Supplementary Information I : An application of CaN modelling to the Barents Sea trophic network

Drouineau, Planque, Mullon

5/2/2021

Contents

1 Introduction

We present here an example of RCaNmodel study for the Barents Sea ecosystem. The CaN model has the same structure as the model presented in the main text of the article: same trophic structure, same trophic parameters, same set of time series of observations (Lindstrøm, Planque, and Subbey (2017)). Due to data access restriction, the values in the observation times series were slightly modified and do not correspond to the original measurements. Most of the original data can be provided on demand.

The network is illustrated in figure 1.

```
knitr::include_graphics("BarentsNetwork.png")
```

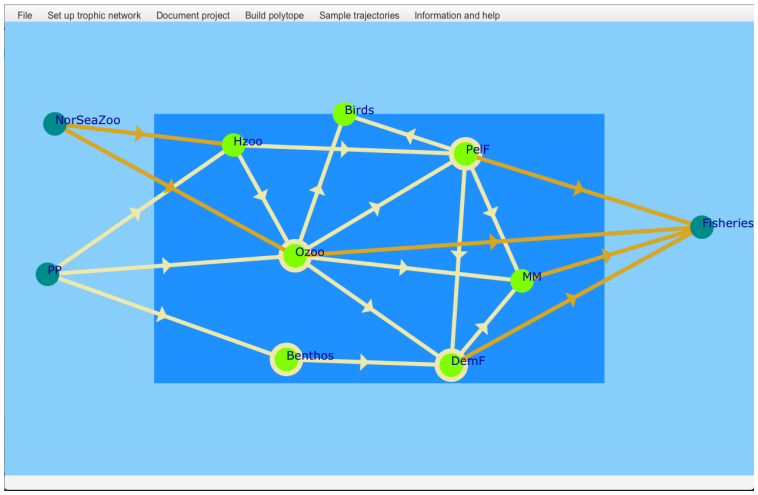


Figure 1: The Barents Sea tropic network. The dashed-line rectangle separate components that are within the model domain (herbivorous and omnivorous zooplankton, benthos, pelagic and demersal fish, mammals and birds) from those that are outside (primary producers, Norwegian Sea zooplankton and fisheries)

The RCaN-file, constructed wit RCaNconstructor, is attached to this supplementary information. The R-Markdown file, used to generate this supplementary material, is also attached.

We first provide information for the installation of RCaNmodel and RCaNconstructor. We then provide a short description of the content of the Barents Sea RCaN-file. Finally, we go through the main steps of the modelling which follow the RCaN file construction: (a) build the polytope, (b) check the polytope status, (c) if required, revise the model by adding and removing constraints, (d) sample the polytope, (d) analyze the samples.

2 Installing RCaNmodel

RCaNmodel requires R (version 3.2+) to be installed.

The R library RCaNmodel can be downloaded from https://github.com/inrae/RCaNmodel. This is done by following installation instructions given in the ReadMe file of the package .

```
# if (!require("RCaNmodel")) {
#  require(devtools)
#  devtools::install_git(
#    "https://github.com/inrae/RCaNmodel.git",
#    subdir="RCaNmodel",
#    dependencies = TRUE)
#  }
```

Then, the library can be loaded

```
library(RCaNmodel) #the RCaNmodel package itself
```

3 The RCaNmodel file for the Barents Sea ecosystem

The food web structure, biological parameters, observational data-series and model constraints are all provided in the RCaN file in the .xlsx format. A RCaN file can be edited using a spreadsheet editor (e.g. libre office,

Table 1: Components

Component	Inside	AssimilationE	Digestibility	OtherLosses	Inertia	Satiation	RefugeBiomass
PP	0	0.00	0.65	0.00	0.00	0.0	0.0000
Hzoo	1	1.00	0.90	8.40	7.58	128.0	0.2300
Ozoo	1	1.00	0.90	5.50	3.10	42.0	0.1300
Benthos	1	0.94	0.60	1.50	0.74	25.2	0.6600
PelF	1	0.90	0.90	2.85	0.90	13.5	0.0250
DemF	1	0.93	0.85	1.65	0.25	5.5	0.0230
MM	1	1.00	0.00	5.50	0.11	10.9	0.0034
Birds	1	0.84	0.00	60.00	0.81	123.0	0.0001
NorSeaZoo	0	0.00	0.00	0.00	0.00	0.0	0.0000
Fisheries	0	0.00	0.00	0.00	0.00	0.0	0.0000

open office, MS Excel) or with the RCaNconstructor. The use of RCaNconstructor is recommended as it ensures that names of model entities (species, fluxes and observations) are consistent across the different worksheets of the RCaN-file.

```
NAMEFILE <- 'annexC_BarentsSeaDummy.xlsx'
```

Importing an RCaN-file in the R environment and building the corresponding polytope are done with one single RCaNmodel command:

```
POLYTOPE <- buildCaN(NAMEFILE)
```

The buildCaN function reads four worksheets from the RCaN-file and uses the information to construct the model, i.e. build the corresponding polytope. The information in these four worksheets pertains to components, fluxes, observational data-series and constraints. These are as follows:

3.0.1 Components

Components correspond to the tropho-species in the systems (figure 1). They are specified in the "Components & input parameter" of the RCaN file, with their names (column Component), a flag to indicate whether they are inside or outside the model domain (column Inside) and their biological parameters. The biological parameters are used in the CaN master equation (which expresses mass conservation and the relationship between fluxes and biomass) and to implement implicit constraints (inertia, satiation and refuge biomass).

The above information is stored in the dataframe 'components_param' in the POLYTOPE object:

```
kable(POLYTOPE$components_param[c(1:8)], caption='Components') %>%
  kable_styling(latex_options = c("striped", "scale_down"), font_size = 8)
```

3.0.2 Fluxes

Fluxes correspond to the transfer of biomass between food-web components (i.e., arrows in figure 1). They are specified in the "Fluxes" worksheet of the RCaN-file, identified by a name (column Flux), a source component (column From) and a sink component (column To) and a flag that indicates whether the flux is trophic or not (column Trophic). Non-trophic fluxes are typically import/export to/from the model domain and fisheries. The fluxes information is stored in the dataframe 'flux_def' in the POLYTOPE object.

```
kable(POLYTOPE$fluxes_def, caption='Fluxes') %>%
  kable_styling(latex_options = c("striped", "scale_down"), font_size = 8)
```

3.1 Observational data-series

Observational time-series in the Barents Sea are derived from monitoring programs, mainly for plankton and fish species. In this example, the original data compiled by the ICES Working Group on the Integrated

Table 2: Fluxes

Flux	From	To	Trophic
PP_Ozoo	PP	Ozoo	1
PP_Hzoo	PP	Hzoo	1
PP_Benthos	PP	Benthos	1
Hzoo_Ozoo	Hzoo	Ozoo	1
Hzoo_PelF	Hzoo	PelF	1
Ozoo_Ozoo	Ozoo	Ozoo	1
Ozoo_PelF	Ozoo	PelF	1
Ozoo_DemF	Ozoo	DemF	1
Ozoo_MM	Ozoo	MM	1
Ozoo_Birds	Ozoo	Birds	1
Benthos_Benthos	Benthos	Benthos	1
Benthos_DemF	Benthos	DemF	1
PelF_PelF	PelF	PelF	1
PelF_DemF	PelF	DemF	1
PelF_MM	PelF	MM	1
PelF_Birds	PelF	Birds	1
DemF_DemF	DemF	DemF	1
DemF_MM	DemF	MM	1
NorSeaZoo_Hzoo	NorSeaZoo	Hzoo	0
NorSeaZoo_Ozoo	NorSeaZoo	Ozoo	0
PelF_Fisheries	PelF	Fisheries	0
DemF_Fisheries	DemF	Fisheries	0
MM_Fisheries	MM	Fisheries	0
Ozoo_Fisheries	Ozoo	Fisheries	0

Table 3: Observations

Year	Prod_Sat	Hzoo_Biomass	Ozoo_Biomass	Pelagics	Demersals	Pel_landings	Dem_landings	Mammals_landings	Shrimp_Landings	Other2Cod	Amphipods2Cod	Krill2Cod	Shrimp2Cod	Capelin2Cod	Herring2Cod	Polarcod2Cod	Cod2Cod	Haddock2Cod	Redfish2Cod	Ghalibut2Cod	BW2Cod
1988	NA	14679.44	43529.139	829.9770	1577.071	0.0000000	845.6424	0.1593497	51.89703	399.3009	540.78713	877.76065	333.23406	16.99053	153.42396	225.03596	44.081667	83.2446088	2.345032	79.23291	306.76467
1989	NA	62470.87	17905.306	1726.2088	1357.936	0.2164889	552.7722	1.2942835	68.97331	649.7203	90.90174	752.98143	368.49970	335.75266	501.87841	207.41643	44.773220	50.9543234	58.819461	19.05301	124.77969
1990	NA	21415.93	6374.478	4242.3886	1527.592	0.1093513	352.6741	0.7716795	87.60667	1341.2284	83.36440	275.38828	255.44702	221.01636	425.12369	192.25748	26.684430	92.1895150	122.435256	99.34333	468.98040
1991	NA	22056.29	24800.401	8148.0740	1844.097	968.4682286	458.3139	0.9755945	80.10714	760.6021	724.70898	205.42631	867.17593	24.92297	357.67837	1341.59229	75.444748	286.4327055	258.478957	414.81162	1044.11252
1992	NA	24294.86	15360.849	3807.2990	2275.857	1177.3058901	717.5540	1.2241822	75.13312	893.0536	1628.43520	485.05147	12.06488	596.36880	473.19781	245.15352	64.374488	65.5712568	76.795792	79.36516	766.48558
1993	NA	28289.67	21933.883	1554.3385	2693.084	637.5448754	863.1006	2.3523738	61.31575	737.1774	943.46484	1258.55999	362.24186	875.23932	571.44250	952.36509	385.170942	540.8406118	263.504271	386.70527	566.22679
1994	NA	72068.36	42414.299	662.6870	3003.467	6.2629768	1111.9482	2.7093421	29.89385	611.2064	1118.45544	1619.46643	328.25391	230.20538	1058.97949	284.63977	49.192463	62.2880251	241.084504	55.41312	260.80771
1995	NA	63593.64	31174.408	271.9579	2380.390	23.4889151	1047.0975	1.7778525	26.81543	810.7648	854.89931	1270.38741	261.02661	194.52965	169.97748	281.88827	441.836793	148.3998164	530.326916	60.16798	81.03442
1996	NA	22666.31	45059.460	514.1673	2080.107	21.2127060	1092.1048	2.6101733	37.74192	588.3792	1043.96389	266.29732	1667.81641	299.28171	51.65745	339.74138	249.806645	116.8362384	173.563310	316.18921	404.93728
1997	NA	46544.98	29820.492	1087.5594	1794.432	8.0939748	1178.0738	3.4422745	36.17519	427.4262	1190.12289	1201.91573	122.73204	24.85079	151.92225	677.06590	101.103704	106.6363364	23.786316	93.15501	376.20652
1998	1244929.4	63707.83	27814.791	2175.9761	1581.752	4.8378310	923.8214	2.9984123	58.93368	401.3099	455.41909	673.24215	299.37562	61.61713	212.57513	583.19625	6.132485	51.4280058	31.784426	73.99408	151.80999
1999	1190620.1	25338.07	27923.229	3347.8926	1450.155	124.3468529	736.9683	2.8012324	76.76087	376.7606	441.36435	329.36581	256.77386	420.71374	65.74522	935.21809	56.017263	47.4242934	70.454166	35.84028	254.82607
2000	1402987.9	25281.52	26322.083	5812.6958	1710.152	485.0358623	667.4603	2.6190307	87.69210	379.6319	254.10452	400.20194	838.66962	399.41288	1491.05627	782.92884	94.988336	134.0517768	23.688050	92.81405	391.15635
2001	1524795.2	40359.65	29704.193	3749.7890	1837.384	666.9092419	707.9541	3.3018101	61.11899	692.7480	141.98186	397.26991	112.61351	48.03670	767.32481	1216.22820	128.706821	8.9907982	110.790917	112.31523	49.46736
2002	1801364.5	59295.66	17107.517	1773.3982	2234.564	742.5967468	789.3222	3.1008171	63.77671	379.9523	470.98928	585.61517	504.31015	452.62267	443.75399	1208.52185	145.199224	13.8078561	24.685290	231.50217	853.38693
2003	1110533.8	59868.65	27333.359	999.1567	2266.198	327.1203266	887.9936	3.9304784	42.94211	562.8829	820.29530	351.09470	781.99556	456.45694	809.78884	812.52471	143.932424	46.9069310	23.198447	68.65482	292.20497
2004	1210904.1	35377.29	21775.759	1231.2410	2197.574	1.6569918	940.3415	2.4252789	45.89937	631.3478	726.01924	712.84409	613.95000	109.35109	666.14177	726.30738	122.421875	34.4701432	114.234027	80.83870	707.73870
2005	1221245.8	50495.63	32745.869	785.4644	2334.495	25.0465987	1023.3049	4.5451612	45.61668	776.1369	292.85823	799.07882	1340.40070	486.84558	121.34057	371.43164	34.872906	468.9030909	418.130755	20.17680	987.21289
2006	1796676.4	24902.82	21844.238	770.0808	2752.106	16.4755134	940.7020	4.0692715	31.97384	898.1250	1649.77319	147.90272	939.05174	316.33398	132.15742	1066.67870	152.773926	96.8820184	97.519665	446.14448	195.17531
2007	937683.2	23217.27	34190.962	1828.2235	3036.128	33.1167332	909.0419	3.6472710	30.80906	1355.7825	831.98399	834.18618	1059.17299	590.30950	817.38581	551.56048	275.855851	328.0031075	39.468283	143.15062	361.18765
2008	1322727.9	35919.89	50585.318	4019.9912	3621.820	21.4988583	815.2806	2.4908542	30.61418	1758.6558	1218.21416	1113.44398	82.52178	711.14552	587.98590	902.06783	119.526208	536.9872209	32.922129	123.28445	897.87154
2009	1592331.2	23746.94	21047.440	2771.5505	4164.949	349.2660768	944.8090	2.3000750	28.36084	1717.5063	307.93811	12.28943	1686.33989	4716.24463	1459.96637	73.18907	112.962382	523.2139506	817.209939	463.00129	1058.94939
2010	961203.9	24551.54	40886.914	3808.1544	4903.197	373.8517281	1055.5068	2.5835010	26.22683	1858.0100	1847.98108	117.29693	18.71796	525.34334	2204.25586	208.96092	509.649327	203.1791274	614.275605	122.47269	2300.57394
2011	2354554.6	25980.02	47318.969	4468.4850	5100.427	408.9314408	1264.2293	2.6369156	31.92544	1785.6411	171.46181	1436.57808	1754.69816	2245.99910	2699.05668	1802.89548	146.469506	649.4047550	10.440106	240.63122	312.51836
2012	1420577.2	23215.32	34926.545	4273.1299	5054.489	321.6454135	1216.3220	2.5240832	26.81827	2309.0224	843.63119	1148.02475	1013.99017	972.86279	1557.88100	1106.67162	254.596653	194.5806579	97.283213	348.20727	123.89923
2013	1696028.3	54592.52	37882.257	4673.4426	5745.538	179.3156186	1399.3792	2.9096473	20.96485	1971.5332	1841.55238	1128.30154	2685.99578	699.80047	1014.50464	875.96105	7.011264	0.3767482	347.569892	441.79260	940.67793
2014	972344.6	36395.02	23028.558	1904.7957	4673.277	67.1840888	1413.8913	3.9237297	22.06228	1546.0168	713.60436	66.05407	922.05087	1525.81840	1597.48008	273.26548	10.172301	388.1004522	205.590201	167.82925	123.83650
2015	2218393.6	33046.88	45558.463	1383.2643	4530.773	121.6526944	1296.0487	3.5216599	35.83166	1672.6683	547.02771	189.46726	753.34938	1249.64950	1356.97425	815.07061	217.699751	276.9726074	383.871374	250.05605	382.06222
2016	1215454.5	27596.69	47850.740	1030.2540	4069.254	0.0000000	1260.4855	2.9387336	31.90037	1691.3071	681.20476	201.02191	2205.94057	181.60152	983.37021	159.85957	34.465288	112.2001555	155.982689	70.34119	812.85388
2017	1231725.1	24211.63	55120.223	2143.1775	3934.111	0.0000000	1346.9690	2.0574188	31.06840	994.6226	594.07060	59.18407	351.22831	296.70633	803.19599	288.77401	106.462093	168.2288582	92.902612	269.02273	917.43226
2018	1762591.6	27849.87	9415.999	1475.5712	3104.531	199.7229446	1184.1594	2.8401587	57.63506	929.8991	475.95681	200.24213	150.52191	1558.77750	145.45107	510.91400	646.737449	274.2548063	50.321508	367.12342	2421.60402
2019	NA	62250.37	20019.056	619.8904	3096.781	0.0000000	1079.4926	2.2356150	80.89128	750.3493	995.06556	464.09713	1278.93570	354.82997	695.99419	820.68035	478.900096	595.8727440	595.241102	204.26504	149.59182

Table 4: Constraints defined by user

Id	Constraint	Time-range	Active	Comment
C01	PP_Hzoo + PP_Ozoo + PP_Benthos <= Prod_Sat * 1.5	1988:2018	1	Max primary production
C02	PP_Hzoo + PP_Ozoo + PP_Benthos >= Prod_Sat / 1.5	1988:2018	1	Min primary production
C03	PP_Hzoo + PP_Ozoo + PP_Benthos <= 200000	1988:2019	1	Max primary production when no data is available
C04	PP_Hzoo + PP_Ozoo + PP_Benthos >= 500000	1988:2019	1	Min primary production when no data is available
C05	NorSeaZoo_Hzoo = 3 * 1600	1988:2019	1	Fixed import of herbivorous zooplankton from the Norwegian Sea
C06	NorSeaZoo_Ozoo = 2 * 1600	1988:2019	1	Fixed import of omnivorous zooplankton from the Norwegian Sea
C07	PelF_Fisheries >= Pel_landings	1988:2019	1	Min pelagic catches
C08	PelF_Fisheries <= Pel_landings * 1.1	1988:2019	1	Max pelagic catches
C09	DemF_Fisheries >= Dem_landings	1988:2019	1	Min demersal landings
C10	DemF_Fisheries <= Dem_landings * 1.1	1988:2019	1	Max demersal landings
C11	MM_Fisheries >= Mammals_landings / 1.1	1988:2019	1	Min marine mammals landings
C12a	MM_Fisheries <= Mammals_landings * 1.1 + 1.5	1988:1991	1	Max marine mammals landings prior to 1994 - seal catches assumed to be max 1500t
C12b	MM_Fisheries <= Mammals_landings * 1.1	1992:2019	1	Max marine mammals landings for the period with data for seals and minke whales
C15	Ozoo_Fisheries >= Shrimp_Landings	1988:2019	1	Min omnivorous zooplankton landings
C16	Ozoo_Fisheries <= Shrimp_Landings * 1.1	1988:2019	1	Max omnivorous zooplankton landings
C17	Benthos <= 66 * 1600 * 3	1988:2019	1	Absolute max benthos biomass
C18	Benthos >= 66 * 1600 / 3	1988:2019	1	Absolute min benthos biomass
C19	MM <= 0.34 * 1600 * 3	1988:2019	1	Absolute max marine mammals biomass
C20	MM >= 0.34 * 1600 / 3	1988:2019	1	Absolute min marine mammals biomass
C21	Birds <= 0.007 * 1600 * 3	1988:2019	1	Absolute max birds biomass
C22	Birds >= 0.007 * 1600 / 3	1988:2019	1	Absolute min birds biomass
C23	Hzoo <= Hzoo_Biomass * 2	1988:2019	1	Max herbivorous zooplankton biomass
C24	Hzoo >= Hzoo_Biomass / 2	1988:2019	1	Min herbivorous zooplankton biomass
C25a	Ozoo <= Ozoo_Biomass * 2	c(1988:2001, 2003:2017)	1	Max omnivorous zooplankton biomass (in 2002, 2018, 2019 krill biomass index is missing)
C25b	Ozoo <= Ozoo_Biomass * 4	c(2002, 2018, 2019)	1	Absolute max omnivorous zooplankton biomass
C26	Ozoo >= Ozoo_Biomass / 1.5	1988:2019	1	Min absolute omnivorous zooplankton biomass
C27a	PelF <= Pelagics * 1.4	c(1990:1992,1995:2000,2004:2015,2017:2019)	1	Max pelagics biomass
C27b	PelF <= Pelagics * 2	c(1988:1989,1993,2001:2003,2016)	1	Max pelagics biomass prior to blue whiting data available
C28	PelF >= Pelagics / 1.4	c(1988:1995,1997:2019)	1	Min pelagics biomass (constraint incompatible with inertia in 1996)
C29	DemF <= Demersals * 1.3	1988:2019	1	Max demersals biomass
C30	DemF >= Demersals / 1.2	1988:2019	1	Min demersals biomass
C31	DemF_DemF >= Cod2Cod + Haddock2Cod + Redfish2Cod + Ghalibut2Cod	1988:2019	1	Min consumption of demersals by demersals
C32	PelF_DemF >= Capelin2Cod + Herring2Cod + Polarcod2Cod + BW2Cod	1988:2019	1	Min consumption of pelagics by demersals
C33	Ozoo_DemF >= Shrimp2Cod + Krill2Cod + Amphipods2Cod	1988:2019	1	Min consumption of omnivorous zooplankton by demersals
C34	(Ozoo_DemF + Benthos_DemF + PelF_DemF + DemF_DemF) <= 2 * (Other2Cod + Amphipods2Cod + Krill2Cod + Shrimp2Cod + Capelin2Cod + Herring2Cod + Polarcod2Cod + Cod2Cod + Haddock2Cod + Redfish2Cod + Ghalibut2Cod + BW2Cod)	1988:2019	1	Max consumption of demersals
C35	Birds >= 0.007 * 1600 * 4	1988:2019	0	false constraint to illustrate incompatibility

Ecosystem Assessment for the Barents Sea (ICES (2020)) have been altered to comply with data access restrictions (not all data is publicly available). They are stored in the "Input time-series" sheet in the RCaN-file. Each row indicate a time-step (year) and each column correspond to a particular data-series. The first row is a header with the name of each date-series. The observational data-series information is stored in the dataframe 'series' in the POLYTOPE object:

```
kable(POLYTOPE$series,caption="Observations")%>%
  kable_styling(latex_options = c("striped", "scale_down"), font_size = 8)
```

3.2 Constraints

Explicit constraints that relate food-web components, fluxes and observations can be specificied in the RCaN file. These are written in the form of inequalities or equalities and are provided in the worksheet "Constraints". A constraint can be set to apply to every year or to selected years only. A constraint can be set to active/inactive (column 'active'), which is practical for testing different combination of constraints without the need for writing/deleting the constraints in the worksheet. The constraint information is stored in the dataframe 'constraints' in the POLYTOPE object.

```
kable(POLYTOPE$constraints[,1:5],caption='Constraints defined by user') %>%
  kable_styling(latex_options = c("striped", "scale_down"), font_size = 8)
```

4 The polytope behind a CaNmod object

In section 3, we used the `buildCaN` function to read the RCaN file, but also to construct the corresponding polytope. The polytope is a multidimensional geometrical object that corresponds to the "space of possibles food-web trajectories". The dimension of the polytope is defined by the number of fluxes times the number of time-steps (years) + the number of components. The different elements of the CaNmod object constructed

using the buildCaN function are as follows:

```
names(POLYTOPE)
```

```
##  [1] "components_param" "species"          "fluxes_def"       "flow"
##  [5] "series"           "ntstep"           "aliases"          "dynamics"
##  [9] "data_series_name" "constraints"      "H"                "N"
## [13] "Nend"             "A"                "AAll"             "C"
## [17] "CAll"             "v"                "vAll"             "L"
## [21] "b"                "bAll"             "symbolic_enviro"
```

Matrices C and A, and vectors b and v correspond to the matrices of active constraints that define the polytope A X <= b and C X = v (AAll, CAll, bAll and vAll are similar but include all constraints, active or not). Matrices N, H, F and L are matrices that are used to describe the dynamics of the system, more specifically, they describe how biomasses at any time can be derived from flows, using the relationship provided in the supplementary material of Planque and Mullon Planque and Mullon (2020).

4.1 Structure of the polytope

The polytope is defined inside a space with a high number of dimensions.

For the Barents sea example, we have:

```
paste("number of inequalities:", nrow(POLYTOPE$A))
```

```
## [1] "number of inequalities: 2593"
```

```
paste("number of equalities:", nrow(POLYTOPE$C))
```

```
## [1] "number of equalities: 64"
```

```
paste("number of parameters (dimensions):", ncol(POLYTOPE$C))
```

```
## [1] "number of parameters (dimensions): 775"
```

The inequalities and equalities result from both implicit and explicit constraints, each line corresponds to a constraint and a specific year. The number of parameters is the dimension of the polytope.

4.2 Suitability of the polytope

The polytope contains the set of possible food-web trajectories. If some constraints are incompatible, the polytope will be empty, signifying that it is not possible to find any food-web trajectories that can simultaneously satisfy all constraints. When there are too few constraints, the polytope can be unbounded. In such case, some of the fluxes can take an unbounded range of values, a somehow unrealistic situation. Checking the polytope is an important step. Users can adopt different strategy: some users try first to specificy as many constraints as possible with the risk of constructing an empty polytope. They then need to relax some of the model constraints in order to find possible solutions. Others tend to start with a reduced set of constraints with the risk of having an unbounded polytope. They then need to add extra constraints to derive possible and plausible food-web trajectories. Both approaches have their pros and cons. Here, we focus on how to identify potential issues with the polytope and the possible ways to resolve them so that the polytope is suitable for sampling. As a recommendation: it is often a good strategy to start working with a limited number of years to limit the dimension of the polytope and decrease computation time. Moreover, it is often convenient to add a temporary constraint that specifies an upper limits to the sum of all flows; it ensures to have a bounded polytope and to carry out some preliminary explorations.

4.2.1 First step: checking the polytope status

The R command *checkPolytopeStatus* returns the current status of the polytope. In the Barents Sea example, the polytope is 'OK' which means that it is non-empty and bounded. In other words, there is a limited set of

possible trajectories.

```
checkPolytopeStatus(POLYTOPE)
```

```
## [1] "polytope ok"
```

In some situations, the function may return 'empty polytope' or 'unbounded polytope'. It might then be necessary to add/activate new constraints or remove/disactivate existing constraints.)

4.2.2 Toggling constraints

Constraints can be activated/deactivate in the RCaN-file, using the 'active' switch in the 'Constraints' worksheet. In addition, it is possible to activate/deactivate constraints directly within R, once the RCaN file has been loaded. We illustrate this below, using the command *toggleConstraint*:

```
POLYTOPE <- quiet(toggleConstraint(POLYTOPE,
                    c("C01","C02","C03","C04","C05","C06",
                      "C07","C08","C09","C10","C11","C12",
                      "C15","C16","C19","C21")))
```

In this example, we removed constraints C01 to C12 and C15, C16, C19 and C21, which bound the input (primary production), the output of the system (fisheries) and the maximum biomass of top predators (birds and marine mammals).

```
checkPolytopeStatus(POLYTOPE)
```

```
## [1] "polytope not bounded"
```

As a result, the polytope is now unbounded. We therefore reactivate the constraint with the 'toggleConstraint' command.

```
POLYTOPE <- quiet(toggleConstraint(POLYTOPE,
                        c("C01","C02","C03","C04","C05","C06",
                          "C07","C08","C09","C10","C11","C12",
                          "C15","C16","C19","C21")))
```

4.2.3 Finding incompatible constraints

We artificially add a new constraint which is incompatible with the existing ones to illustrate a situation where the polytope is empty.

```
POLYTOPE<-quiet(toggleConstraint(POLYTOPE, "C35"))
```

From section 3.2, we can see that constraint C35 (the biomass of birds should exceed 33.6) is not compatible with constraints C21 (the biomass of birds should NOT exceed 44.8), i.e. the two can not be fulfilled simultaneously. As a consequence, checking the polytope status returns 'empty polytope':

```
checkPolytopeStatus(POLYTOPE)
```

```
## [1] "empty polytope"
```

The RCaNmodel function 'findingIncompatibleConstr' is then useful to identify which constraints are incompatible:

```
options(width = 60)
incomp <- findingIncompatibleConstr(POLYTOPE)
```

```
...
## [1] "###polytope is ok when following constraints are relaxed:"
## [1] " C35 : 1988,  C35 : 1989,  C35 : 1990,  C35 : 1991,  C35 : 1992,  C35 : 1993,  C35 : 1994,  C35
## [1] "####Those constraints seem incompatible with:"
```

```
## [[1]]
## [1] " C35 : 1988:  C21 : 1988,  C35 : 1988,  C35 : 1989,  C35 : 1990,  C35 : 1991,  C35 : 1992,  C35
##
## [[2]]
## [1] " C35 : 1989:  C21 : 1989,  C35 : 1988,  C35 : 1989,  C35 : 1990,  C35 : 1991,  C35 : 1992,  C35
##
## [[3]]
...
```

The function returns that constraint C35 is problematic (see ###polytope is ok when following constraints are relaxed) and that it is incompatible with constraints C21 ("####Those constraints seem incompatible with:").

In such situation, it is necessary to relax some of the above constraints (C35, C21) to obtain a non-empty polytope. The choice of the constraint to relax is left to the modeller, based on domain knowledge.

In the following, we revert to a model version in which constraint C35 is inactive and the polytope is OK:

```
quiet(POLYTOPE<-toggleConstraint(POLYTOPE, "C35"))
```

4.3 Bounds of the polytope

To obtain a first overview of the polytope, it is useful to compute its limits in all dimensions. This can be achieved with the RCaNmodel function *getAllBoundsParam*. The output is a dataframe with a row per parameter/dimension. It is also possible is to compute the bounds for a small set of dimensions only. This can be done using the function *getBoundParam*. This is illustrated below for the two dimensions corresponding to the flux from Omnivorous Zooplankton to Birds in 2017 and 2018:

```
getBoundParam(POLYTOPE, which(colnames(POLYTOPE$A)=="Ozoo_Birds[2017]"))
```

```
##      min      max
##    0.000 2666.667
```

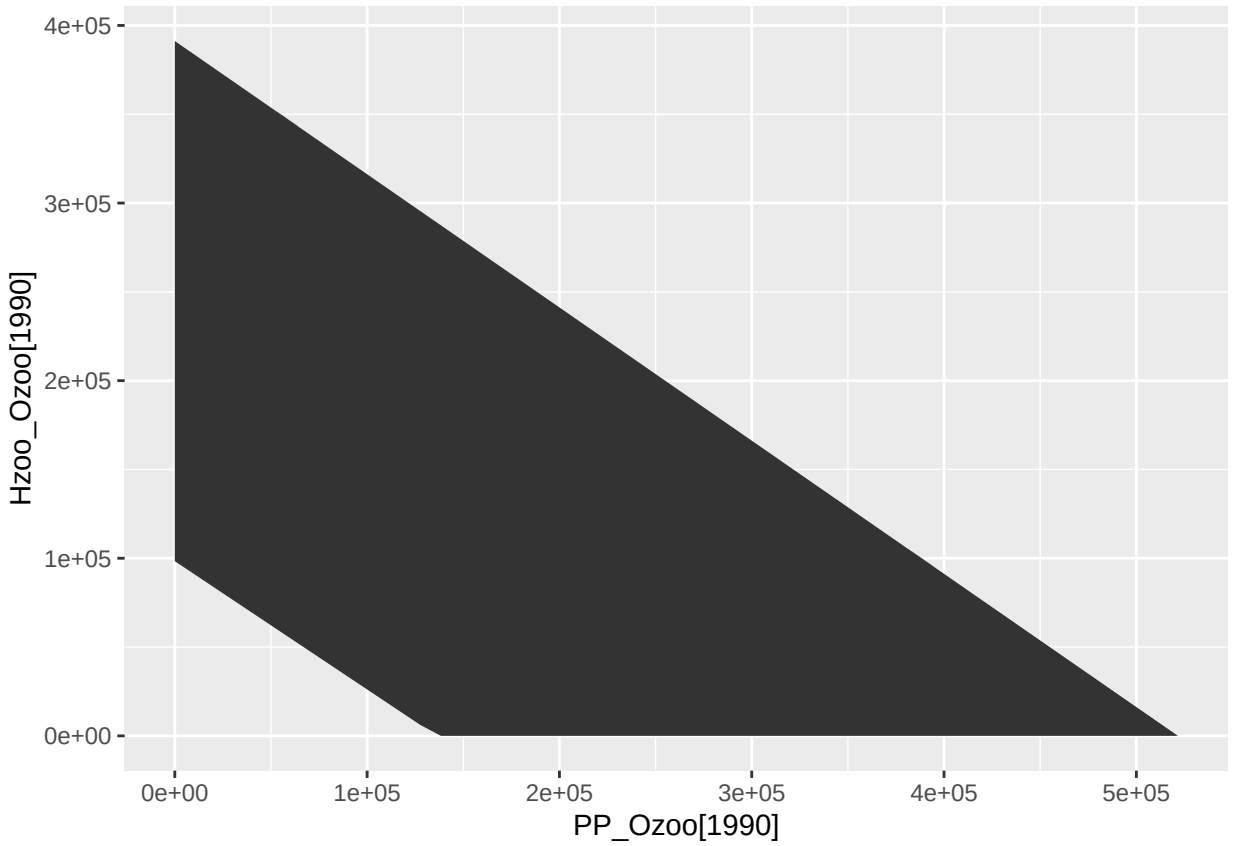
```
getBoundParam(POLYTOPE, which(colnames(POLYTOPE$A)=="Ozoo_Birds[2018]"))
```

```
##      min      max
##    0.000 2666.667
```

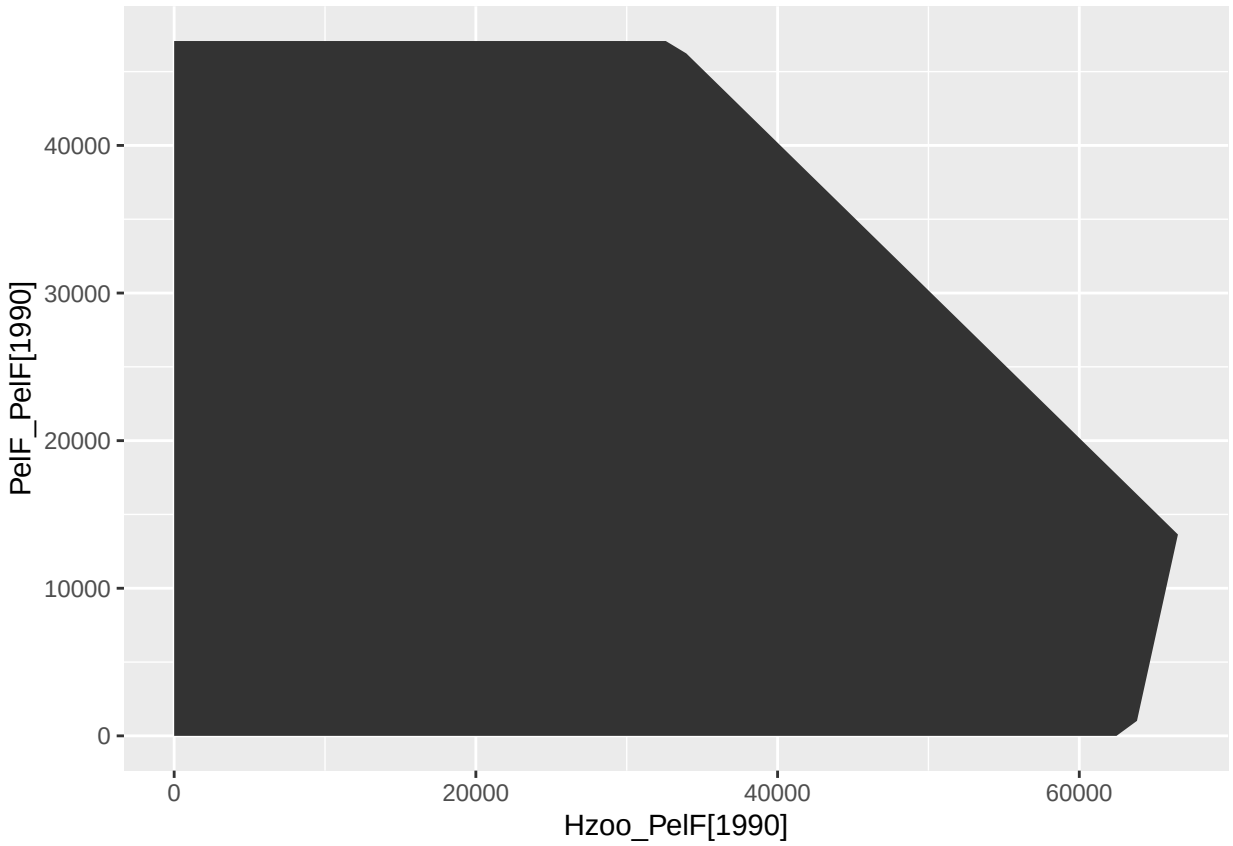
4.4 2D-slices of the polytope

2-dimensional pictures of the polytope are useful to visualise how the bounds of different fluxes are jointly constrained. These are obtained using the RCaNmodel function *plotPolytope2D*:

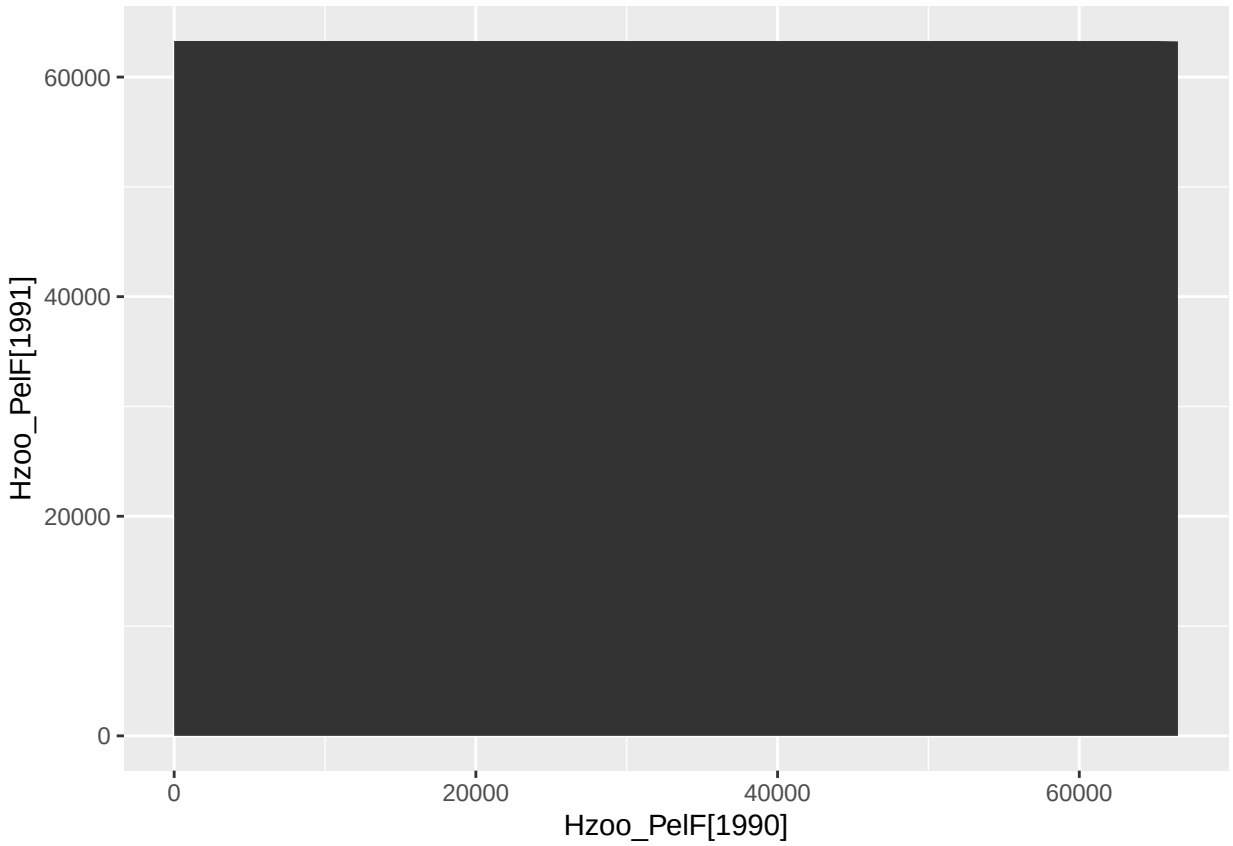
```
start_time <- Sys.time()
plotPolytope2D(POLYTOPE,
               c('PP_Ozoo[1990]', 'Hzoo_Ozoo[1990]'),
               progressBar=FALSE)
```



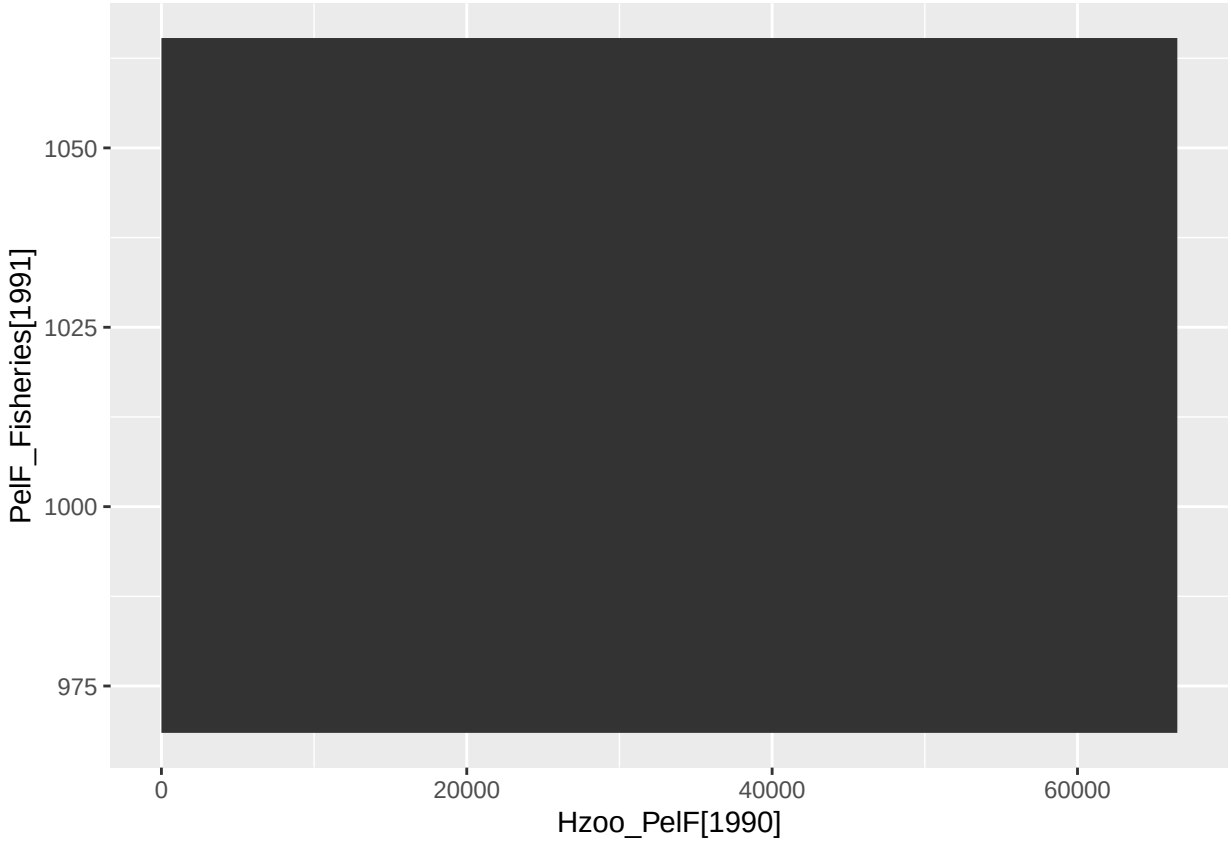

```
plotPolytope2D(POLYTOPE,
               c('Hzoo_PelF[1990]', 'PelF_PelF[1990]'),
               progressBar=FALSE)
```




```
plotPolytope2D(POLYTOPE,
               c('Hzoo_PelF[1990]', 'Hzoo_PelF[1991]'),
               progressBar=FALSE)
```




```
plotPolytope2D(POLYTOPE,
               c('Hzoo_PelF[1990]', 'PelF_Fisheries[1991]'),
               progressBar=FALSE)
```




```
end_time <- Sys.time()
end_time - start_time
```

```
## Time difference of 3.809366 mins
```

5 Sampling the polytope

5.1 Sampling

The RCaNmodel function *sampleCaN* is used to sample of the polytope, i.e. to sample several random possible trajectories of the food-web dynamics. The output of this command is a mcmc.list object (defined in package coda). The function parameters include: * N: the number of samples * thin: the thinning, i.e. the number of samples that are left out from the output * nchains: the number of sampling chains * ncore: the number of CPU cores used * method: one of gibbs (default) or hitandrun

With N=100, thin=10, and nchain=2 the *sampleCaN* function will sample two chains of 1,000 values and return one output every 10, that is a total of 200 valid samples

```
if (file.exists("sampleCaN.rdata")){
  load("sampleCaN.rdata")
} else {
system.time(SAMPLE <- sampleCaN(POLYTOPE, N=100, thin=10, nchain=2, ncore=2))
  save(SAMPLE, file = "sampleCaN.rdata")
}
```

5.2 Convergence of the sample chains

Check the mixing of the Monte Carlo Marko chain algorithm that is used to build the sample can be achieved with commands from the coda library.

```
summ <- summary(SAMPLE$mcmc)
n = 31
kable( head(cbind.data.frame(summ$statistics, summ$quantiles) %>%
              select('Mean', 'SD', '2.5%', '50%', '97.5%'),
         n=n),
  caption=paste("summary of", n, "parameter distributions"),
          digits=0) %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

5.2.1 Gelman diagnostic

Gelman and Rubin (1992) proposed a general approach to asses the convergence of MCMC outputs in which $m > 1$ parallel chains are run with starting values that are overdispersed relative to the posterior distribution. Convergence is diagnosed when the chains have 'forgotten' their initial values, and the output from all chains is indistinguishable. The gelman.diag diagnostic is applied to a single variable from the chain. It is based a comparison of within-chain and between-chain variances, and is similar to a classical analysis of variance.

```
gelman.diag(SAMPLE$mcmc[,c('PP_Benthos[1990]',
                           'Ozoo_Birds[1990]',
                           'DemF_Fisheries[1990]',
                           'PelF_DemF[1990]')],
            multivariate = FALSE)
```

```
## Potential scale reduction factors:
##
##                       Point est. Upper C.I.
## PP_Benthos[1990]            1.03       1.06
## Ozoo_Birds[1990]            1.04       1.21
## DemF_Fisheries[1990]        1.04       1.20
## PelF_DemF[1990]             1.06       1.26
```

Theoretically a statistics below 1.05 suggests an appropriate convergence.

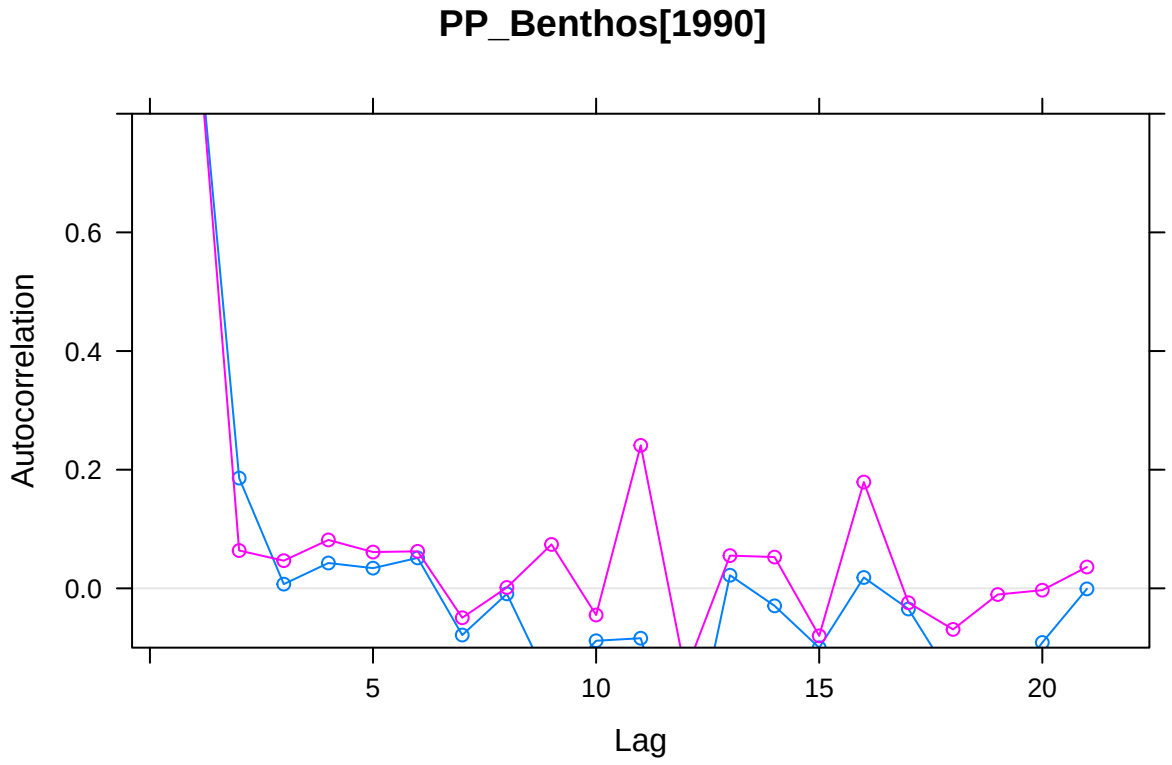
5.2.2 Autocorrelation function

The *acfplot* function from the coda library shows the autocorrelation function of sampling chains. It is illustrative of the independence of successive samples. If the samples are not serially correlated (which is desired) the autocorrelation should remain around zero (except for the first value which is always 1). Otherwise, it might be worthwhile to increase the thinning during RCaNmodel sampling and/or to discard the first samples.

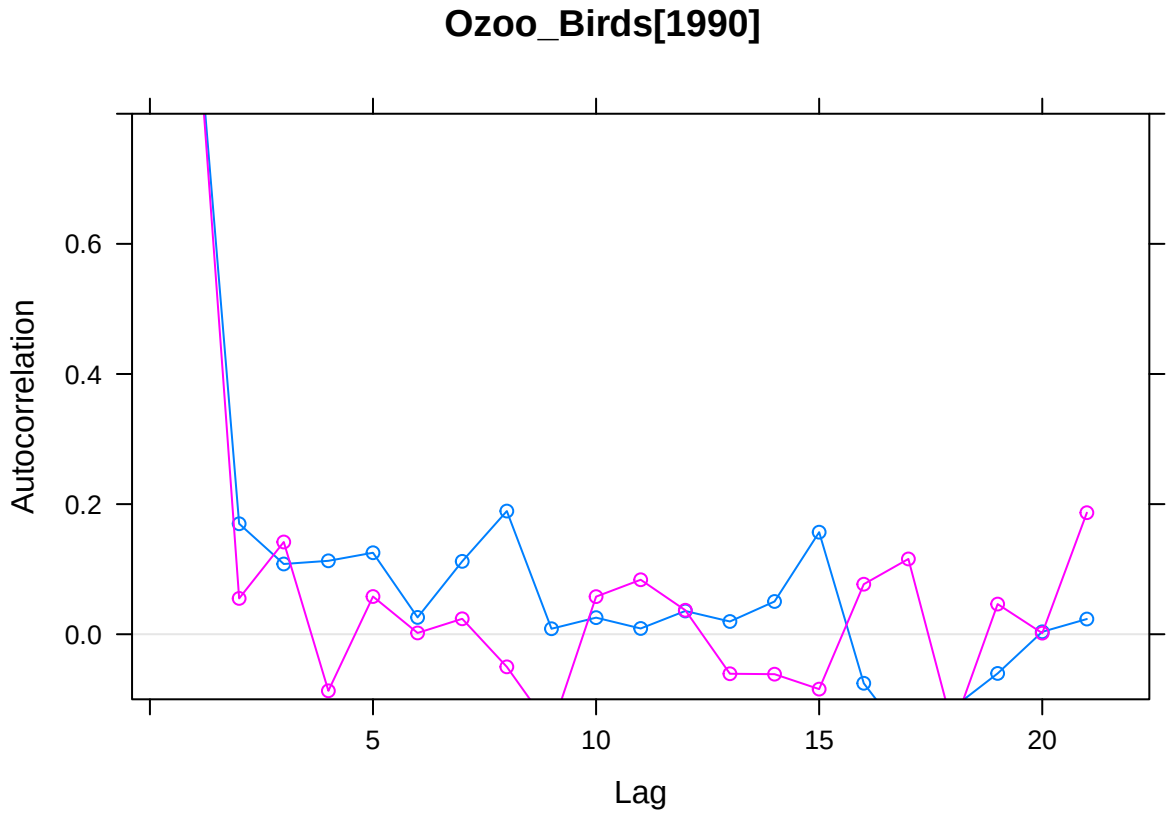
```
acfplot(SAMPLE$mcmc[,'PP_Benthos[1990]'],
        ylim=c(-0.1,0.8),
        main='PP_Benthos[1990]')
```

Table 5: summary of 31 parameter distributions

	Mean	SD	2.5%	50%	97.5%
PP_Ozoo[1988]	214808	133334	17269	191936	494022
PP_Hzoo[1988]	865830	230974	470279	850205	1344889
PP_Benthos[1988]	659391	227324	282740	649583	1096390
Hzoo_Ozoo[1988]	150207	91959	7901	137421	333707
Hzoo_PelF[1988]	7929	4651	420	7910	16060
Ozoo_Ozoo[1988]	1147095	683535	99058	1056117	2622754
Ozoo_PelF[1988]	8130	4656	600	7775	17055
Ozoo_DemF[1988]	2717	662	1791	2605	4220
Ozoo_MM[1988]	6043	1702	2463	6471	8669
Ozoo_Birds[1988]	1057	662	44	1029	2229
Benthos_Benthos[1988]	368841	277017	13122	300944	950440
Benthos_DemF[1988]	882	654	50	732	2388
PelF_PelF[1988]	2071	1604	54	1698	6231
PelF_DemF[1988]	1516	527	740	1454	2646
PelF_MM[1988]	1976	1571	57	1551	5730
PelF_Birds[1988]	883	605	26	773	2013
DemF_DemF[1988]	599	320	223	516	1297
DemF_MM[1988]	499	336	20	442	1276
NorSeaZoo_Hzoo[1988]	12800	0	12800	12800	12800
NorSeaZoo_Ozoo[1988]	3200	0	3200	3200	3200
PelF_Fisheries[1988]	0	0	0	0	0
DemF_Fisheries[1988]	888	25	847	891	928
MM_Fisheries[1988]	1	0	0	1	2
Ozoo_Fisheries[1988]	55	2	52	55	57
PP_Ozoo[1989]	82979	52060	7915	79744	186820
PP_Hzoo[1989]	459605	141233	209739	444772	713586
PP_Benthos[1989]	926577	308764	411940	903412	1511172
Hzoo_Ozoo[1989]	80289	38522	10231	81098	147290
Hzoo_PelF[1989]	14209	7982	1572	15063	28555
Ozoo_Ozoo[1989]	510648	299139	37509	486652	1087051
Ozoo_PelF[1989]	12748	8220	199	11067	29727




```
acfplot(SAMPLE$mcmc['Ozoo_Birds[1990]'],
        ylim=c(-0.1,0.8),
        main='Ozoo_Birds[1990]')
```

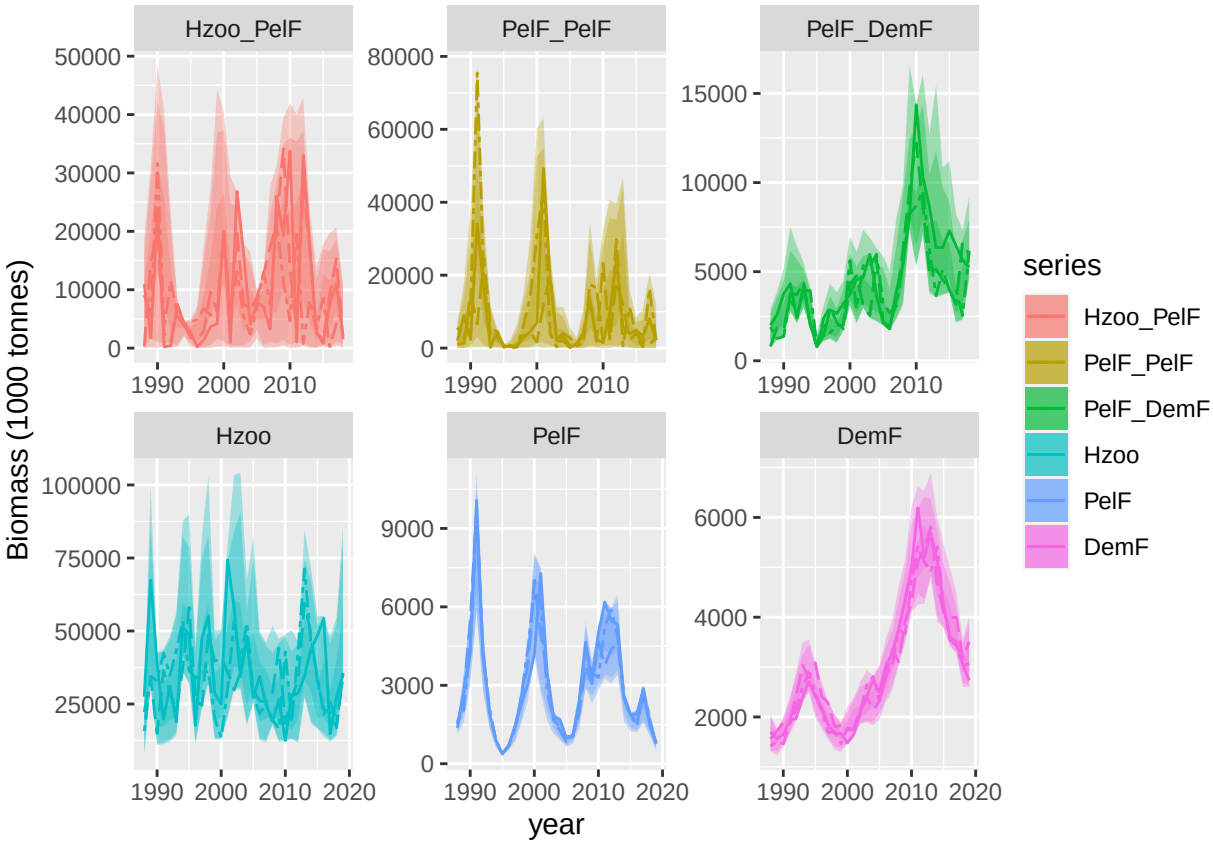


6 Visualisation of the model results

The graphical representation of the possible dynamics of the food-web is a important output of the model. The RCaNmodel function *ggSeries* produces standardised plots of the time-series of components and/or fluxes. By default, for each component, 3 randomly picked samples/trajectories are plotted (plain, dashed and dotted lines) together with the envelopes containing 100%, 95% and 50% of the sampled trajectories.

For components, we get:

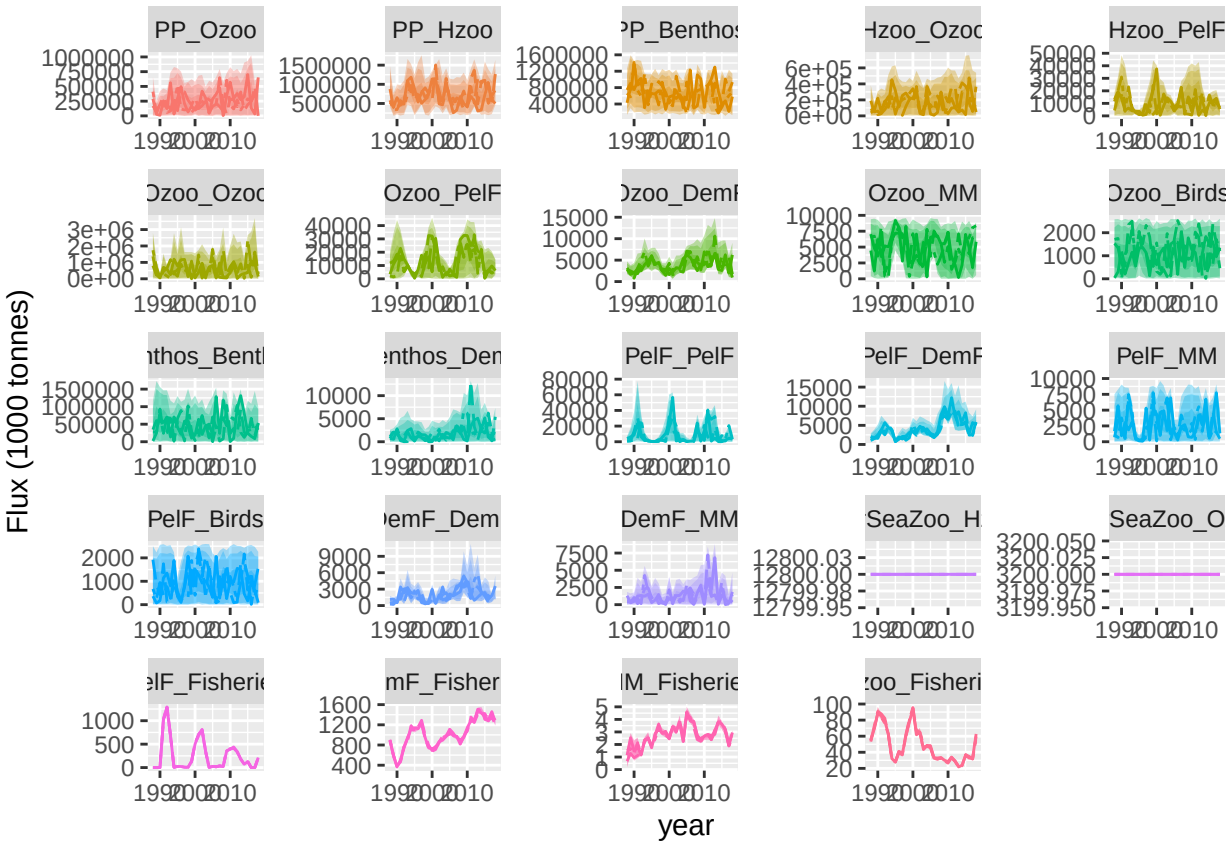
```
ggSeries(SAMPLE,c('Hzoo_PelF','PelF_PelF','PelF_DemF','Hzoo','PelF','DemF'))+
  ylab("Biomass (1000 tonnes)")
```



In this example we observe different situations for the different species: for pelagic fish (PelF) and demersal fish (DemF) the constraints imposed in relation to the observational data-series constrain the trajectories of biomass which are all situated within a narrow band. On the other hand, there is little information provided to constrain directly the trajectories of marine mammal and Birds which leads to large variability between the different samples.

The dynamics of fluxes can be plotted in a similar fashion:

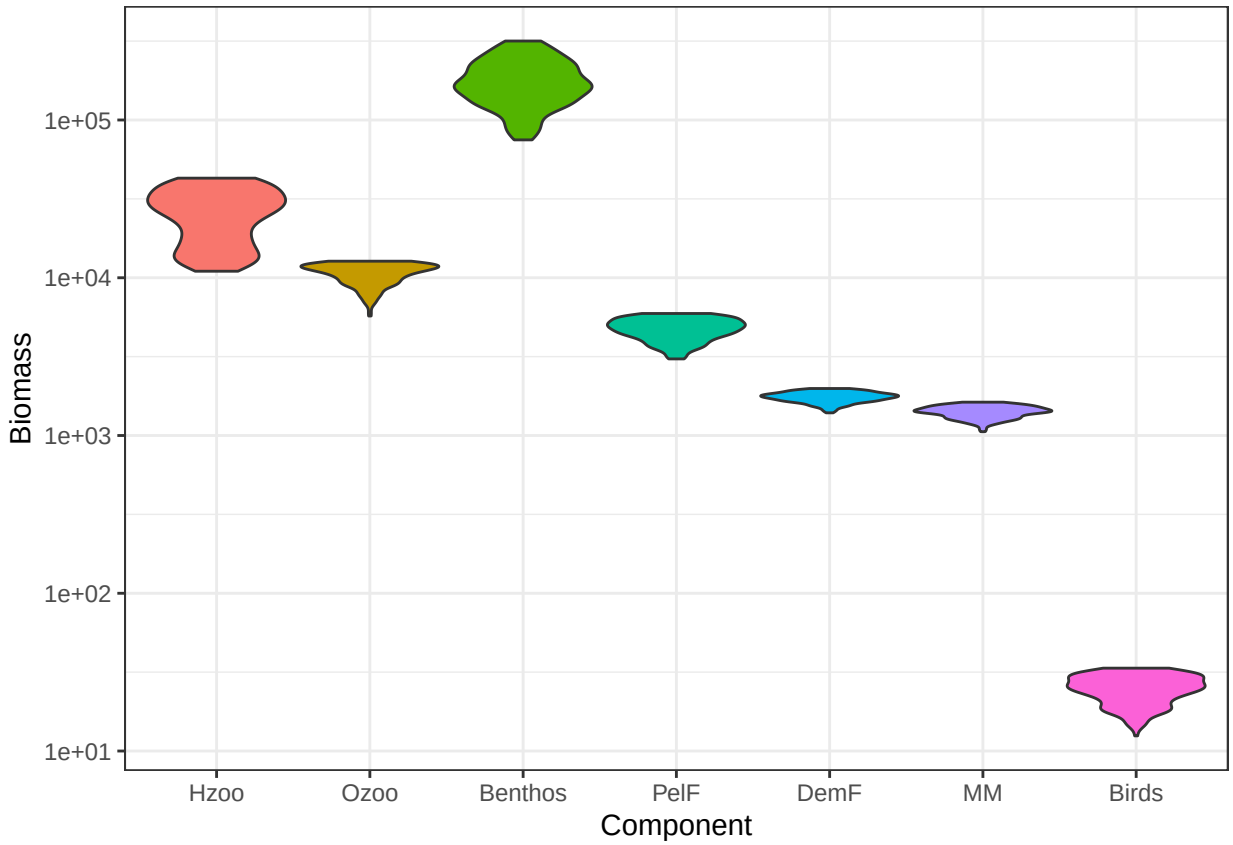
```
ggSeries(SAMPLE, POLYTOPE$flow, TRUE) +
  theme(legend.position = "none") +
  ylab("Flux (1000 tonnes)")
```



6.1 Empirical distributions of biomass and fluxes

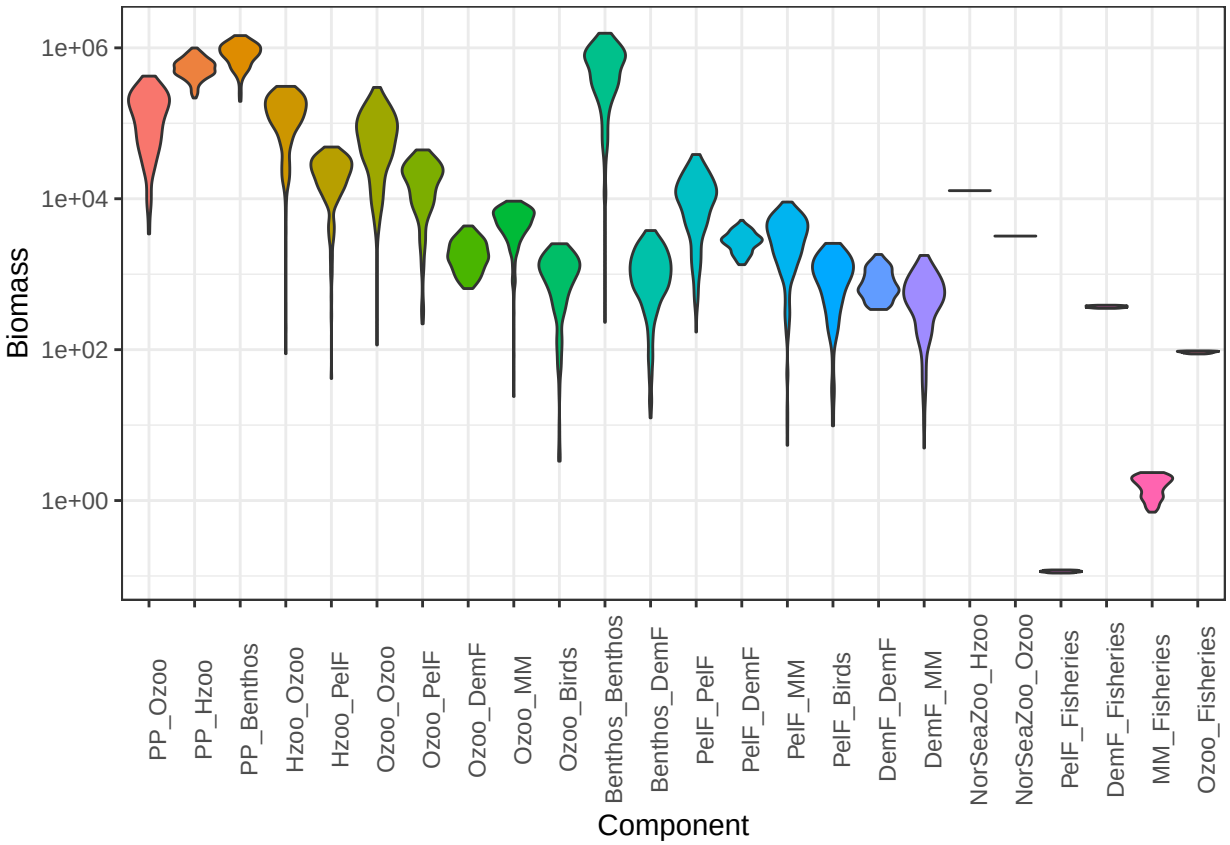
Exploring the distribution of biomass and fluxes across samples and across years is also a useful output of the model. The RCaNmodel function *ggViolin* produces standardised violin plots of these distributions. By default, these are presented on a log scale which allows for comparison between biomass or fluxes that can span several orders of magnitude. For example, the distribution of species biomass for the year 1990 is obtained as follows:

```
ggViolin(SAMPLE,POLYTOPE$species,year=1990,TRUE) +
  xlab('Component') +
  ylab('Biomass')
```



The violin plots of all fluxes in 1990 are produced as follows::

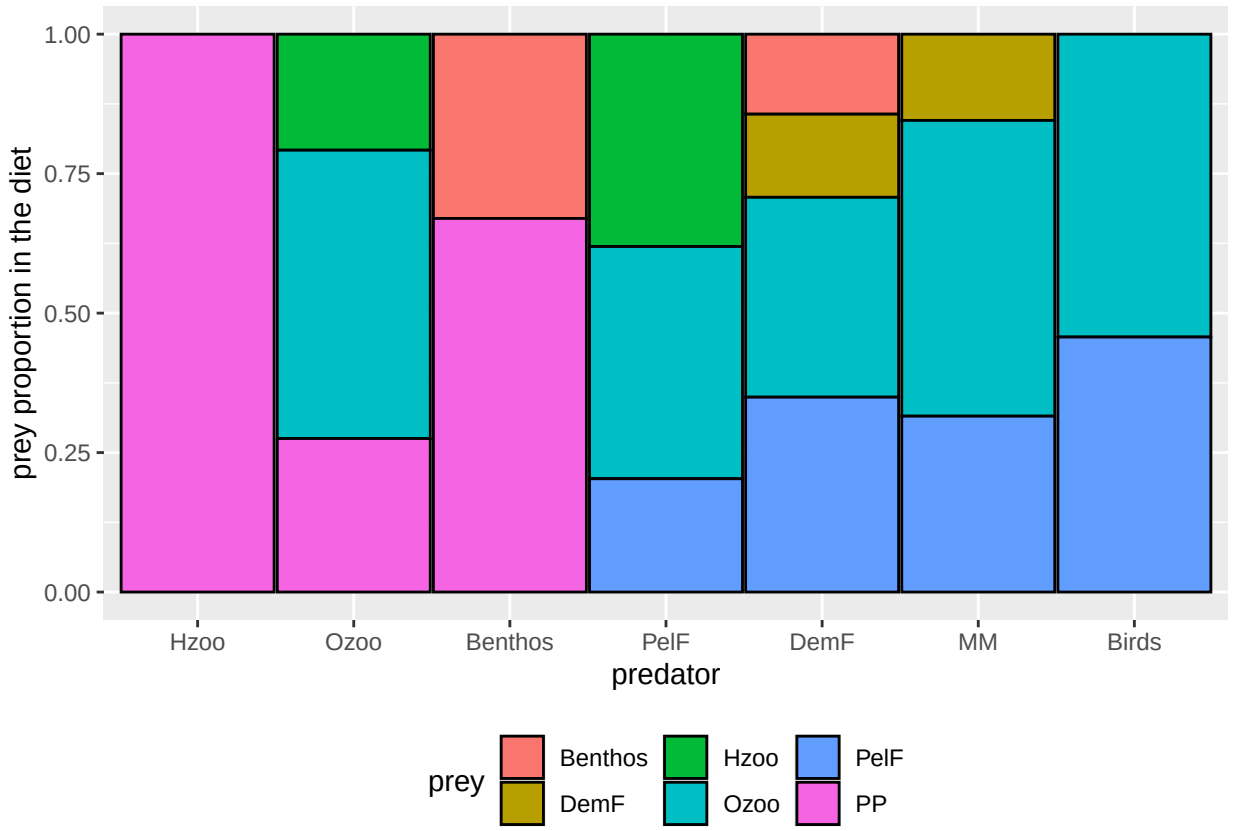
```
ggViolin(SAMPLE,POLYTOPE$flow,year=1990,TRUE) +
  xlab('Component') +
  ylab('Biomass') +
  theme(axis.text.x=element_text(angle=90))
```



6.2 Diet composition

The diet matrix is a key parameter of many trophic models, such as e.g. Ecopath (Christensen and Walters (2004)). In CaN models, diet is non deterministic, in the sense that trophic fluxes are constrained but not determined by biomass. Diets are thus emerging from the model and it is interesting to explore the diets of individual species. A convenient way is to use the RCaNmodel function *ggDiet* to draw the average diet for each predators:

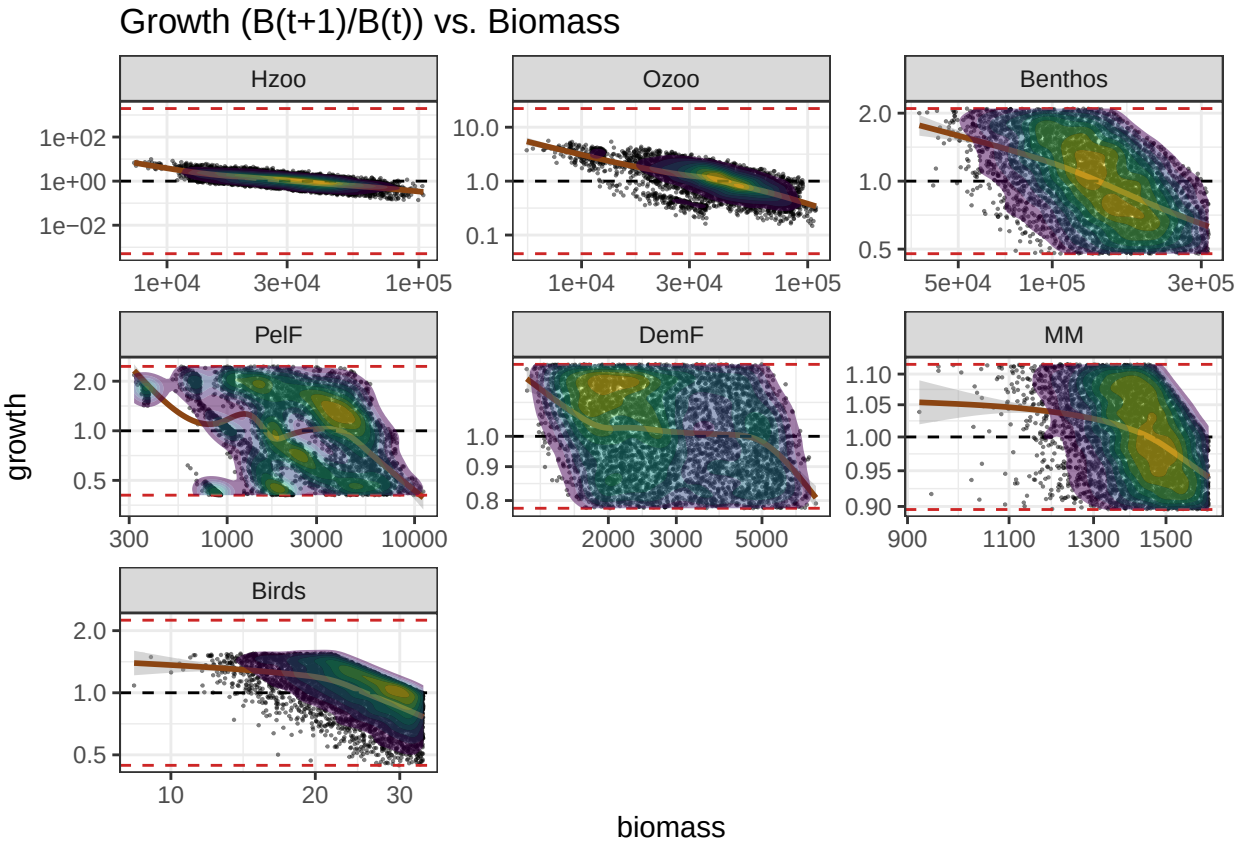
```
ggDiet(SAMPLE, POLYTOPE$species) + ylab('prey proportion in the diet')
```



6.3 Growth and density-dependence

Density-dependence usually reflects how growth is related to population size In RCaNmodel trajectories this can be expressed as growth in biomass in relation to biomass. The RCaNmodel function *ggGrowth* plots the growth rate (measured as the ratio of biomass at time $t+1$ over the biomass at time t) as a function of the biomass (at time t). Plotting density-dependence for all species is done as follows:

```
ggGrowth(SAMPLE, POLYTOPE$species)
```

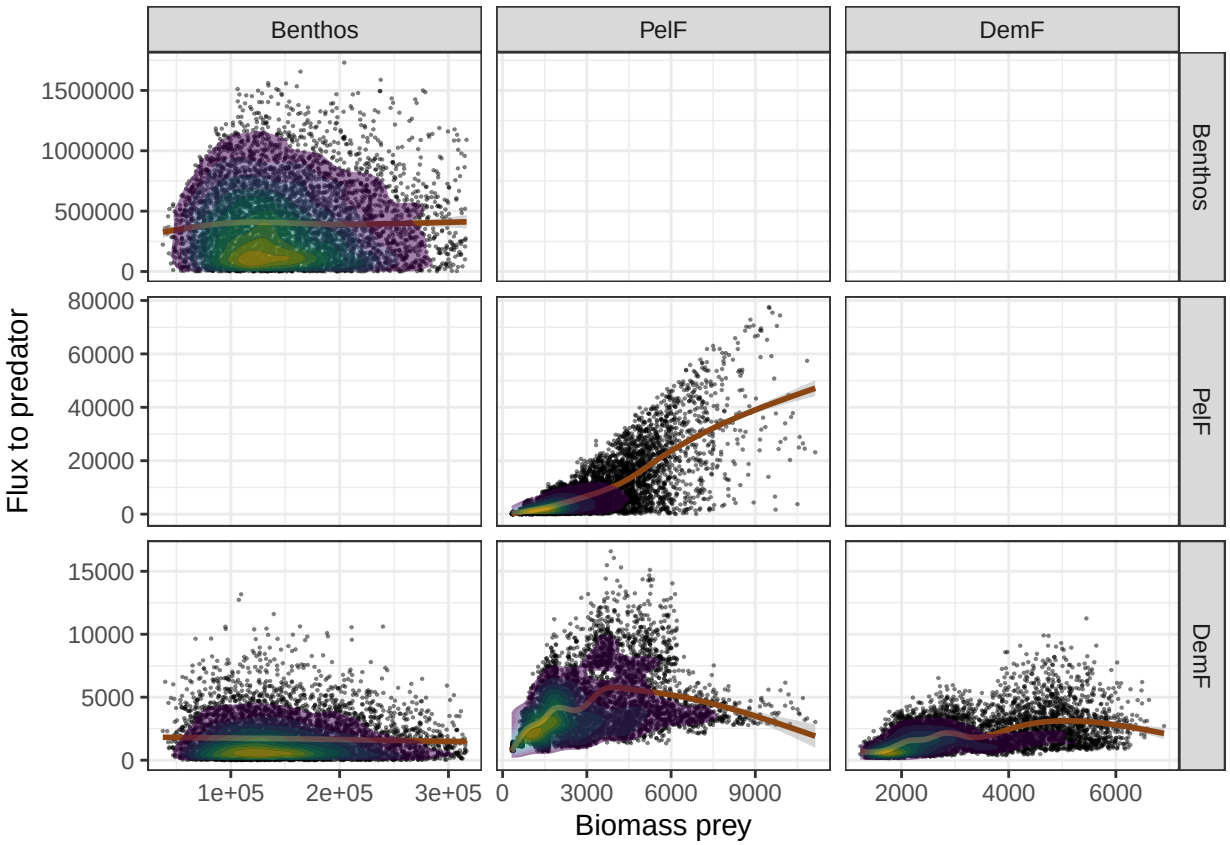


This illustrates how density-dependence of the food-web components can emerge from the food-web dynamics. The upper and lower red-dotted lines visualise the minimum and maximum growth rates set by the inertia constraints for each component. The thick lines are smoothers which highlight the possible underlying relationship between growth and biomass.

6.4 Trophic functional relationships

The RCaNmodel function *ggTrophicRelation* plots the empirical trophic functional relationships (TFRs) between the biomass of a prey and the quantity of that prey consumed by a predator. Below is the command for plotting the TFR between benthos, pelagic fishes and demersal fishes in the Barents Sea example.

```
ggTrophicRelation(SAMPLE, POLYTOPE$species[3:5])
```

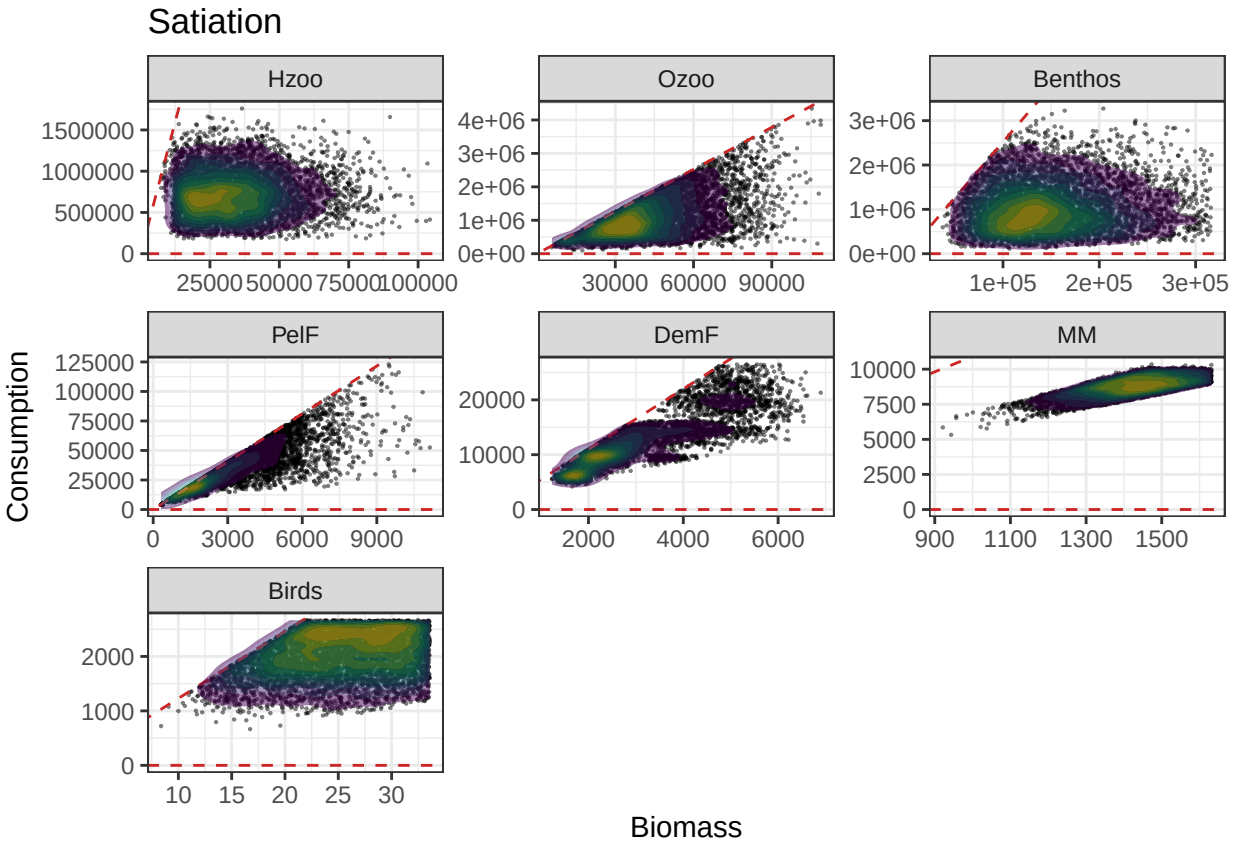


Above, we see that the predation of benthos by demersal fishes does not increase with the biomass of benthos, suggesting that benthos is not a limiting factor in the diet of demersal fishes.

6.5 Satiation

The function *ggSatiation* plots the total biomass of prey eaten by a predator, as a function of the biomass of the predator. In addition the plot includes an indication (upper red dotted line) of the maximum possible consumption rate, derived from the satiation parameter of the predator species. An example is illustrated below for all model components and over the entire sampling period. It shows that marine mammals never reach their satiation. On the other hand, demersal fish are often closed to satiation, suggesting that they are not food limited.

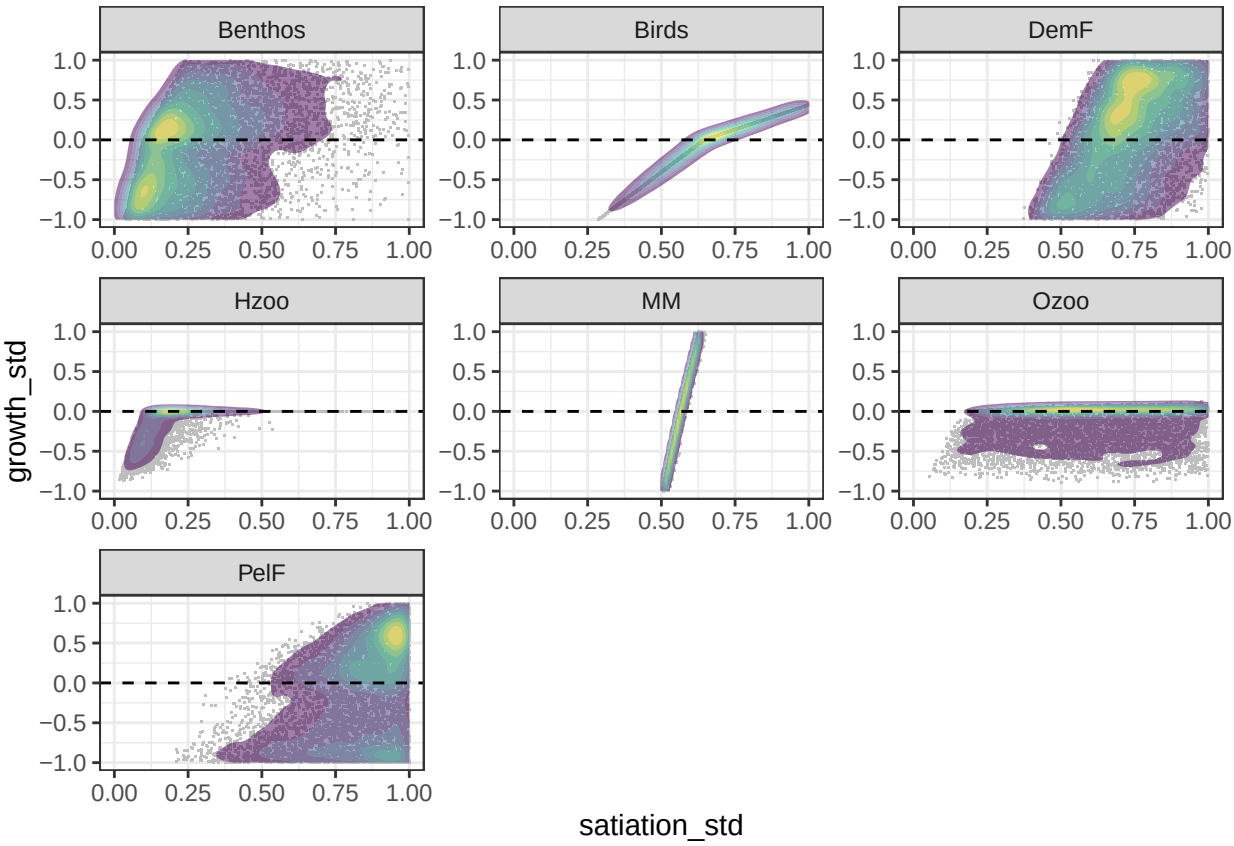
```
ggSatiation(SAMPLE, POLYTOPE$species)
```



6.6 Feeding and Growth

The RCaNmodel function *ggSatiatInertia* is used to explore the relationships between population growth and feeding. The plot shows standardized total consumption(0=no feeding, 1=feeding to satiation) as a function of standardized population growth/mortality (-1 = maximum mortality, 0 = no change in biomass, 1 = maximum growth). A positive relationship between growth and satiation is indicative of bottom up control.

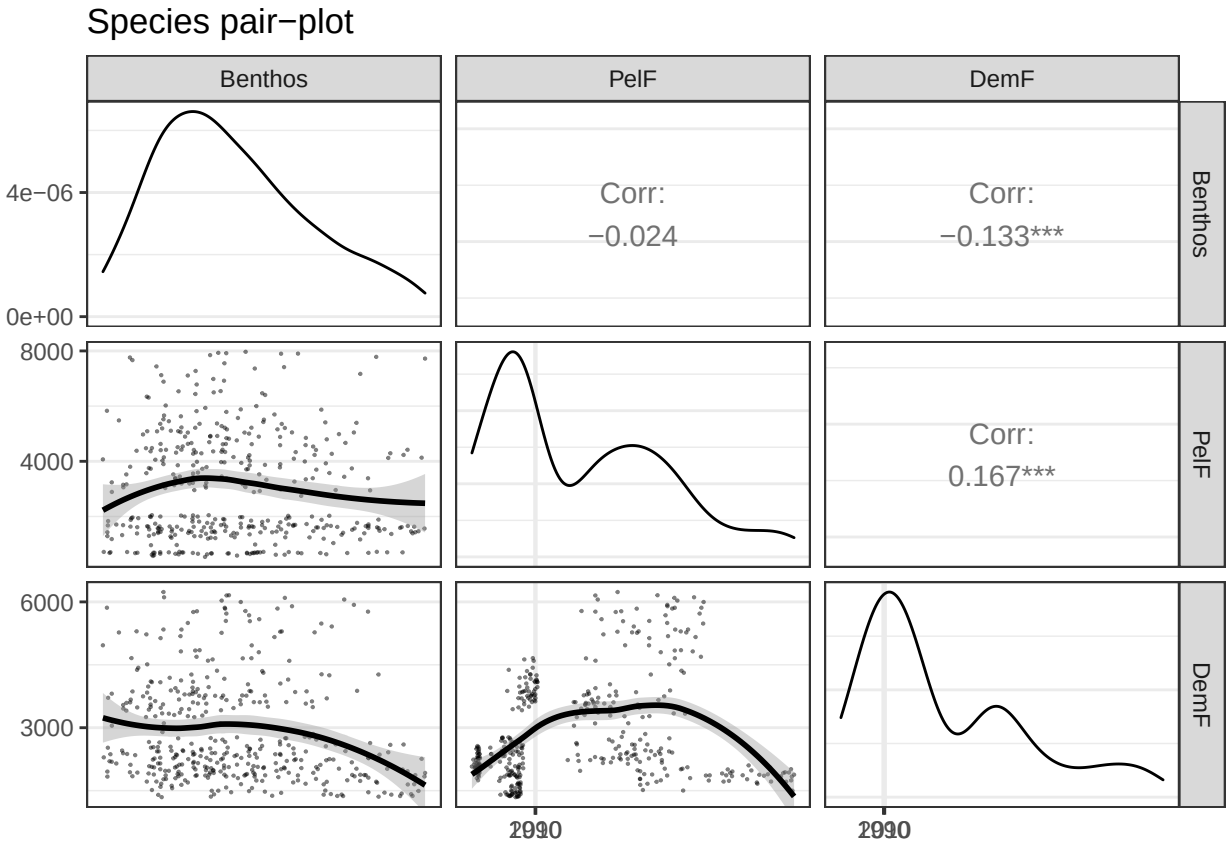
```
ggSatiatInertia(SAMPLE, POLYTOPE$species)
```



6.7 Pair-plots

The function *ggPairsBiomass* is used to explore the relationships between the biomass of different food-web components/species. The plot shows the individual component density function in the diagonal, the scatterplots of one species against the other in the lower triangle and the Kendall correlation coefficient in the upper triangle. An example is illustrated below for three model components and over the entire sampling period. The function parameter *frac* indicates the fraction of the samples which will be plotted. This is useful for large sample size which can lead to overloaded figures.

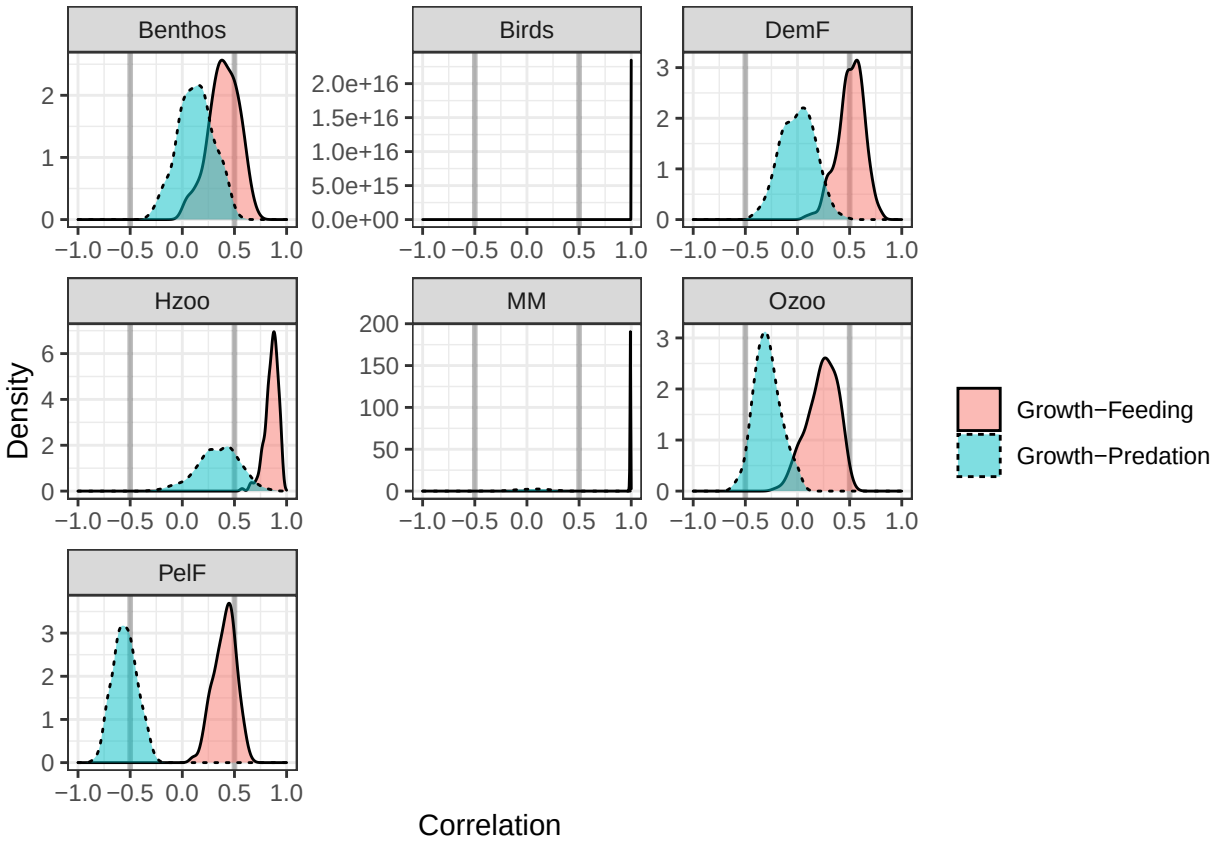
```
ggPairsBiomass(SAMPLE,c("Benthos","PelF","DemF"),logscale=FALSE, frac = .25) + #we only 25% of the poin
  scale_y_continuous(n.breaks=3)+
  scale_x_continuous(breaks=c(1990,2010))
```



6.8 Top-Down and Bottom-Up controls

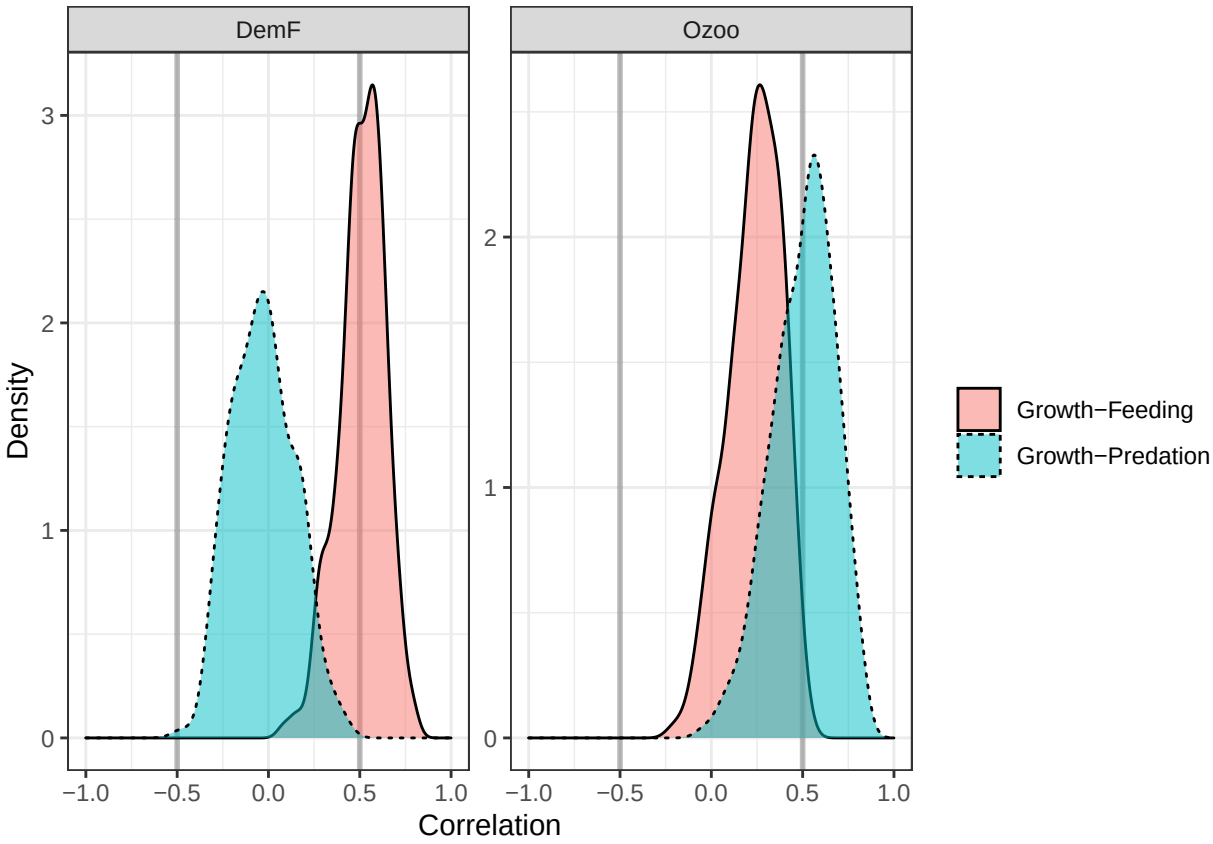
One important aspect of food-web dynamics is the presence of possible trophic controls (Bottom-up or Top-Down). We define here bottom-up as a situation in which the growth of a species/component is positively correlated to the consumption by this species. Similarly, Top-down control is defined as a situation in which the growth of a species/component is negatively correlated to the predation/fishing on this species. The RCaNmodel function *ggTopDownBottomUp* plots the density function of the correlation coefficients between growth and consumption/predation:

```
ggTopDownBottomUp(SAMPLE)
```



This function display the correlation between the growth of the species with both (1) its total ingestion of prey per unit of biomass (in red) and (2) the how much it is predated per unit of biomass (in green). A red density plot tending to 1 indicates that growth is correlated to feeding, and therefore a bottom up control. Conversely, a green density plot tending towards minus one indicates that growth is negatively correlated to the predation and suggests a top down control. It is possible to select which fluxes are accounted for in this diagram (for example to exclude non-trophic fluxes):

```
# we restrict the analysis to Ozoo and DemF and limit to the predation by
#respectively DemF/PelF and MM
ggTopDownBottomUp(SAMPLE,
                  species=list(Ozoo=c("DemF","PelF"),
                               DemF=c("MM")))
```



Christensen, Villy, and Carl J Walters. 2004. "Ecopath with Ecosim: Methods, Capabilities and Limitations." *Ecological Modelling*, Placing Fisheries in their Ecosystem Context, 172 (2): 109–39. https://doi.org/10.1016/j.ecolmodel.2003.09.003.

ICES. 2020. "Working Group on the Integrated Assessments of the Barents Sea (WGIBAR)." Journal Article. *ICES Scientific Reports* 2 (30): 212pp. https://doi.org/10.17895/ices.pub.5998.

Lindstrøm, U., B. Planque, and Sam Subbey. 2017. "Multiple Patterns of Food Web Dynamics Revealed by a Minimal Non Deterministic Model." Journal Article. *Ecosystems* 20: 163–82. https://doi.org/10.1007/s10021-016-0022-y.

Planque, B, and C. Mullon. 2020. "Modelling Chance and Necessity in Natural Systems." Journal Article. *ICES Journal of Marine Science* 77 (4): 1573–88. https://doi.org/10.1093/icesjms/fsz173.