |  |

Adjusted R-squared: 0.9691

In November 2017:

Ordered model coefficients for standardized data-frame:

|  |  |
| --- | --- |
| Self-infection | 0.665 |
| Sum proportion arriving | 0.240 |
| Intercept | 0 |
| Number of cages | -0.200 |
| Weeks until fully stocked | -0.376 |
| Sum proportion leaving | -0.664 |
| Week stocking starts | -0.866 |
|  |  |

Adjusted R-squared: 0.9902

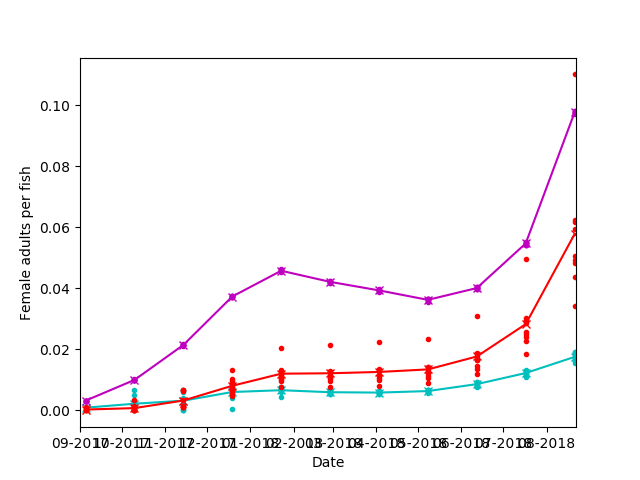
So earlier in the epidemic start date is more important and number of cages is less negative as the ratio of farm to farm infections to external pressure is smaller.

Uniform simulation

Self infection in loch Fyne connectivity matrix ranges from 0.001 to 0.05, sum in 0.02-0.32 and sum out 0.08-0.15.

Input for the uniform simulations is 0.025 and therefore 0.225 for sum in and sum out. Stocking starts immediately for all cages (=0) and the number of cages is 8.

New simulations with 0.012 for connectivity matrix and more memory.



F,{7714c6cc-2cb3-44a2-b3f2-a1cf8211cc3a}{107},10.54167,7.916667

Cyan: pre-EID

Red: new equivalent of pre-EID including not deleting farm to farm infection

Magenta: Uniform equivalent of red with connectivity = 0.012 for all farms

Potential memory drain:

Line 423 F4Fyne offspring.append all other simulations line 184 offspring.drop or just ran out of memory with no error.

Line # Mem usage Increment Line Contents

================================================

5 49.0 MiB 0.0 MiB @profile

6 def func1():

7 49.0 MiB 0.0 MiB np.random.seed(5643)

8 52.0 MiB 3.0 MiB df = pd.DataFrame({'a':np.random.normal(0,1,100000), 'b':np.random.normal(0,1,100000)})

9 57.1 MiB 5.1 MiB df = df.drop(df[(df['a']<-0.5)&(df['b']<-0.5)].index)

10 57.1 MiB 0.0 MiB return df

Line # Mem usage Increment Line Contents

================================================

5 50.3 MiB 0.0 MiB @profile

6 def func2():

7 50.3 MiB 0.0 MiB np.random.seed(5643)

8 52.1 MiB 1.8 MiB df = pd.DataFrame({'a':np.random.normal(0,1,100000), 'b':np.random.normal(0,1,100000)})

9 57.1 MiB 5.1 MiB df.drop(df[(df['a']<-0.5)&(df['b']<-0.5)].index, inplace=True)

10 57.1 MiB 0.0 MiB return df

Line # Mem usage Increment Line Contents

================================================

5 50.9 MiB 0.0 MiB @profile

6 def func1():

7 50.9 MiB 0.0 MiB np.random.seed(5643)

8 52.0 MiB 1.0 MiB df = pd.DataFrame({'a':np.random.normal(0,1,100000), 'b':np.random.normal(0,1,100000)})

9 57.1 MiB 5.2 MiB df2 = df.drop(df[(df['a']<-0.5)&(df['b']<-0.5)].index)

10 57.1 MiB 0.0 MiB del df

11 57.1 MiB 0.0 MiB df = df2

12 57.1 MiB 0.0 MiB del df2

13 57.1 MiB 0.0 MiB print(len(df.a))

14 57.1 MiB 0.0 MiB return df

69008 30992

49936 50064

30928 69072

279 99721

Filename: ./model/memdrain.py

Line # Mem usage Increment Line Contents

================================================

5 66.8 MiB 0.0 MiB @profile

6 def func1():

7 66.8 MiB 0.0 MiB np.random.seed(5643)

8 67.5 MiB 0.8 MiB df = pd.DataFrame({'a':np.random.normal(0,1,100000), 'b':np.random.normal(0,1,100000)})

9 67.5 MiB 0.0 MiB df2 = pd.DataFrame(columns=['a','b'])

10 67.5 MiB 0.0 MiB bins = [-3, -0.5, 0, 0.5, 3]

11 78.1 MiB 10.6 MiB for i in range(4):

12 77.9 MiB -0.2 MiB df2 = df2.append(df[(df.a > bins[i]) & (df.a <= bins[i+1])].copy())

13 78.1 MiB 0.2 MiB df = df.drop(df[(df.a > bins[i]) & (df.a <= bins[i+1])].index)

14 78.1 MiB 0.0 MiB print(len(df.a), len(df2.a))

15 78.1 MiB 0.0 MiB return df2

(mypython3\_6) [dlipschu@node1h18 slm]$ python -m memory\_profiler ./model/memdrain.py

100000 30992

100000 50064

100000 69072

100000 99721

Filename: ./model/memdrain.py

Line # Mem usage Increment Line Contents

================================================

5 66.8 MiB 0.0 MiB @profile

6 def func1():

7 66.8 MiB 0.0 MiB np.random.seed(5643)

8 67.6 MiB 0.8 MiB df = pd.DataFrame({'a':np.random.normal(0,1,100000), 'b':np.random.normal(0,1,100000)})

9 67.6 MiB 0.0 MiB df2 = pd.DataFrame(columns=['a','b'])

10 67.6 MiB 0.0 MiB bins = [-3, -0.5, 0, 0.5, 3]

11 73.3 MiB 5.7 MiB for i in range(4):

12 73.3 MiB 0.0 MiB df2 = df2.append(df[(df.a > bins[i]) & (df.a <= bins[i+1])].copy())

13 #df = df.drop(df[(df.a > bins[i]) & (df.a <= bins[i+1])].index)

14 73.3 MiB 0.0 MiB print(len(df.a), len(df2.a))

15 73.3 MiB 0.0 MiB return df2

Line # Hits Time Per Hit % Time Line Contents

==============================================================

4 @profile

5 def func1():

6 1 12 12.0 0.0 np.random.seed(5643)

7 1 11881 11881.0 17.2 df = pd.DataFrame({'a':np.random.normal(0,1,100000), 'b':np.random.normal(0,1,100000)})

8 1 1188 1188.0 1.7 df2 = pd.DataFrame(columns=['a','b'])

9 1 2 2.0 0.0 bins = [-3, -0.5, 0, 0.5, 3]

10 5 9 1.8 0.0 for i in range(4):

11 4 19041 4760.2 27.6 df2 = df2.append(df[(df.a > bins[i]) & (df.a <= bins[i+1])].copy())

12 4 35923 8980.8 52.0 df = df.drop(df[(df.a > bins[i]) & (df.a <= bins[i+1])].index)

13 4 985 246.2 1.4 print(len(df.a), len(df2.a))

14 1 1 1.0 0.0 return df2

Line # Hits Time Per Hit % Time Line Contents

==============================================================

4 @profile

5 def func1():

6 1 11 11.0 0.0 np.random.seed(5643)

7 1 11034 11034.0 34.4 df = pd.DataFrame({'a':np.random.normal(0,1,100000), 'b':np.random.normal(0,1,100000)})

8 1 1128 1128.0 3.5 df2 = pd.DataFrame(columns=['a','b'])

9 1 2 2.0 0.0 bins = [-3, -0.5, 0, 0.5, 3]

10 5 10 2.0 0.0 for i in range(4):

11 4 19282 4820.5 60.1 df2 = df2.append(df[(df.a > bins[i]) & (df.a <= bins[i+1])].copy())

12 #df = df.drop(df[(df.a > bins[i]) & (df.a <= bins[i+1])].index)

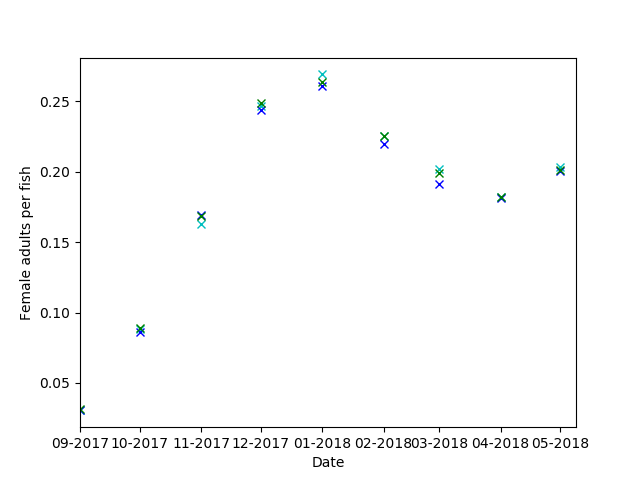
13 4 633 158.2 2.0 print(len(df.a), len(df2.a))

14 1 1 1.0 0.0 return df2

So deleting the offspring that have been allocated to cages as I go saves neither time nor memory, in fact it costs both time and memory => DELETE IT!

2020-02-17

First cage of first farm is no longer growing faster than the other cages (see figure below) and all the simulations mentioned in the previous notes are either running or queuing on Eddie. Scripts have been written for the accompanying figures and these will be added to the notes as soon as there is enough data.



2020-02-06

Simulations run/running:

Group 1:

Cyan: Pre-EPIC with continuous treatment

Red: Equivalent of pre-EPIC with some updates including hatching time as a function of temperature and external influx into the reservoir

Magenta: Equivalent of red but with uniform values for all farms e.g. equal number of cages, temperature, distance between farms etc (a lot of values were still set for Loch Fyne previously)

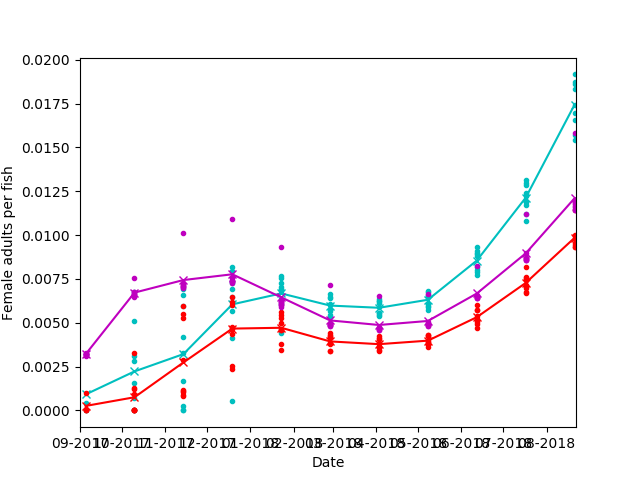


Fig. 1. Group 1 Lice counts

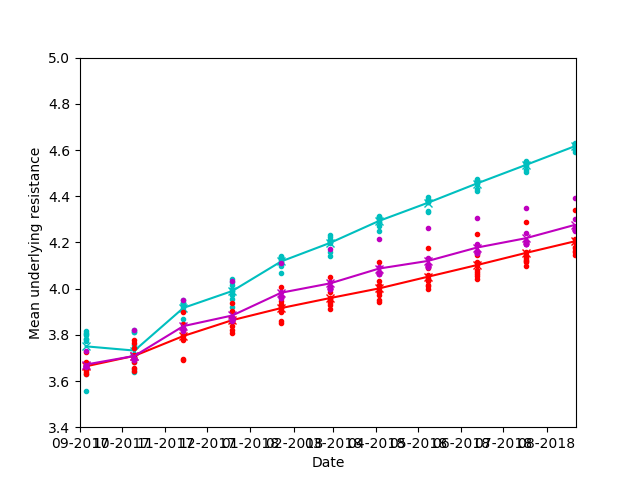


Fig.2. Group 1 BVs

It is reassuring to see:

* Fig1 - That the updates did not drastically alter the shape of the distributions (red vs cyan)
* Fig2 - That including an external influx in the reservoir slows down evolution of resistance in the farms as we might expect
* That less resistance (fig2) gives lower lice counts (fig1) as we would expect
* Changing only the farm characteristics (magenta vs red) affects the lice counts distribution (fig1) but not so much the BVs (fig2)
* We do see less variation between the farms in the uniform simulation (magenta) BUT there seems to be an outlier every month. It is in fact the same farm every month (see fig3) and as it is farm 1 I believe that it may be an artefact in the way the values are aggregated for output and am currently investigating this

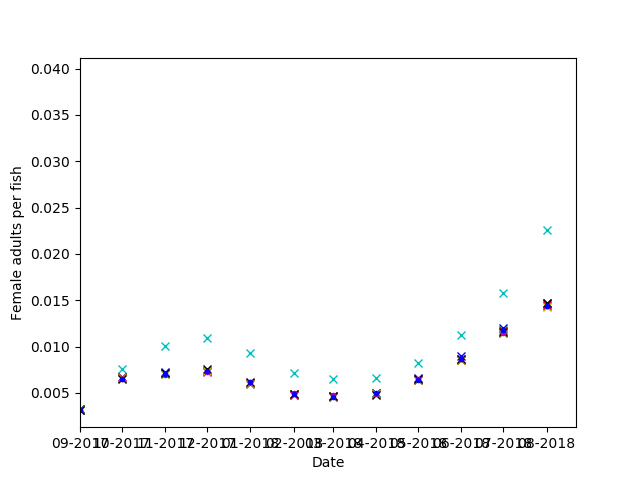


Fig. 3. Group 1 lice counts per farm

Group 2:

Cyan: Same as magenta from group 1 (preEPIC with minor updates and uniform values for farms)

Red: Equivalent of cyan but with estimation of reservoir BVs as weighted mean from farms, reservoir and influx

(Magenta: Equivalent of red but with external pressure on farms from reservoir as a linear function of the number of offspring created on the farms) was running but mysteriously vanished from Eddie(!?)

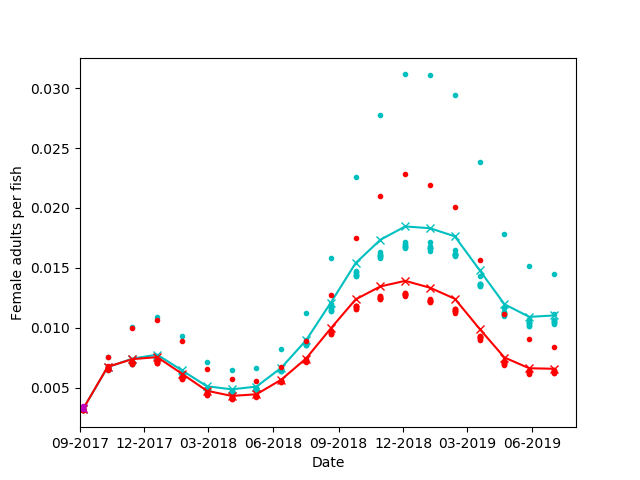


Fig. 4. Group 2 lice counts

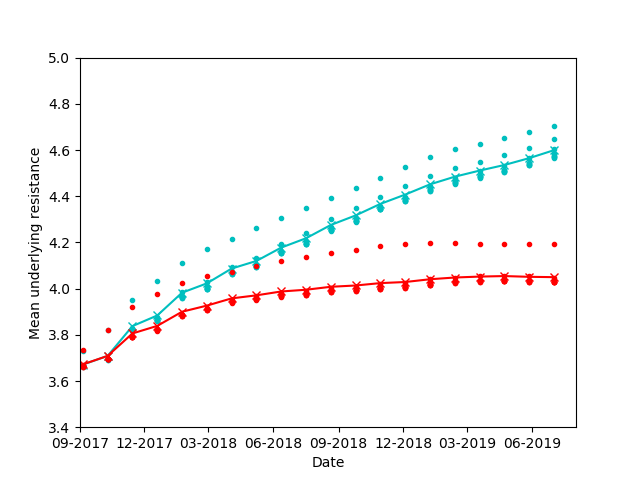


Fig. 5. Group 2 BVs

Reassuring to see:

* Lower resistance (fig5) -> lower counts (fig4)
* The cyclical shape of the counts (fig4 seems consistent with real lice counts (see fig6&7 and <https://veterinaryrecord.bmj.com/content/vetrec/151/25/753.full.pdf>)
* Cyan BVs were sampled from 

Red BVs µr,t = p\*3.5 + (1-p)\*(x\*µr,t-1 + (1-x)\*µf,t-1, \sigma^2)

where x is the proportion off offspring created on the reservoir in the loch (1-x on the farms in the loch)

This pulls the BVs down (fig 5)

To be examined:

* As mentioned for group 1, currently examining the higher values for farm 1

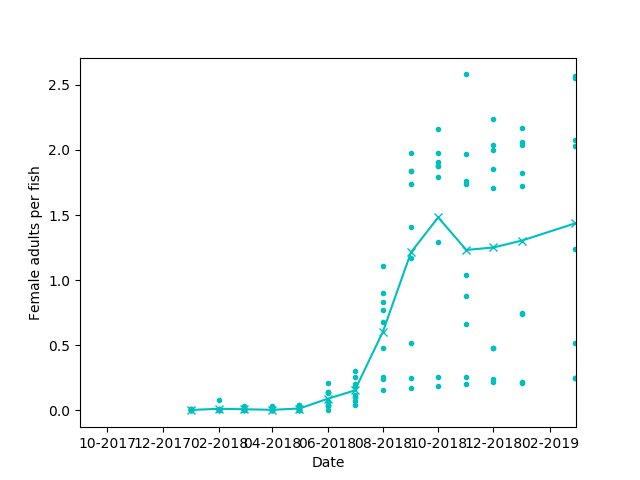


Fig 6 SSPO lice counts for Loch Fyne

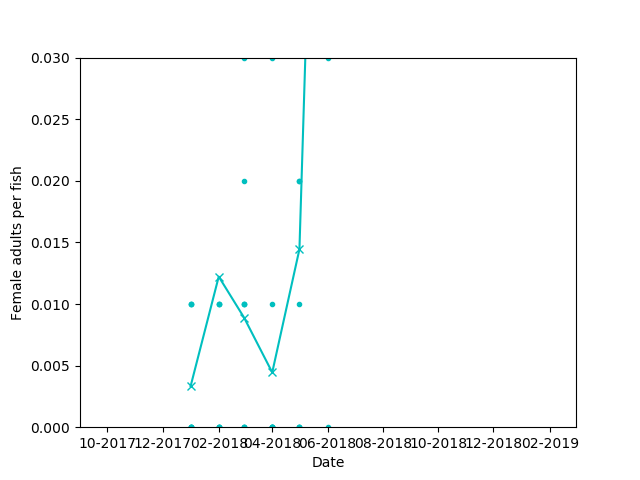


Fig 7 Figure 6 zoomed in

Actions:

1. Debugging of farm 1 outlier
2. Removed from queue as debugging needs to happen first but to be seen next time:

Group 3:

Examines excess number of matings in female lifetime (~4 instead of ~2.5)

Cyan: Same as magenta from group 2 (uniform, new reservoir BV estimation, external pressure from reservoir as function of farm offspring)

Red: number of mating events per timestep = min(available males, females) - Pois(\lambda=2)

Magenta: Same as red but with age preference squared

Group 4:

Cyan: Equivalent to magenta from group 3 but with variables from Loch Fyne

Red: Loch Fyne SSPO lice counts (values in fig 6)

2020-01-28

See git for step 1.

Step 2:

Using a small simulation with the exact outputs I needed wasn’t possible because if the starting population is too small the epidemic doesn’t start. So I did some back calculations based on existing outputs.

I’m not including the resulting graphs because I want to talk you through them but my conclusion is that it’s plausible enough to go ahead.

Also while going through the simulations with a fine comb I realised that we needed to have external pressure as a function of the number of offspring created on the farms instead of a constant that’s why we weren’t seeing the growth we needed. I am also going to run this now as part of step 3.

2020-01-21

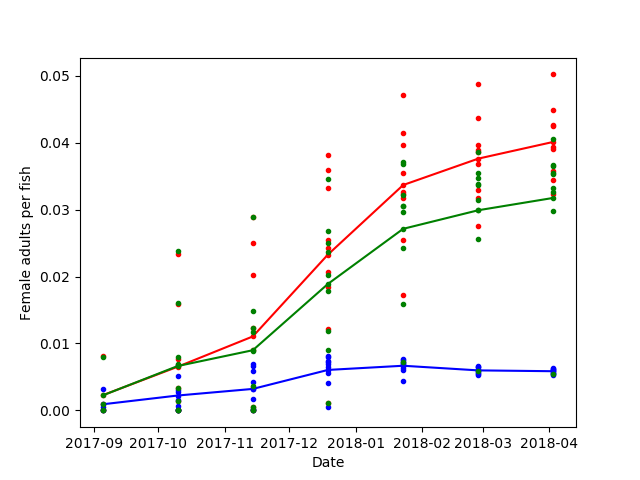
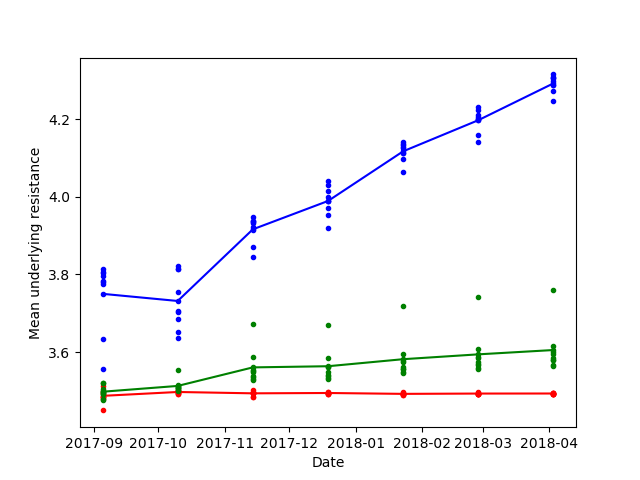
Currently running:

* Simulation with explicit reservoir – BV outputs
* Updated pre-epic simulation (BVs from bimodal distribution) - baseline

Next steps:

1. Make sure the baseline model is fully debugged
2. Check BV outputs are roughly consistent with estimated normal distribution with weighted mean
3. Start sensitivity analysis (plots will look like the following)
4. Finish developing simulation for single gene and repeat sensitivity analysis with it
5. Repeat connectivity and treatment strategy analysis from epidemics poster with a choice of parameters stemming from sensitivity analysis
6. Write paper

Next meeting Friday 31/1 to discuss outcomes of 1 & 2



2020-01-16

Jess notes

Key upcoming admin actions: Skype/call in near future, arrangement of in-person meetings

Request to Debby: update on current progress and milestones:

Progress on milestones from previous period (from below):

1. Update?? work on single-gene code version
2. Done Poster to Ant and Jess by Nov 25
3. Done Amended poster to printer by Nov 27
4. Update?? Start the addition runs on Roslin nodes
5. Done Amend figures for poster

Other progress?

Milestones for Jan 16-25

1. (Proposed by Jess) Write update on single-gene output
2. (Proposed by Jess) Update on figures from current runs
3. (Proposed by Jess) Update experiment list, prioritise
4. (Proposed by Jess) Generate paper outline, with key results identified

2019-11-21

Skype meeting Debby/Jess

Progress on milestones from previous week:

1. Done: Meeting with Andy law, get ahold of nodes for computation
2. Done: Start computation with three levels of connectivity
3. Worked on? In parallel: work on single-gene code version

Milestones for Nov 21-Nov 28:

1. Poster to Ant and Jess by Nov 25
2. Amended poster to printer by Nov 27
3. Start the addition runs on Roslin nodes
4. Amend figures for poster

2019-11-14

22 of the 70 jobs are now running. The queued ones have now been submitted in reverse order on the new nodes too.

The simulation has been modified and submitted (with and without BV generating function) for a single gene with complete dominance.

2019-11-13

Andy Law confirmed that I can run my simulation on the old Eddie nodes ahead of the conference as they won’t be decommissioned in that time and it should avoid the queues on the new nodes. He is looking into the possibility of me accessing the Roslin nodes.

14 batches of 5 iterations each have been sent to the old nodes:

A homogeneous network where all farms are 7 days apart and all values in the connectivity matrix is:

* Low 0.001
* Medium 0.025
* High 0.05

The low and high values correspond roughly to the lowest and highest values, respectively, in the Loch Fyne connectivity matrix.

For each of these 3 treatment scenarios were simulated:

* Treatment after SSPO thresholds reached
* Pre-emptive treatment
* Treatment after SSPO thresholds reached in all but one farm

These same scenarios were examined in a simulation using the Loch Fyne connectivity matrix. In the scenario where only one farm treats pre-emptively the following farms were chosen:

* Most connected overall
* Least connected overall
* The strongest source (but doesn’t receive much)

Overall that’s 70 jobs, 17 of which are currently running and providing outputs and 58 are queued.

2019-11-11

Meeting at Stirling, discussion of current state and forward plan.

Milestones for week November 11-18th:

1. Meeting with Andy law, get ahold of nodes for computation
2. Start computation with three levels of connectivity
3. In parallel: work on single-gene code version

Poster to be sent to Ant and Jess by 25th morning at latest

Poster to printer by Wednesday the 27th

2019-11-06

I removed the BV generating function because it’s better to have a slow but functioning model than one that crashes due to memory issues. I have set a test to run on the new eddie nodes but they are still queued which may be due to not all nodes being set up yert or ongoing issues with the new nodes (old nodes are being decommissioned and therefore not to be used).

I am also looking through the data wrangling I had started doing with the norwegian data and what is still required to use it.

2019-08-27

Increased code efficiency see github

2019-08-21

The number of times adult females are mating in their lifetime seems to be around 4-5 times instead of 2-3. At the moment to reflect the males’ preference for younger females I’ve set the probability of a particular female being chosen for mating as 1-(her age/sum  
(all available adult female ages)+1), I could accentuate this preference by taking their age\*\*2. Also at the moment the number of matings is limited only by the number of available mates, I could add weights or stochasticity that would reduce this but this would exacerbate slower increase we observe in the model vs reality.

2019-08-07

Before the crash I had changed the number of days hatching to be temperature dependent rather than seasonal which is more in line with current understanding and could potentially explain why we see such a sharp increase in May 2018. However, I didn’t get any useful results yet because the resulting simulation was so slow. I had done some profiling and was in the process of making the simulation more efficient. I have now updated the model and I’m waiting for Andy Law to tell me when it’s safe for me to run the new simulation given the current Eddie upgrades that are happening at the moment. Tomorrow I plan to do some cleaning up and update all the different versions of the model on git.

2019-05-23

Looking at when EMB was administered and the effect it’s had (see MFlice.xlsx). ~ Entire loch treated in Dec, Feb & May. Mostly effective in Dec and Feb (although counts are already low) but most farms experience increases in lice counts following treatment in May. Is this true in other Lochs?

This would suggest perhaps a low external pressure but very high heritability I.e. much more importance on within Loch/cycle dynamics than external pressure. -> running variations along the theme of ext. Pressure 100 and h^ 0.45

Upcoming IDE meeting in Edinburgh: would like to present poster -> could this be something to explore for it

Also upcoming deadline 7/6 for Epidemics7 (in December in US). Any thoughts on this conference?

The biggest question marks I can see with this model at the moment are the external pressure, the distribution of resistance and it’s genetic architecture (see params.docx). Does it make sense to everyone that the first paper would then examine which combination of these could reflect reality and what impact different combinations have on the evolution of resistance?

It makes sense to me because once this is established in a paper I can then go on to use the model to look at the effect of different types of treatments/co-operation as intended.

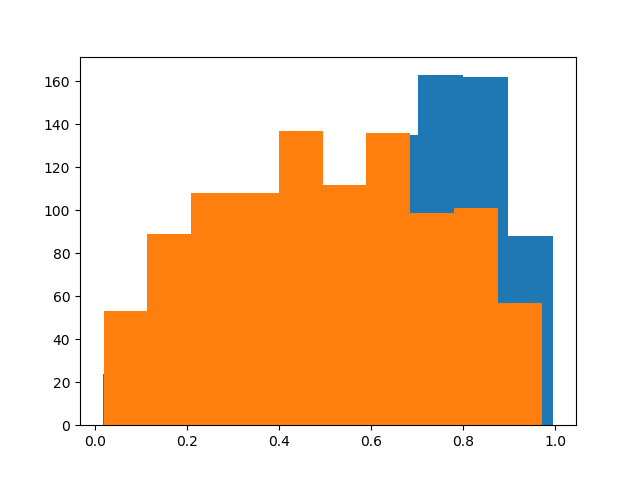
2019-05-22

I’ve been a complete numpty and the reason none of the changes were having much effect was because the function that samples BVs from both distributions (see params.docx, a\_x) was still set to constants instead of taking the input parameter values. I’ve now fixed it and I’m running mu 0.5 h^2 0.33, mu 0.5 h^2 0.2 and mu -0.5 h^2 0.2.

2019-05-14

I had changed the mean for the underlying genetic value of EMB resistance from –3.5 to –0.5 but there are still too many lice dying. The distribution of the resulting phenotype of susceptibility to EMB (1 – resistance) is still highly skewed with most lice being highly susceptible to EMB (blue distribution below).

I have now changed the mean to 0 (orange distribution below) and am running it with an external pressure of 250 as that is more consistent with the lice counts at the start of the season.



2019-05-03

Increasing heritability from 0.2 to 0.33 in simulation with external pressure made very little difference to lice counts in the first year (MFlice.csv).

Also, upon closer examination the effect of EMB in the simulation is too drastic and there’s an overestimation when the lice population had time to recover. I’m going to run the external pressure=500 simulation with much higher frequency of resistance in the base population.

2019-04-30

MFlice.csv

Shows Monthly lice counts from SSPO lice reports. Pre-2018 they’re loch-wide values, afterwards there’s farm values.

Next columns show model outputs with external pressure of 500/750/1000 per day.

The 4 initial months (pre-2018) show slight overestimates compared to loch-wide count of “0”. Post June there’s a marked underestimation in the model outputs compared to the reports (least pronounced with external pressure 1000). Especially as the course of EMB administered in May does not seem to have as big an impact as previous courses.

I also noted that there doesn’t seem to be any lice counts>0.5 before the periods of treatment. Lice counts do increase to 0.5 (and higher) after June but no more oral treatments are reported.

For now, I am running the model with external pressure 1000 again but with increased heritability of resistance. I’m hoping to get lice counts that are close enough to say that our model parameters are feasible ‘enough’. Then I will do some kind of sensibility analysis at the end to see how much decisions re parameters affect our results. It seems like the best I can do with what I’ve got without spending forever on this, but I’m open to suggestions.

In the meanwhile, I will re-write the model with different modes of inheritance to examine the impact of modes of inheritance.

2019-04-05

The two big question marks in terms of model parameters are the external pressure and the heritability etc for treatment resistance.

External pressure

The external pressure is responsible for starting the epidemic so it is fair to assume that the chosen value is reasonable if we can match lice counts within the first few months before treatment.

Unfortunately we only have Loch-wide lice counts for the last quarter of 2017 and they’re all 0. But treatments are given for lice counts> 0.5. The first treatments begin in October.

The first cages are stocked in August but not all farms and not all cages are stocked. So I’m looking for an external pressure which would have lice counts>0.5 by 1/10/17 in the first stocked cages but round down to a loch wide value of approx.. 0 if we give a zero lice value to all the empty cages.

Example output ext pressure 500:

Farm Date mean s.d.

1 2017-09-01 00:00:00 nan nan

2 2017-09-01 00:00:00 0.027275 0.0

3 2017-09-01 00:00:00 nan nan

4 2017-09-01 00:00:00 nan nan

5 2017-09-01 00:00:00 nan nan

6 2017-09-01 00:00:00 0.00311 0.00519910088381

7 2017-09-01 00:00:00 0.0 0.0

8 2017-09-01 00:00:00 nan nan

9 2017-09-01 00:00:00 0.0 0.0

Submitting array jobs with external pressure 500, 750, 1000 (30 each)

EMB resistance

* Do some more reading and talk to Armin
* Once the external pressure value is set, run simulation over several production cycles and see if mean resistance curve over time matches literature

Note in case similar error occurs again:

Could not find platform independent libraries <prefix>

Could not find platform dependent libraries <exec\_prefix>

Consider setting $PYTHONHOME to <prefix>[:<exec\_prefix>]

Fatal Python error: Py\_Initialize: Unable to get the locale encoding

ImportError: No module named 'encodings'

Probably due to default Anaconda being upgraded. Check module avail, conda info –envs and python –V. The solution is creating a new mypython venv with the right version of python for the default version of anaconda (it seems to go to default even if you specify the version when loading the anaconda module). Make sure you update the versions in the shell script.