CDA graphs

Loading modules

In [1]:

```
import pandas as pd
from plotly import __version__
import plotly.tools as tls
import plotly.plotly as py
from plotly.graph_objs import *
from plotly.offline import download_plotlyjs, init_notebook_mode, iplot
import plotly.graph_objs as go
init_notebook_mode()
```

Importing data

In [13]:

```
fams_repland = pd.read_csv("../ipython_data/fams_teleost.RepeatLandscapes.fixe
d.csv",
                           skipinitialspace=True, sep = " ", index_col=False)
annot = pd.read_csv("../ipython_data/names.tab", encoding = "UTF-8", sep = "\t
",
                    names = ["ALIAS", "ORDER", "SPECIES", "COMMON_NAME"])

rl = pd.merge(annot, fams_repland)

# The de novo dataframe is exactly as the total data frame in structure,
# but without the repeats detected only in RepBase.

denovo = pd.read_csv("../ipython_data/denovo_only.dataframe", sep = "\t", inde
x_col = False, na_values='na',
                        names = ["ALIAS", "NAME", "CLASS", "FAM", "CLASS_FAM",
"FULL",
                                 "LENGTH", "FRG", "FULL_FRG", "NR_FRG", "AVG_DI
V",
                                 "MED_DIV", "AVG_DEL", "MED_DEL", "AVG_INS", "M
ED_INS",
                                 "LEN_MASKED", "AVG_LEN_MASKED", "MED_LEN_MASKE
D", "GENOME_PERC",
                                 "LEN_OVERLAP", "GENOME_PERC_OVERLAP", "LEN_MAS
KED_OVERLAP"])
denovo = pd.merge(denovo, annot)

# Changing wrongly spelled name of Haddock.
denovo['COMMON_NAME'][denovo.COMMON_NAME == 'Haddoc'] = 'Haddock'

alias74 = pd.read_table('../ipython_data/74aliases', names = ['ALIAS'])
```

```
/Users/williambrynildsen/.local/lib/python2.7/site-packages/ipyker
nel/__main__.py:20: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
```

Data manipulation steps for making CDA graphs

In [14]:

```
fam_plot = rl.groupby(['ALIAS', 'SPECIES', 'Rfam', 'ORDER', 'COMMON_NAME']).su
m().reset_index()
fam_plot = pd.merge(alias74, fam_plot)
```

In [15]:

```
fam_plot.columns.values[5:] = ['1','2','3','4','5','6','7','8','9','10','11','
12','13','14','15','16','17','18','19','20',
                               '21','22','23','24','25','26','27','28','29',
'30','31','32','33','34','35','36','37',
                                '38','39','40','41','42','43','44','45','46
','47','48','49','50']
```

I am interested in the activities of the top five TE families/superfamilies in each fish genome.

In [6]:

```
tt_frame = denovo.groupby(['SPECIES', 'FAM']).sum()[['GENOME_PERC']].reset_ind
ex()
```

In [8]:

```
# This function will give me the top five (the number can be changed) TE famil
ies by genome coverage
def get_top_fams(species, number):
    return tt_frame[(tt_frame['SPECIES']==species) &
                    (tt_frame['FAM']!='DNA') &
                    (tt_frame['FAM']!='Simple_repeat') &
                    (tt_frame['FAM']!='Unknown') &
                    (tt_frame['FAM']!='Low_complexity') &
                    (tt_frame['FAM']!='LTR') &
                    (tt_frame['FAM']!='SINE') &
                    (tt_frame['FAM']!='LINE')].sort_values(by='GENOME_PERC', as
cending = False).head(n = number)
```

This is the function for making CDA graphs

In [9]:

```
def repeat_landscape(df,fam):
    def make_trace(df):
        colors = {'hAT-Ac' : 'AEA01D',
                  'Gypsy' : '68402B',
                  'DIRS' : '2696D8',
                  'L2' : 'FDF7B5',
                  'TcMar-Tc1' : 'F93B67',
                  'Rex-Babar' : '8C5D58',
                  'PIF-Harbinger' : '22428B',
                  'RTE-BovB' : 'E2D3C9',
                  'hAT-Charlie' : 'A1AFC9',
                  'ERV1' : 'D67336',
                  'R2-Hero' : 'E63EAB',
                  'hAT-Tip100' : '96DEAF',
                  'Kolobok-T2' : '74578E',
                  'Pao' : '818092',
                  'Penelope' : '3A2E30',
                  'PIF-ISL2EU' : '95CEF4',
                  'RTE-X' : 'A8DACD',
                  'TcMar-Tc2' : '525D29',
```

```
            'TcMar-Mariner' : 'Black',

            'Jockey' : 'Blue',
            'L1' : 'Purple',
            'Maverick' : 'Pink',
            'L1-Tx1' : 'Turqoise',
            'Copia' : 'Gray',
            'Gypsy' : 'Green',
            'Gypsy-Cigr' : 'Black',
            'PiggyBac' : 'White',
            'CR1' : 'Gold',
            'Dong-R4' : 'Silver',
            'hAT' : 'Orange',
            'TcMar-Tigger' : 'Red',
            'Academ' : 'Beige',
            'Crypton' : 'Blue'}


#    Gets the index for each superfamily, in each fish

def make_dict(fam, df):
    dicto = {}
    for i in fam:
        dicto.update({i : df[df['Rfam']==i].index.values[0]})
    return dicto

data = []
for i in fam:
    if i in df.Rfam.unique():
            trace = go.Bar(
                x = df.columns[1:],
                y = df[df['Rfam']==i].loc[make_dict(fam, df)[i]][1:1000],
                name = i,
                marker=dict(color = colors[i]),
                )
            data.append(trace)
    return data

    stacklayout = go.Layout(
            barmode = 'stack',
            xaxis = dict(autotick = False, #title = 'Divergence from conse
nsus (%)',
                         zeroline = True,
                         showline = True,
                         showticklabels = True,
                         mirror='ticks',
                         zerolinewidth = 2,
                         linewidth = 2,
                         dtick = 5),
            yaxis = dict(autotick = True, #title = 'Percentage of genome (
%)',
                         autorange = True, showgrid = False,
                         zeroline = True,
                         showline = True,
                         mirror = 'ticks',
                         range = [0,1,2,3,4,5],
                         zerolinewidth = 2,
                         linewidth = 2),
```

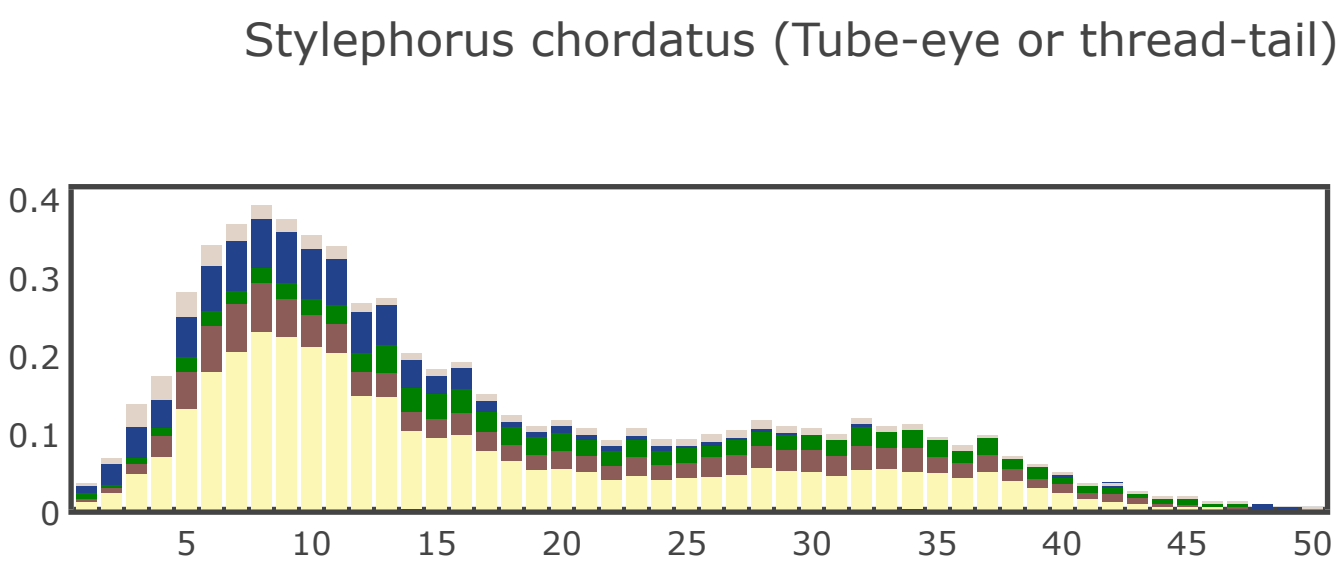
```
                height = 300,
                width = 700,
                plot_bgcolor = 'White',
                paper_bgcolor = 'White',
                title = '%s (%s)' % (df[df['SPECIES']==i].SPECIES.unique()[0],
df[df['SPECIES']==i].COMMON_NAME.unique()[0]),
                showlegend = True,
                legend = dict(font=dict(size=12), traceorder = 'normal'),
                font = dict(size = 12)
        )
    return go.Figure(data = make_trace(df), layout = stacklayout)
```

In [11]:

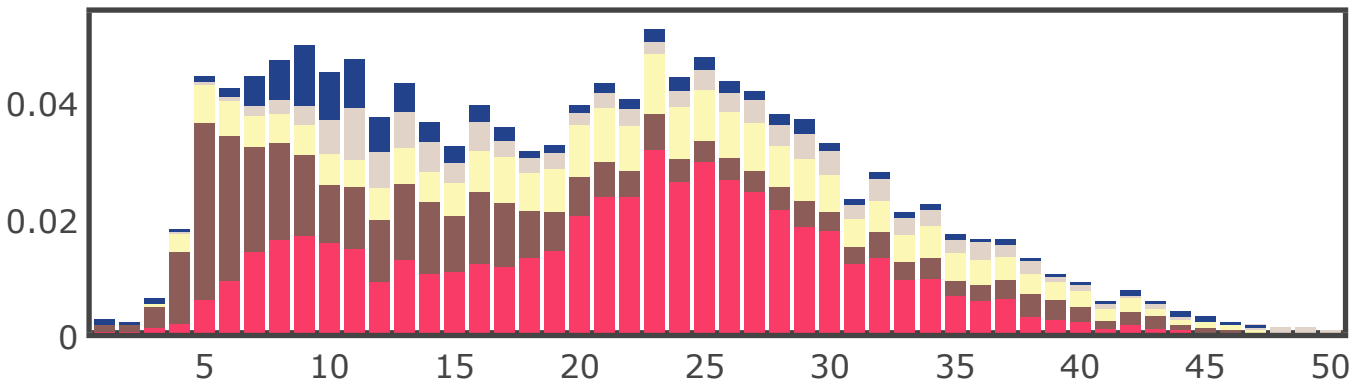
```
# Generating CDA graphs for each fish
```

In [18]:

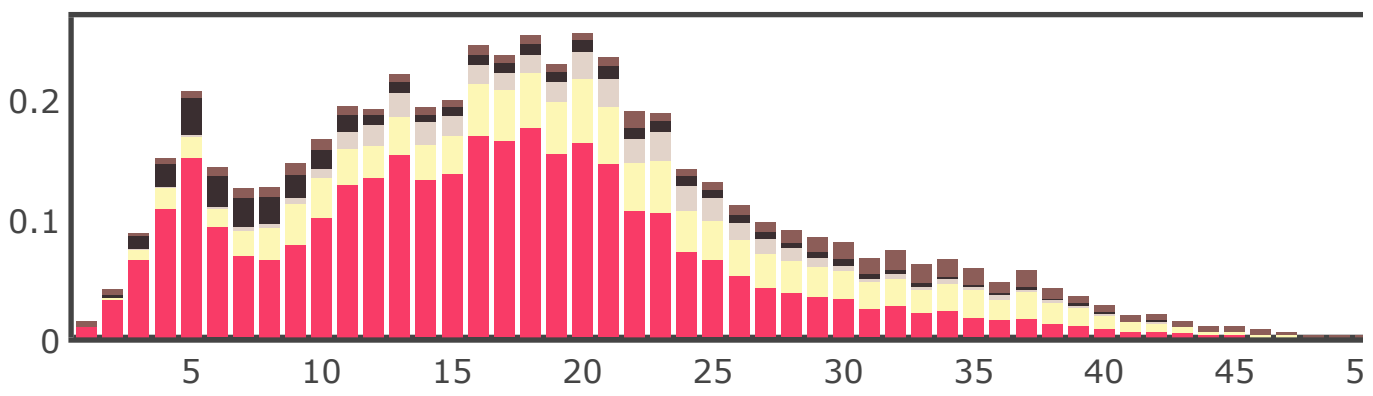
```
for i in set(fam_plot.SPECIES):
    iplot(repeat_landscape(fam_plot[fam_plot['SPECIES']==i],
                                    get_top_fams(i, 5).FAM.tolist()))
    py.image.save_as(repeat_landscape(fam_plot[fam_plot['SPECIES']==i],
                                    get_top_fams(i, 5).FAM.tolist()),
                     '../figures/RLfigures/%s.pdf' % fam_plot[fam_plot['SPECIES
']==i].ALIAS.unique()[0]\
                      , format='pdf', scale=4)
```



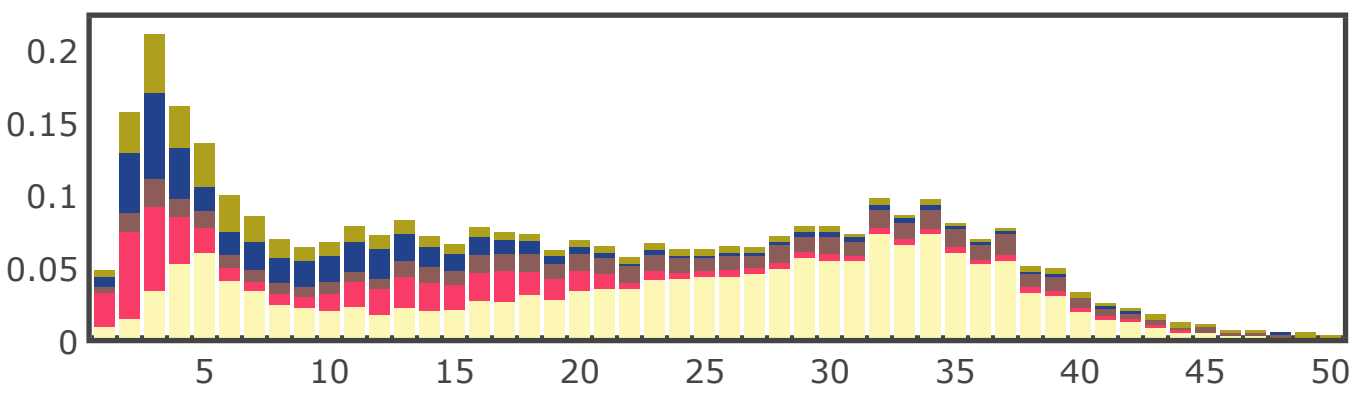
Selene dorsalis (African moonfish)



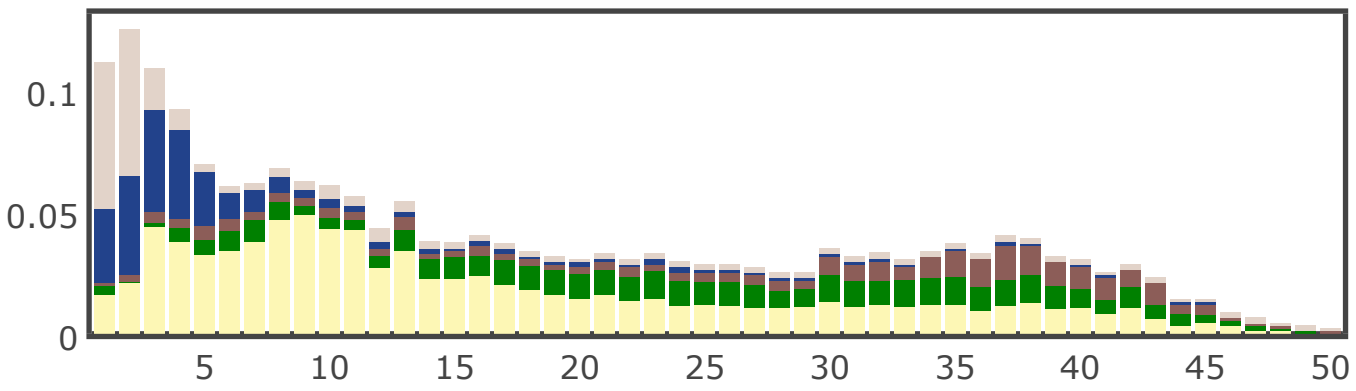
Chromis chromis (Damselfish)



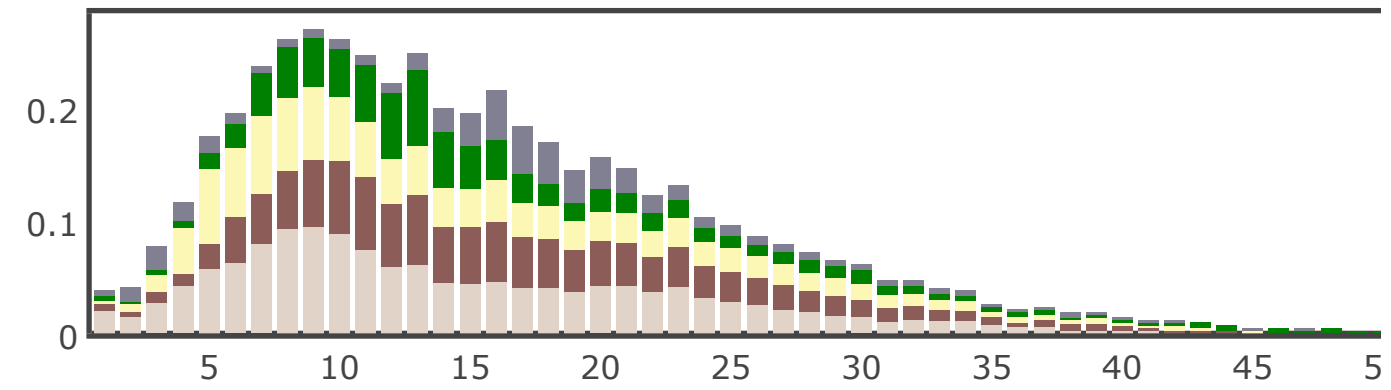
Rondeletia loricata (Redmouth whalefish)



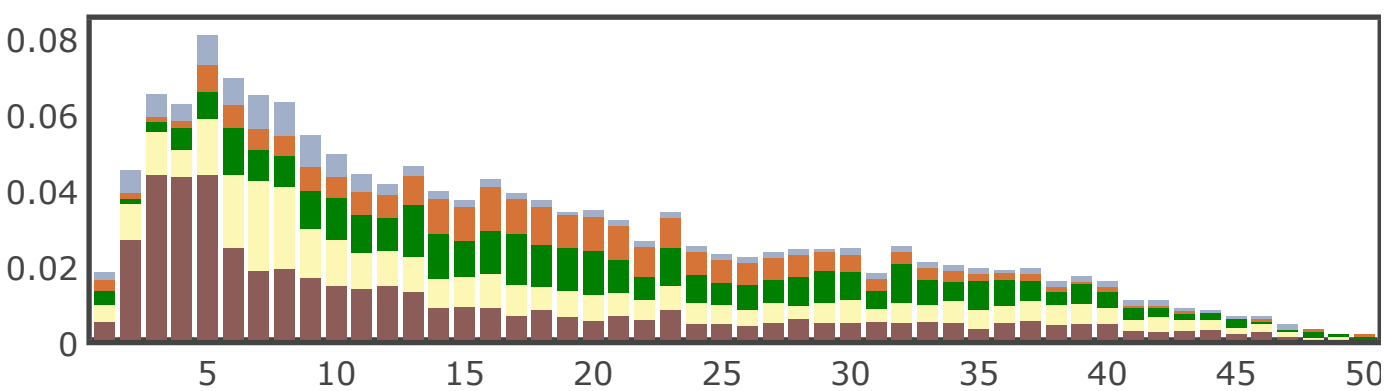
Arctogadus glacialis (Arctic cod)



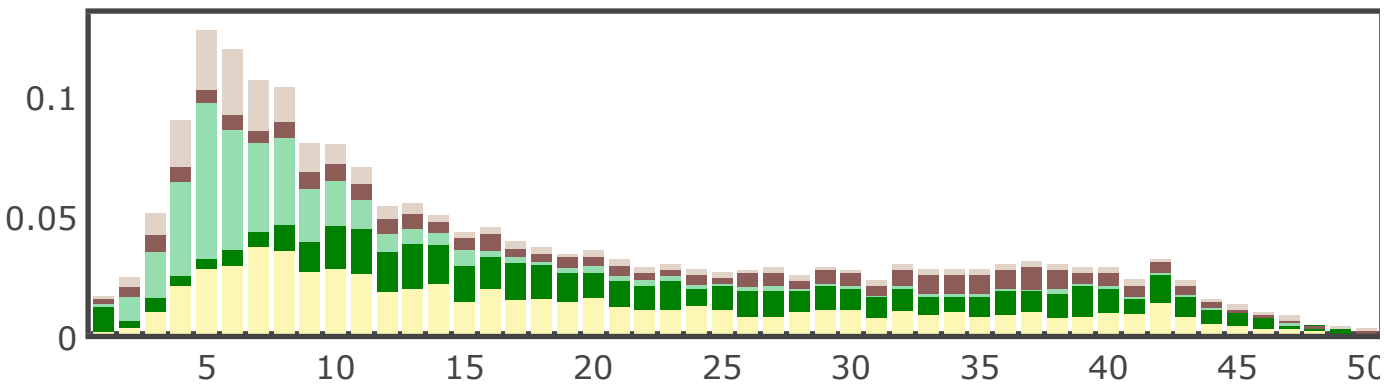
Bregmaceros cantori (Striped codlet)



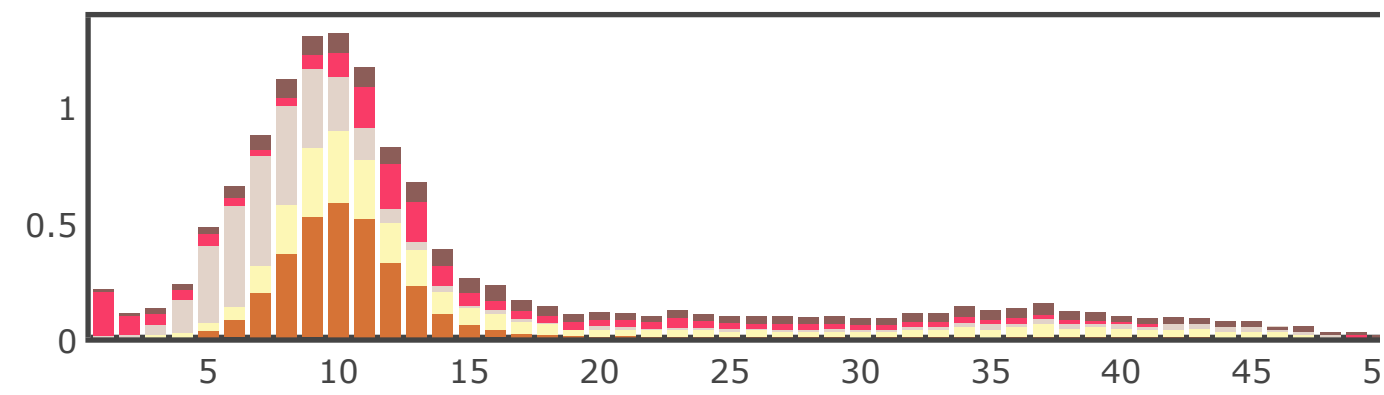
Trachyrincus scabrus (Roughsnout grenadier)



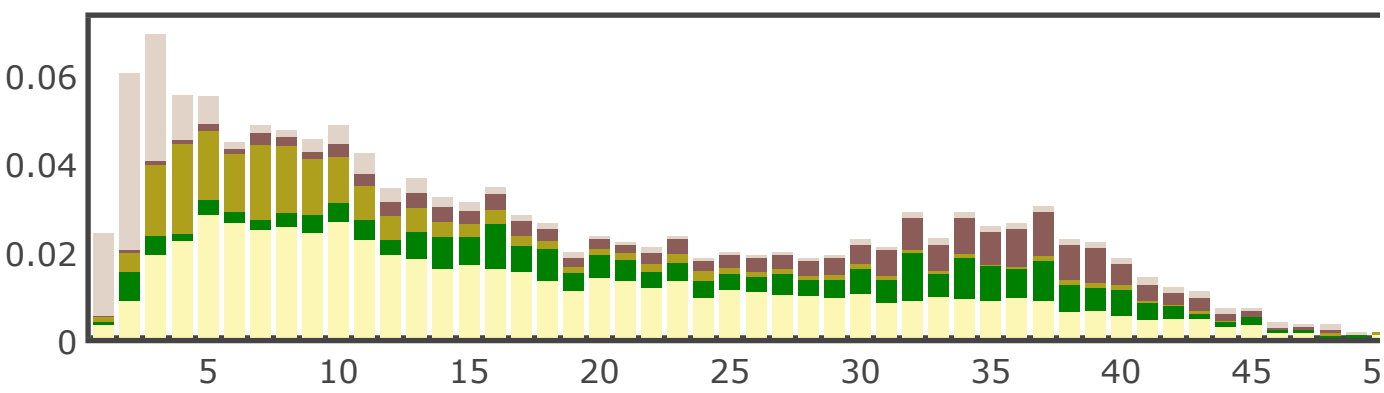
Trisopterus minutus (Poor cod)



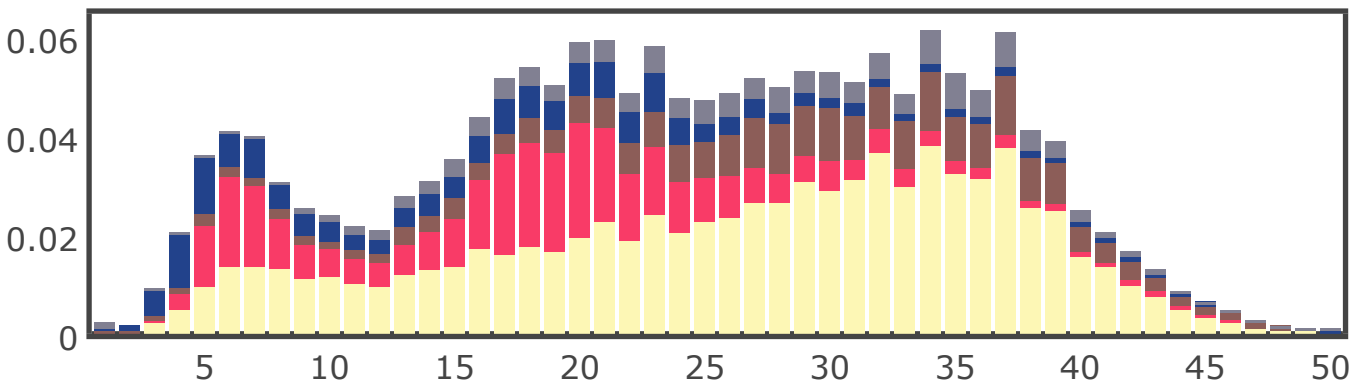
Lampris guttauts (Opah)



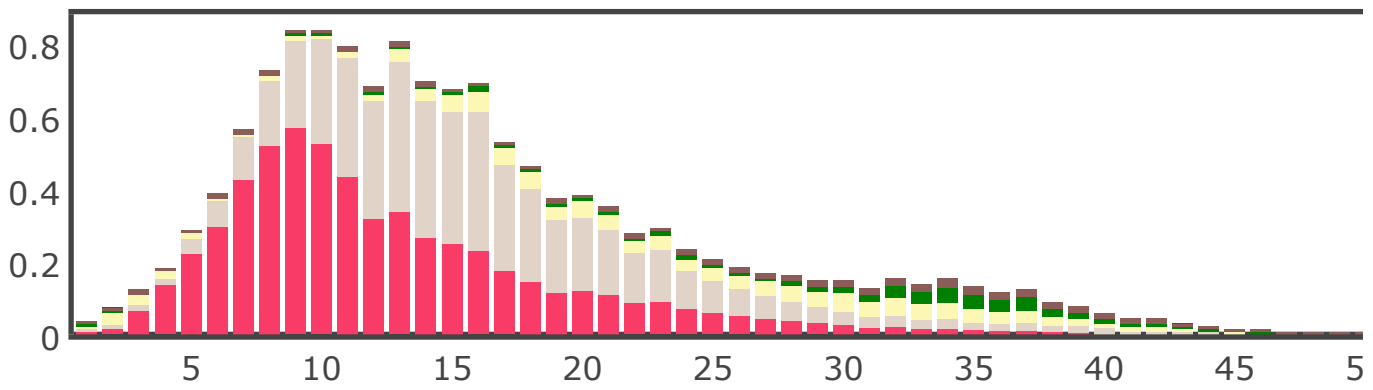
Gadus morhua (Atlantic cod)



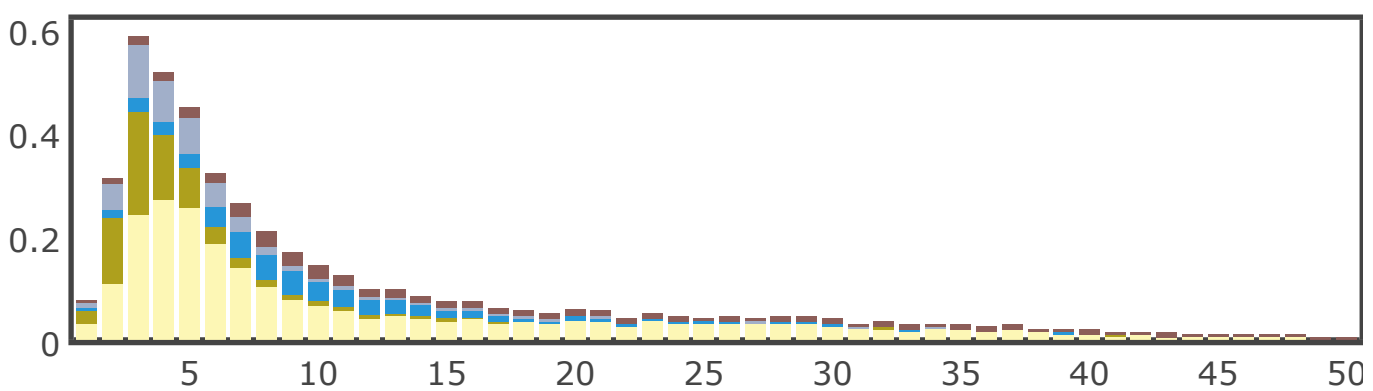
Beryx splendens (Splendid alfonsino)



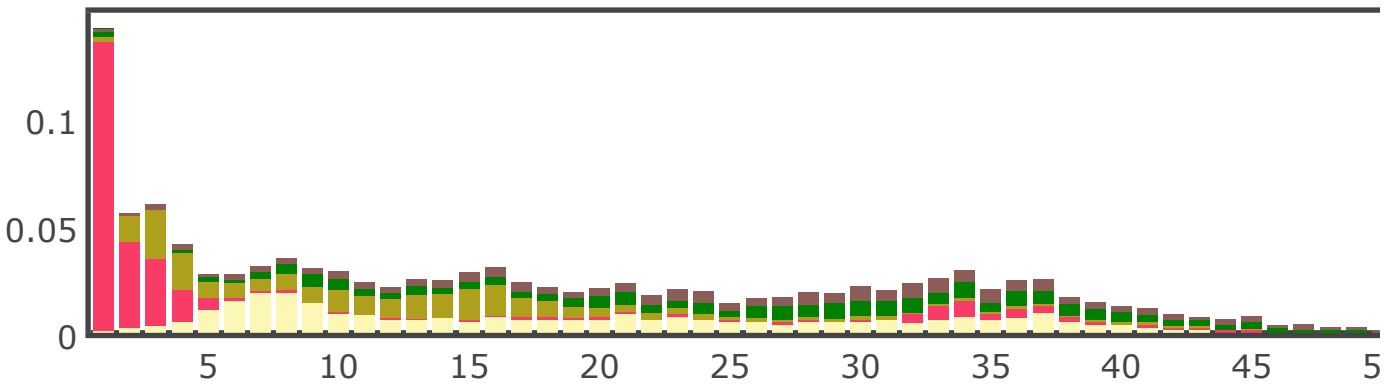
Chatrabus melanurus (Pony toadfish)



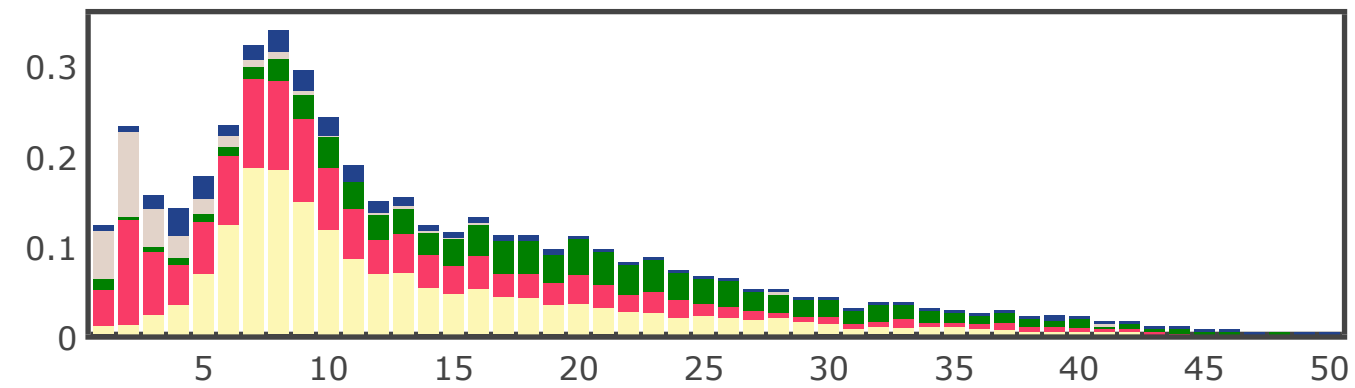
Chaenocephalus aceratus (Blackfin icefish)



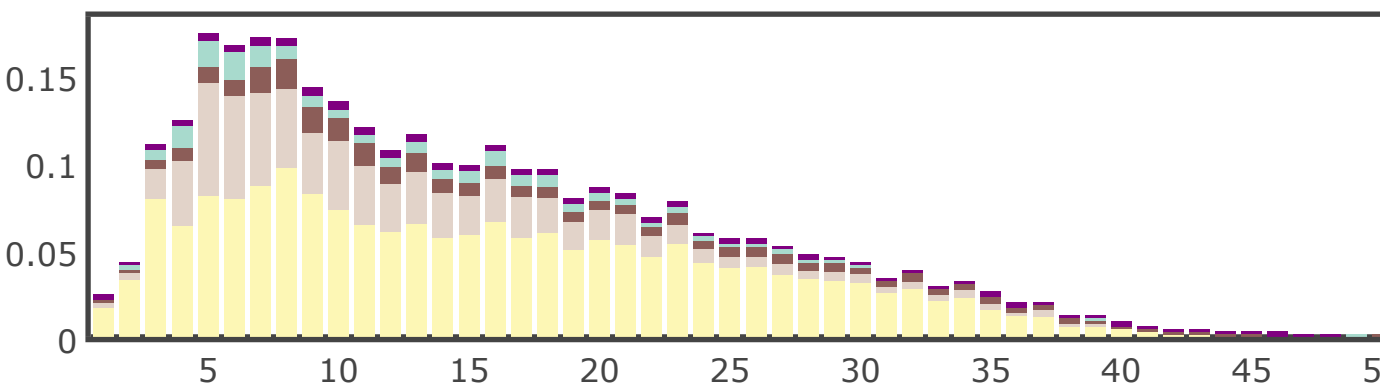
Molva molva (Ling)



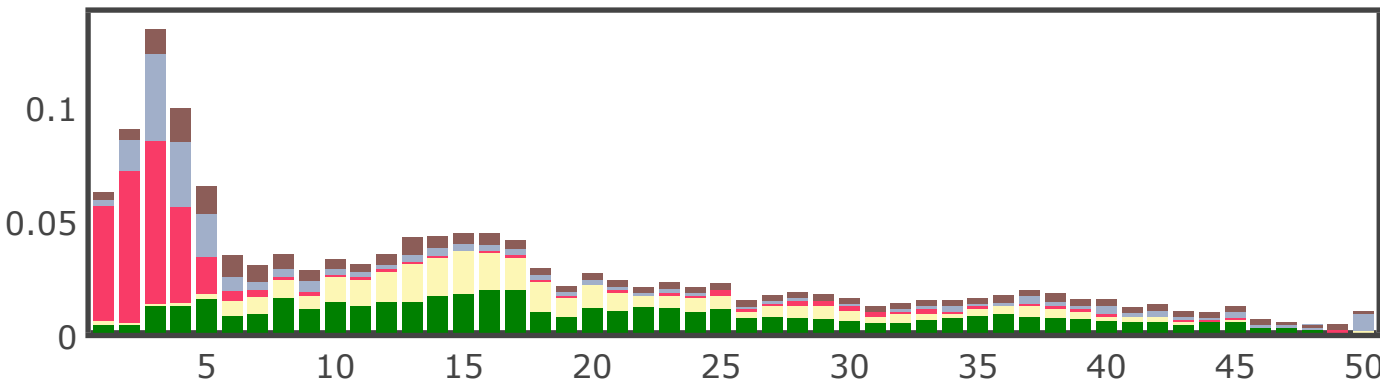
Percopsis transmontana (Sandroller)



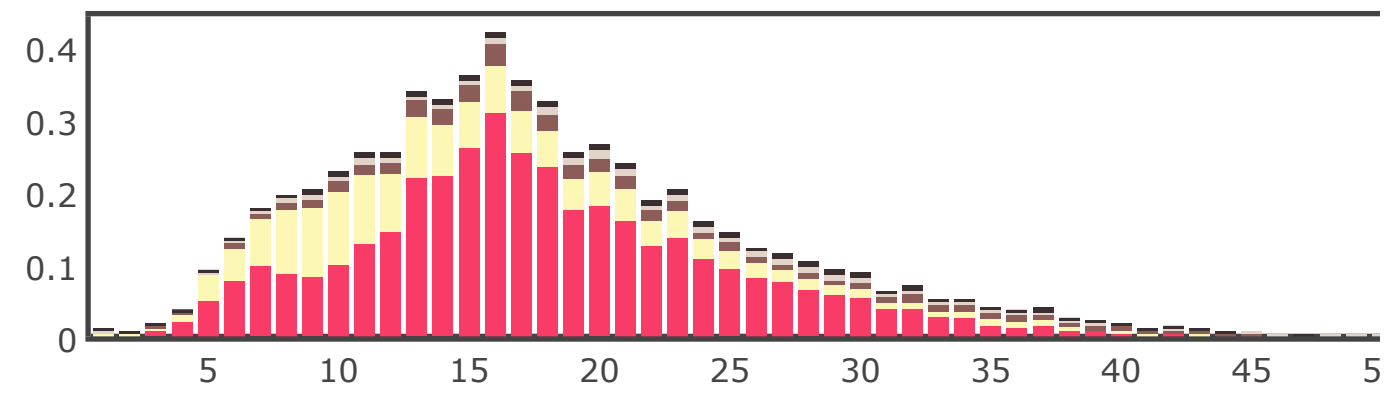
Benthosema glaciale (Glacier lantern fish)



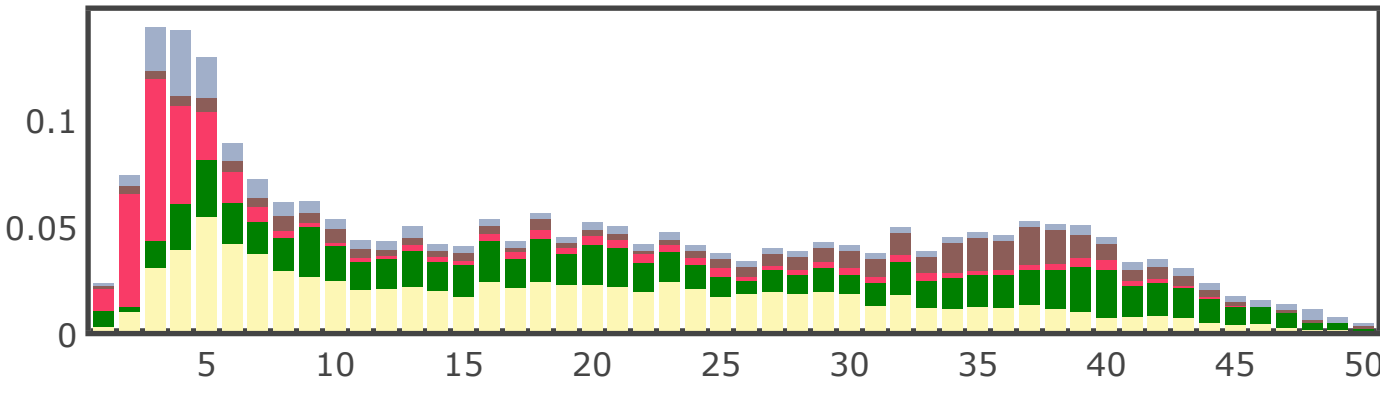
Lota lota (Burbot)



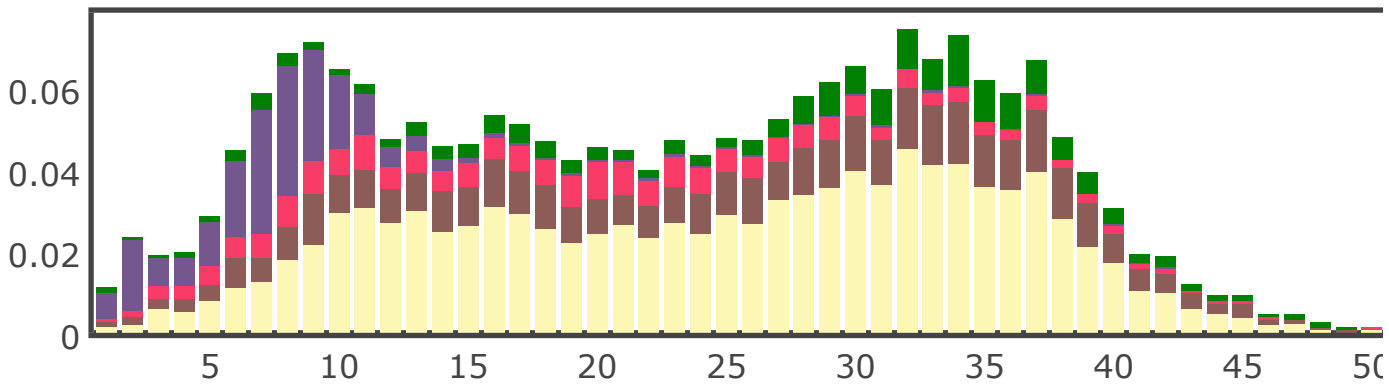
Helostoma temminckii (Kissing gouramis)



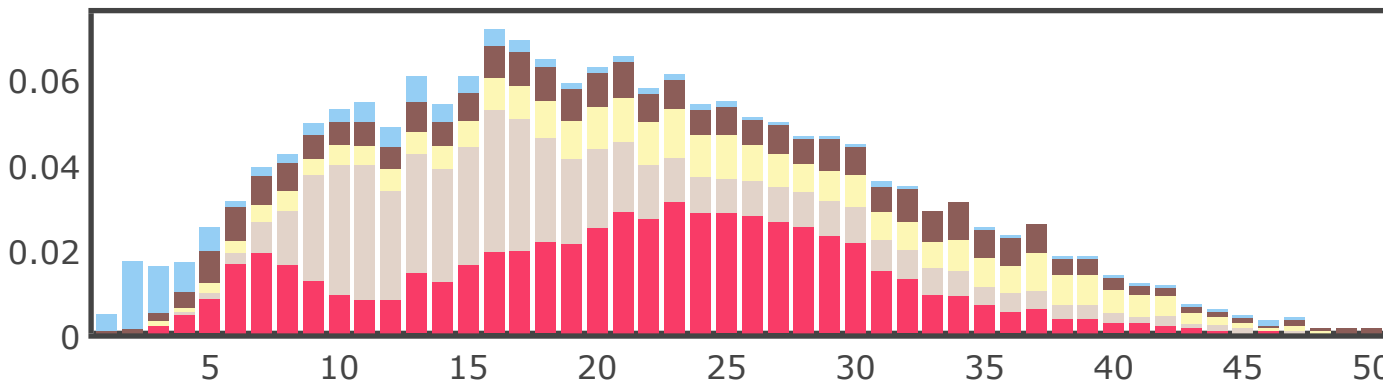
Melanogrammus aeglefinus (Haddoc)



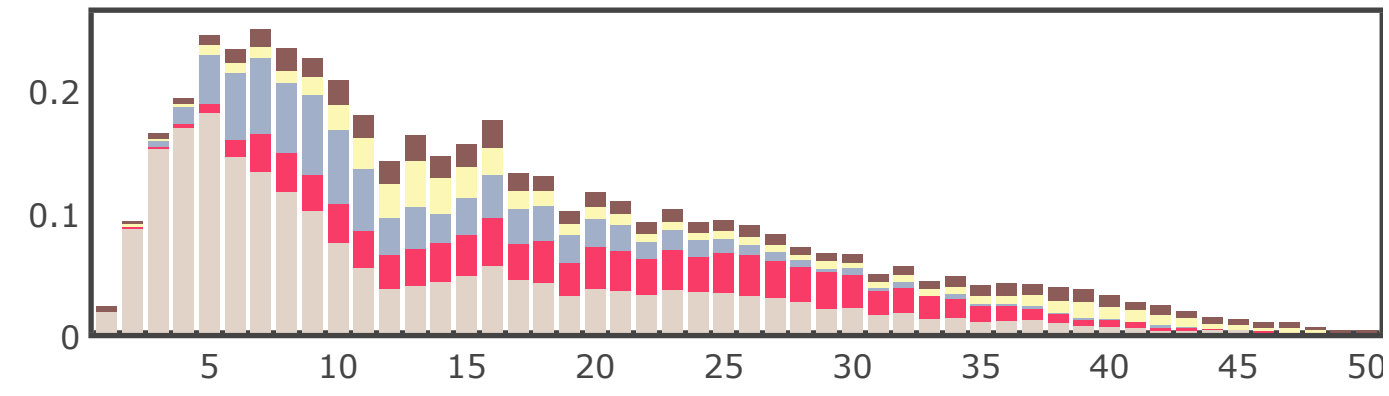
Polymixia japonica (Silver eye beardfish)



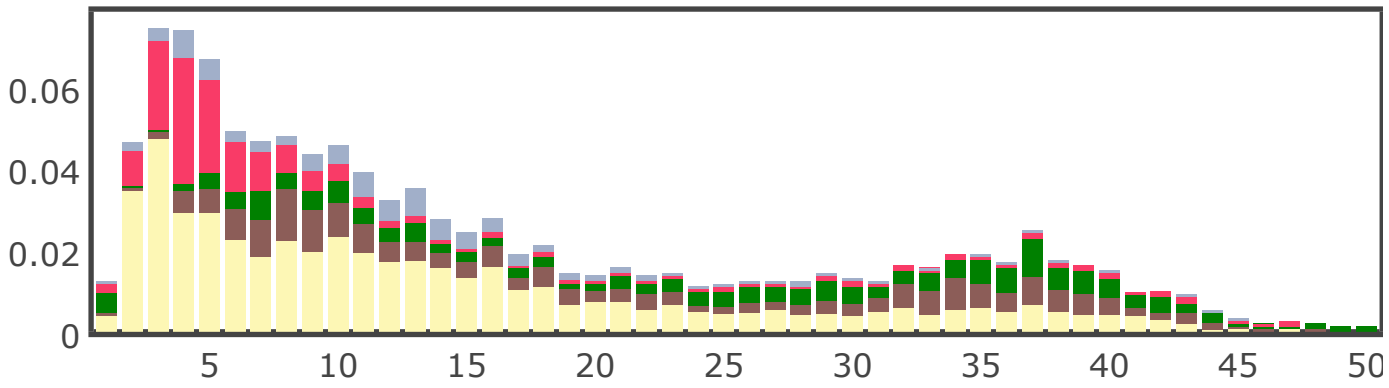
Pseudochromis fuscus (Brown dottyback)



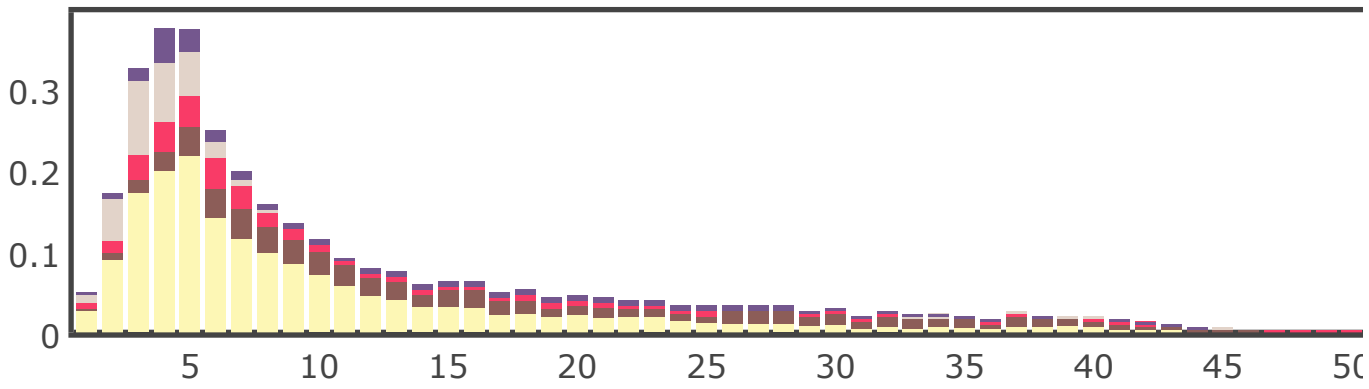
Antennarius striatus (Striated frogfish)



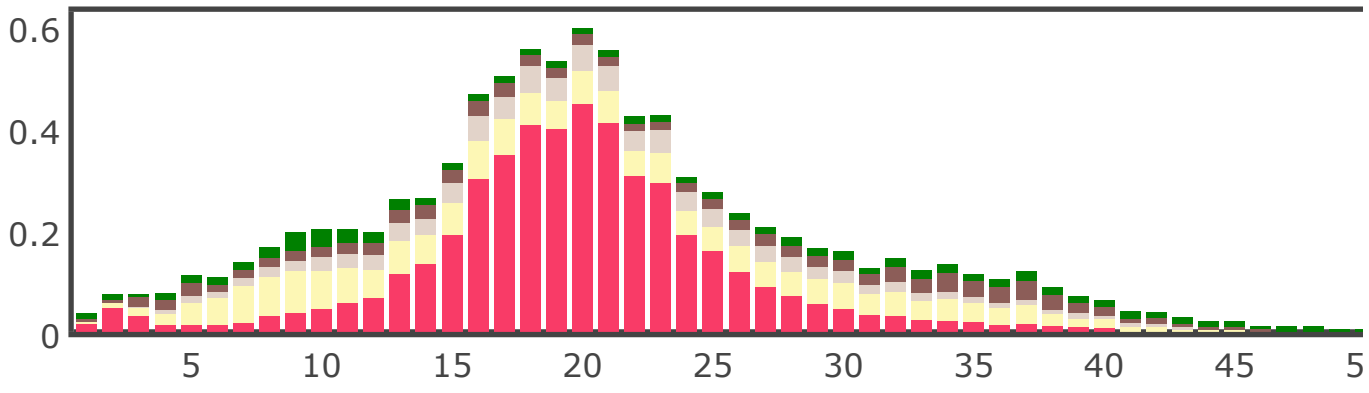
Laemonema laureysi (Guinean codling)



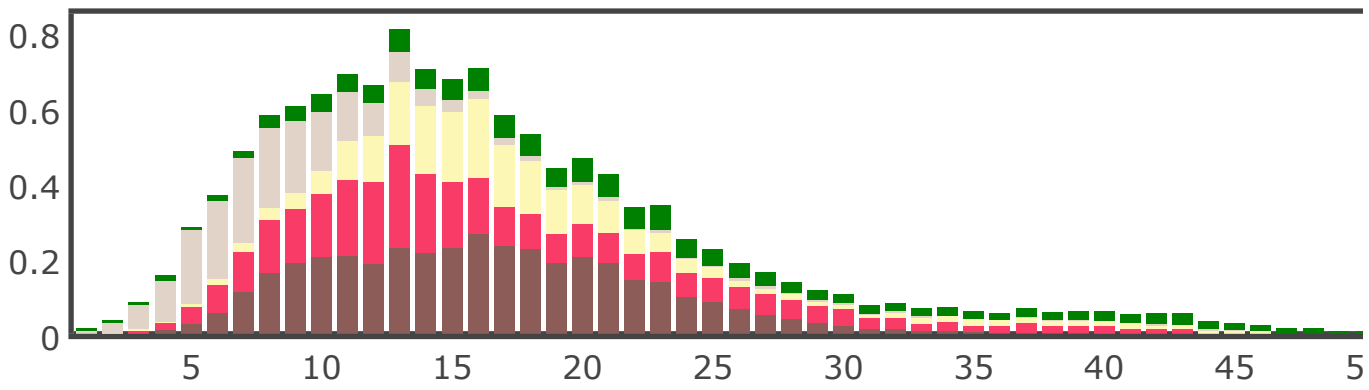
Phycis phycis (Forkbeard)



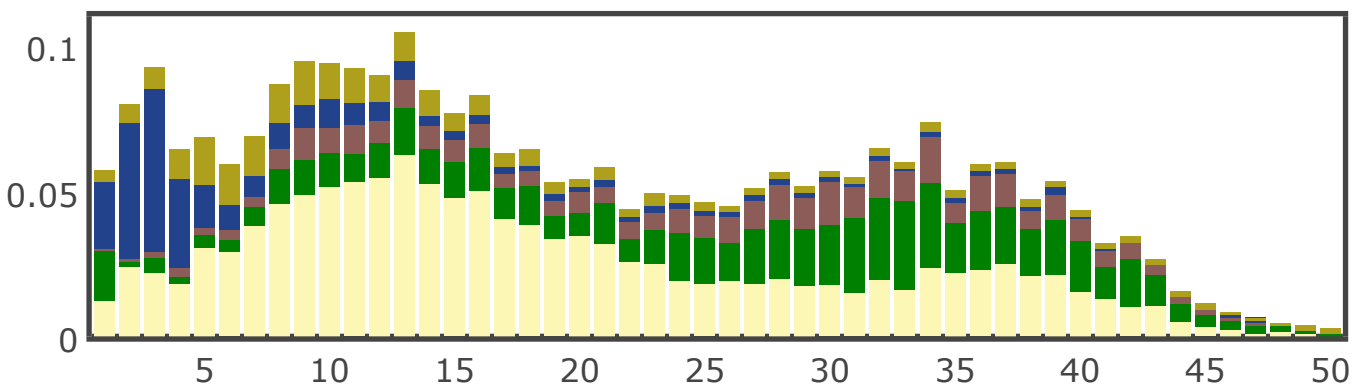
Oreochromis niloticus (Nile tilapia)



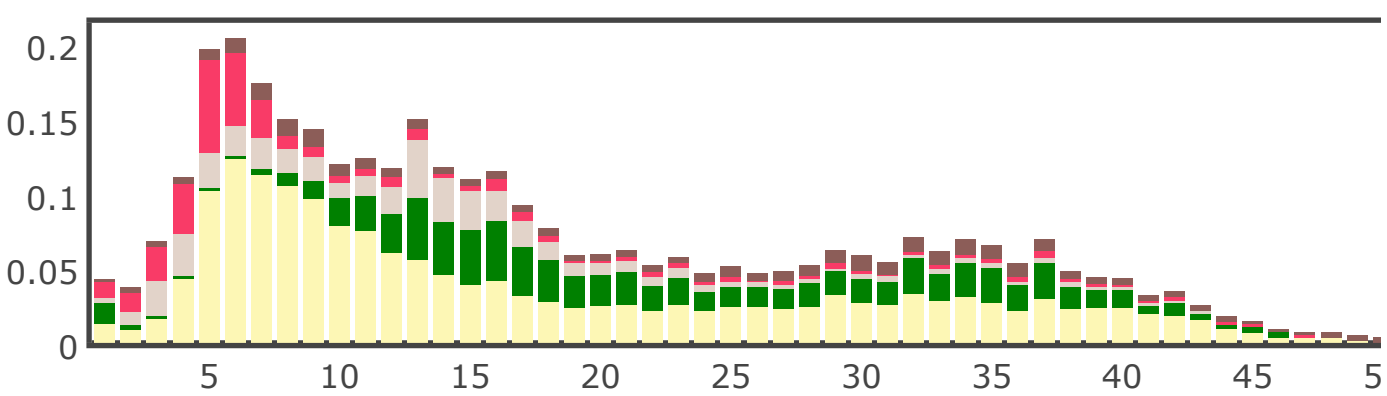
Guentherus altivela (Jellynose)



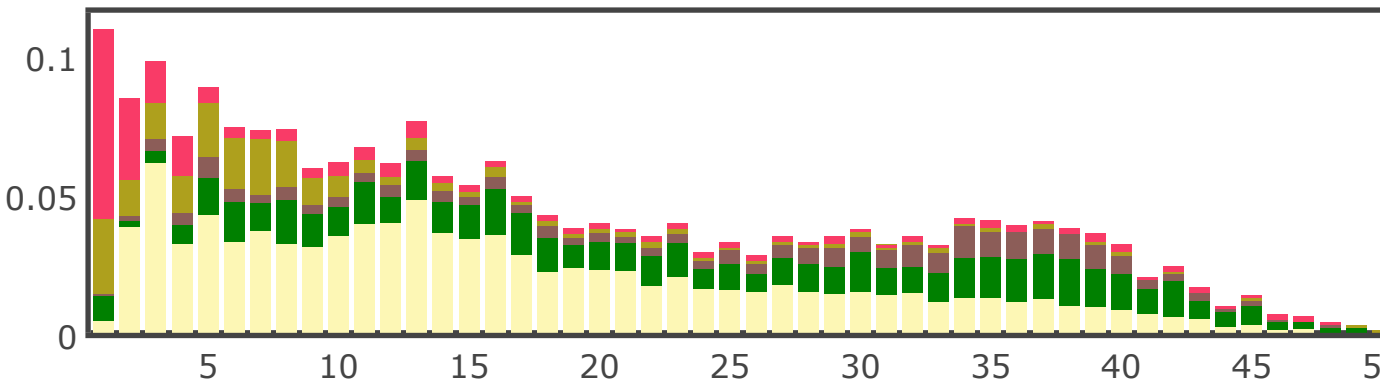
Merluccius polli (Black hake)



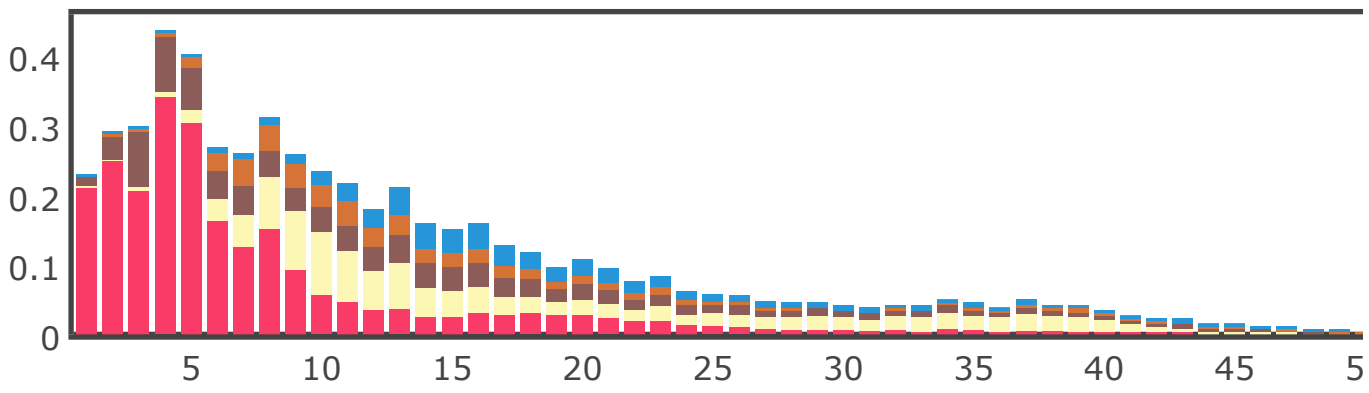
Melanonus zugmayeri (Arrowtail)



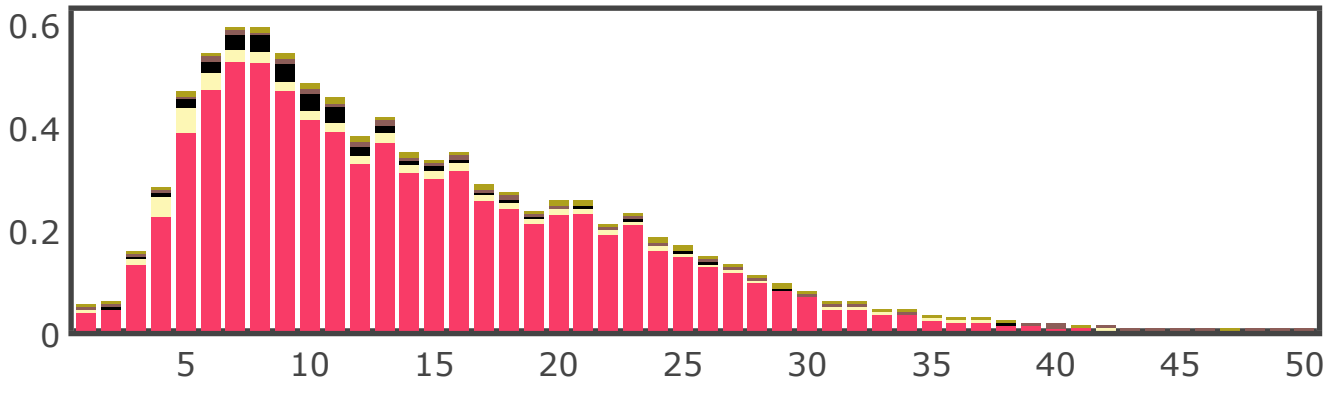
Pollachius virens (Saithe)



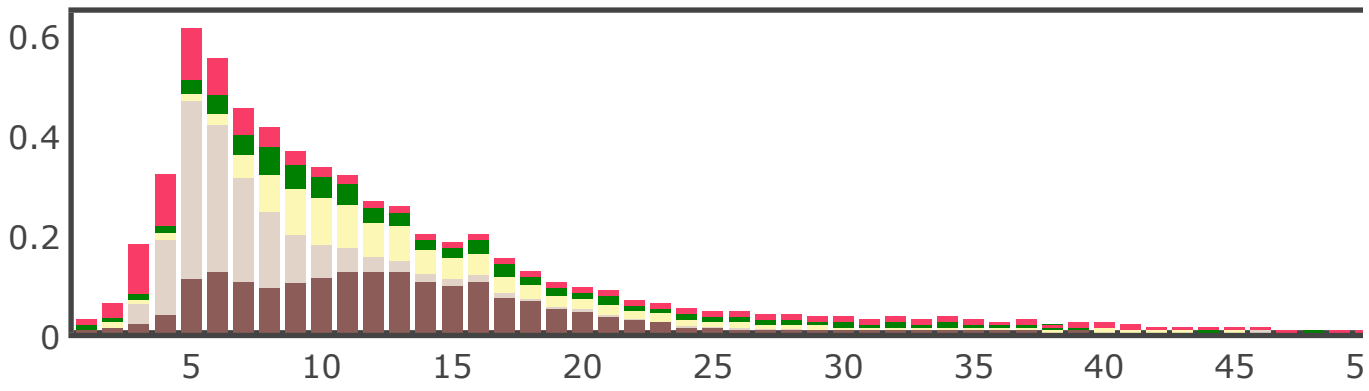
Cyttopsis roseus (Red dory)



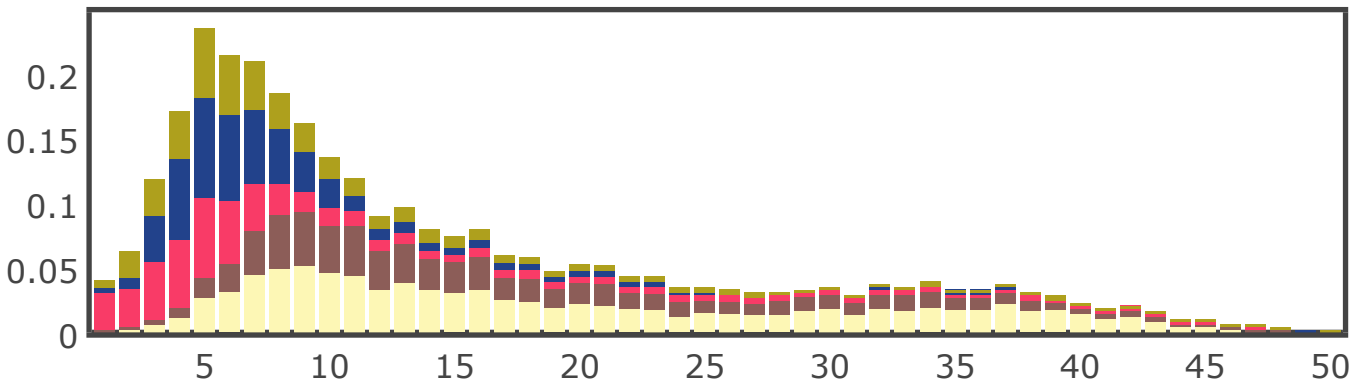
Astyanax mexicanus (Blind cave fish)



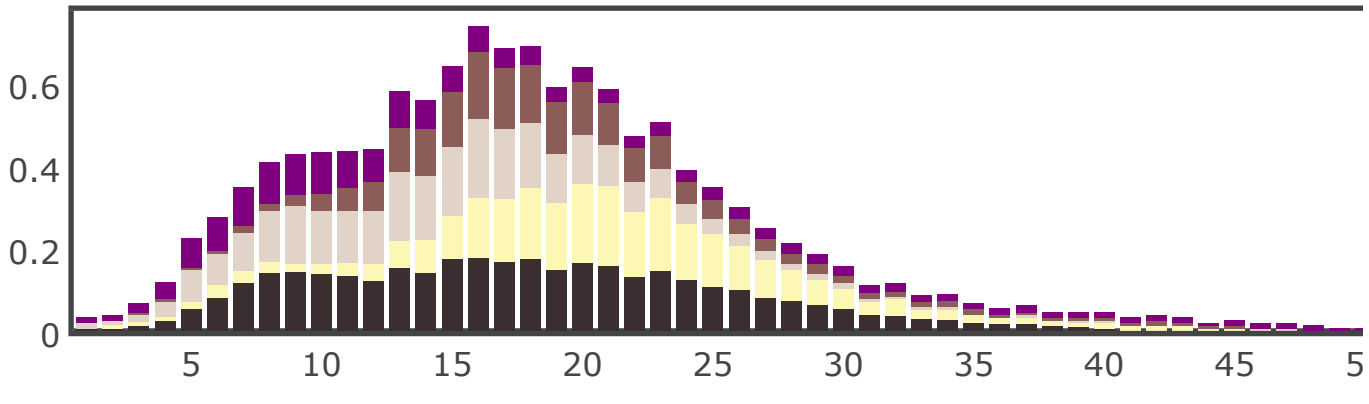
Malacocephalus occidentalis (Western softhead grenac



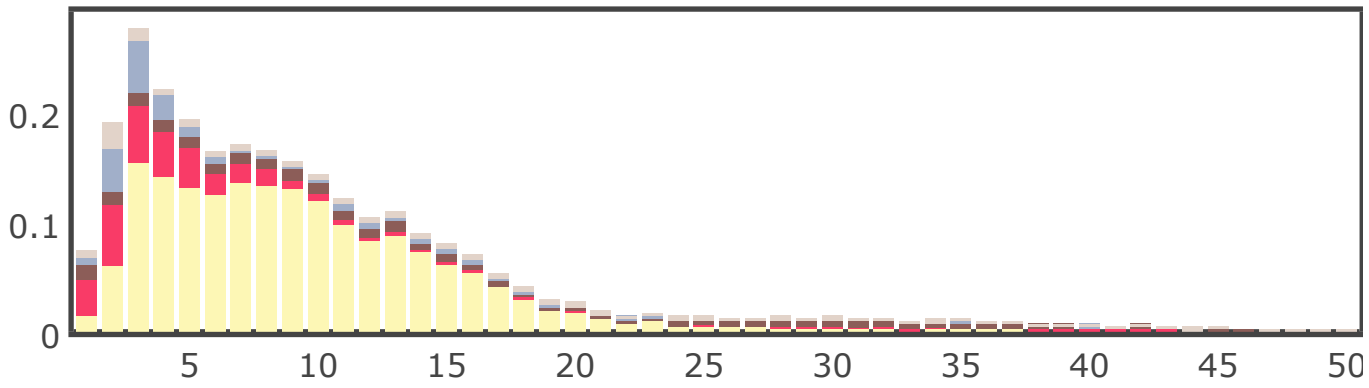
Perca fluviatilis (European perch)



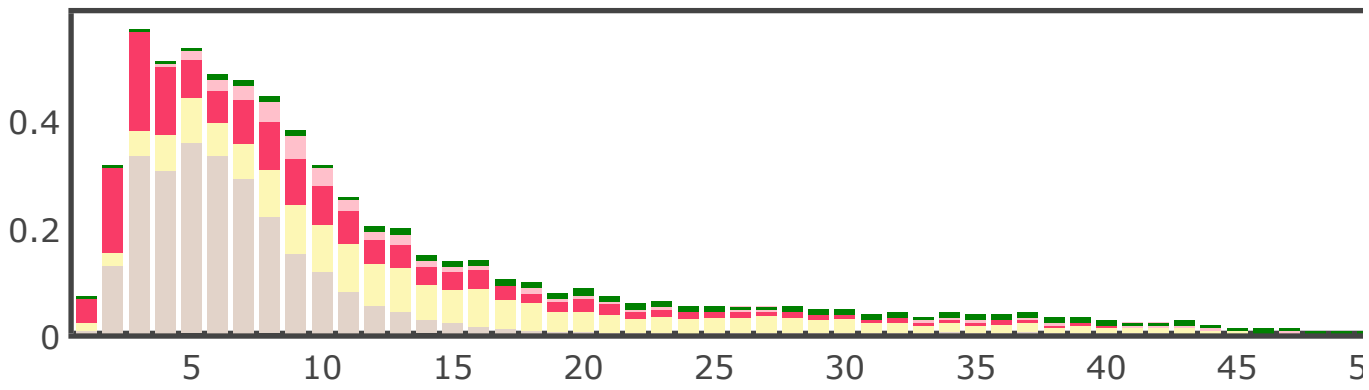
Regalecus glesne (King of herring)



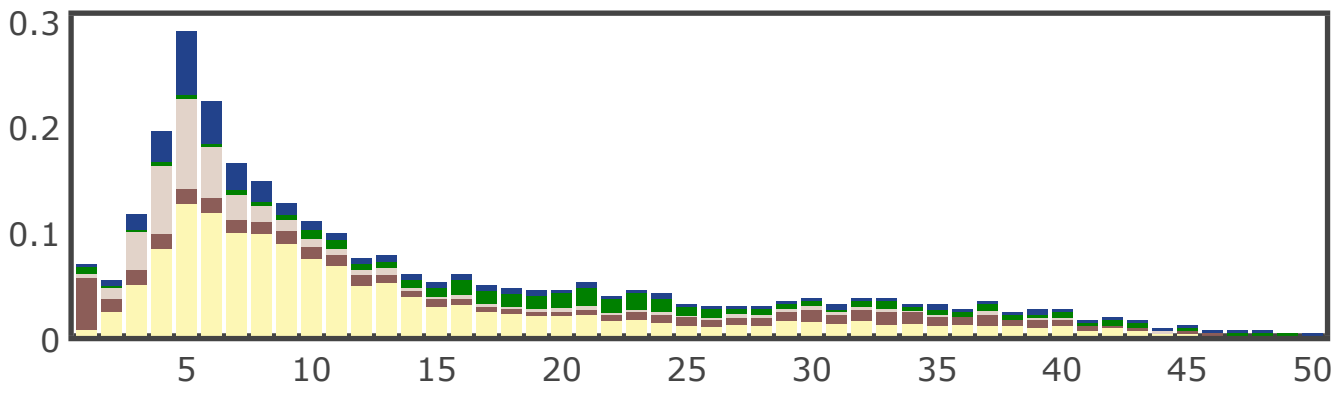
Bathygadus melanobranchus (Vaillants grenadier)



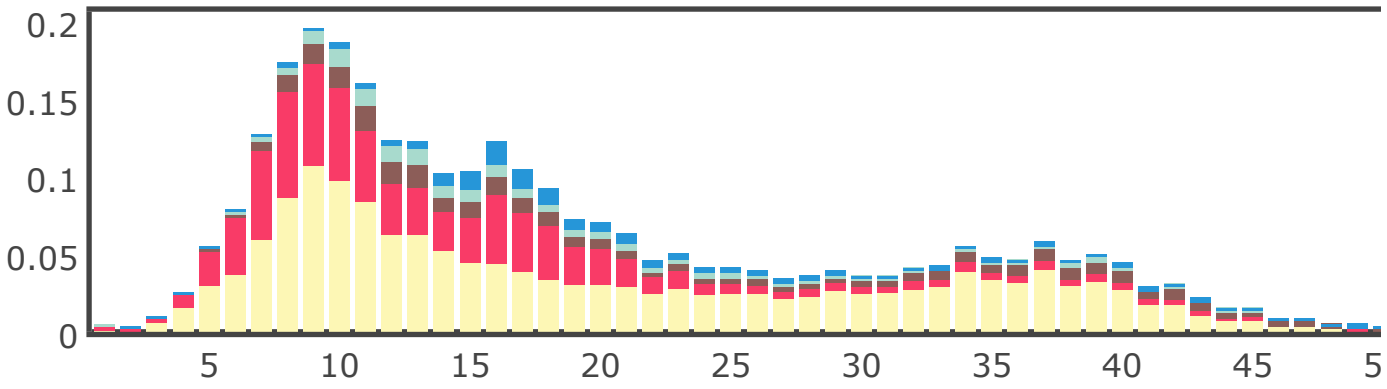
Typhlichthys subterraneus (Southern cavefish)



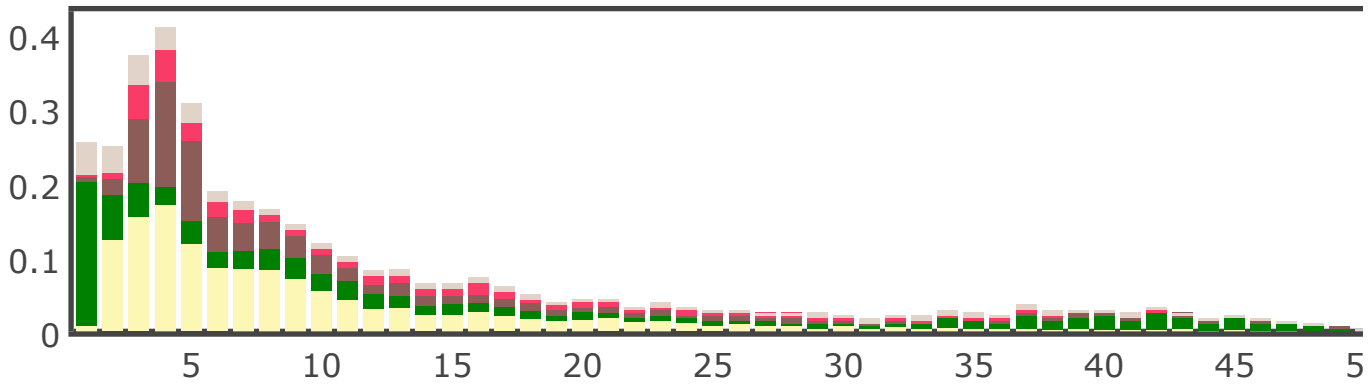
Gadiculus argenteus (Silvercod)



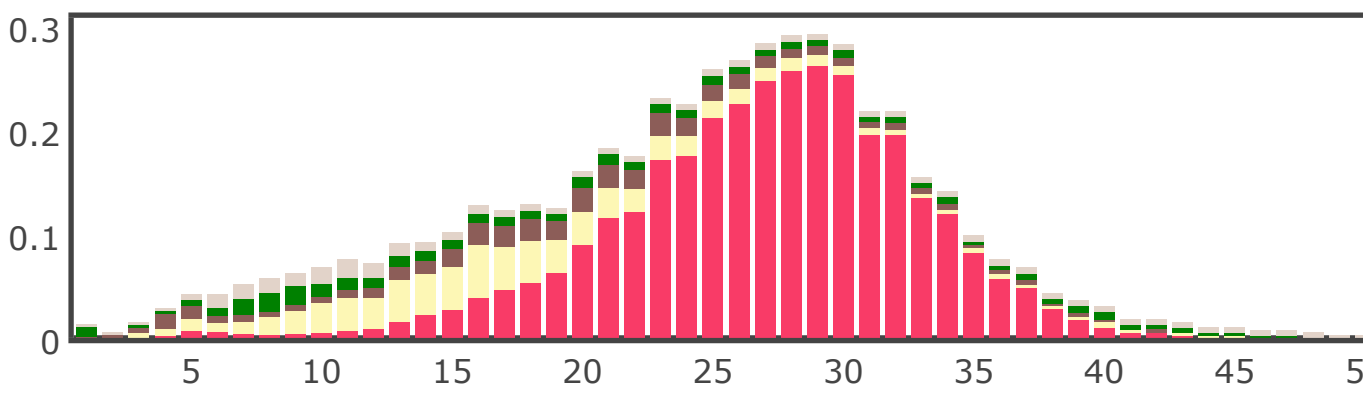
Thunnus albacares (Yellowfin tuna)



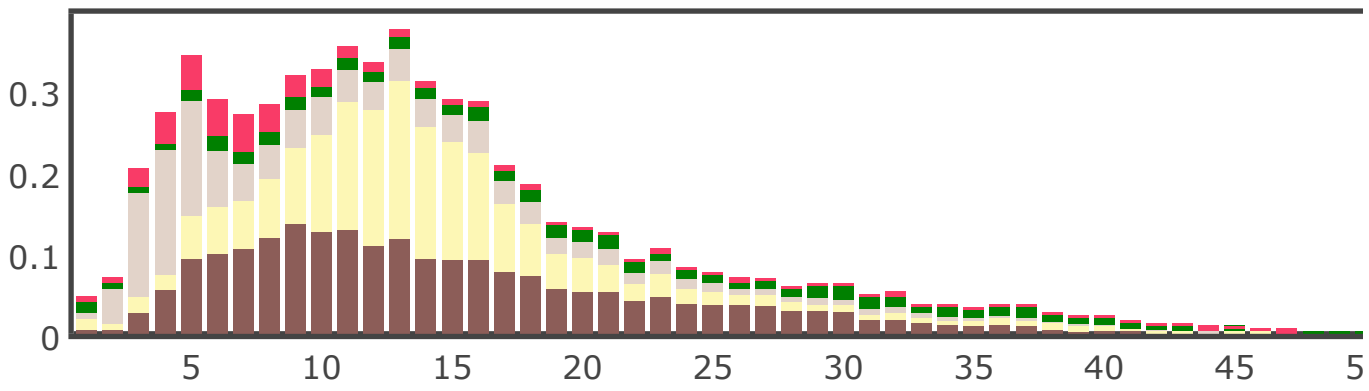
Gasterosteus aculeatus (Three-spined stickleback)



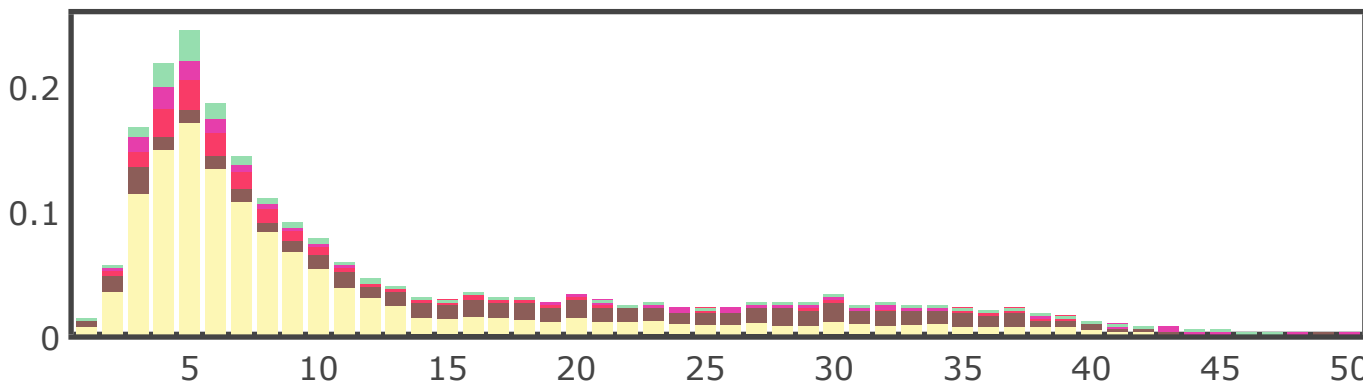
Poecilia formosa (Amazon molly)



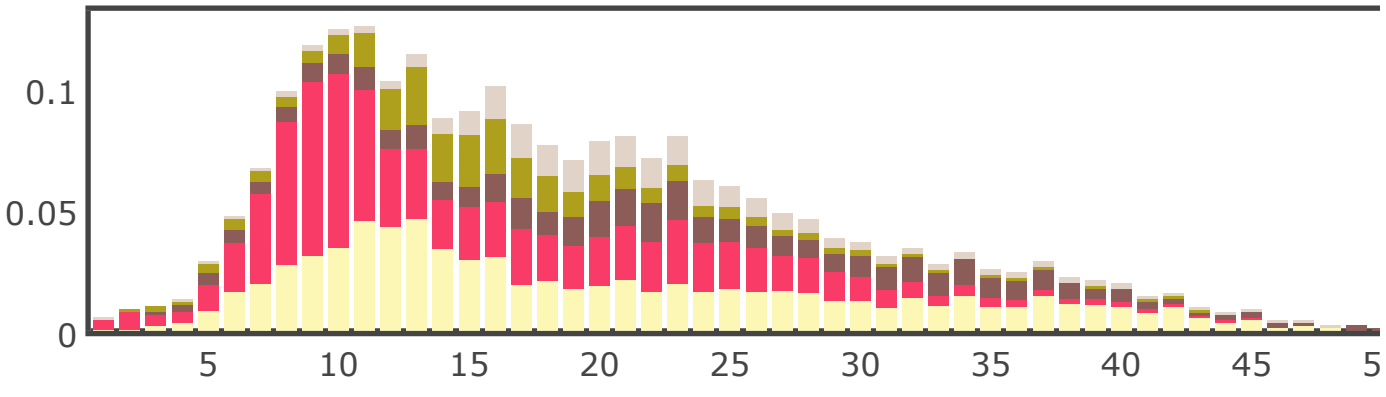
Macrourus berglax (Roughhead grenadier)



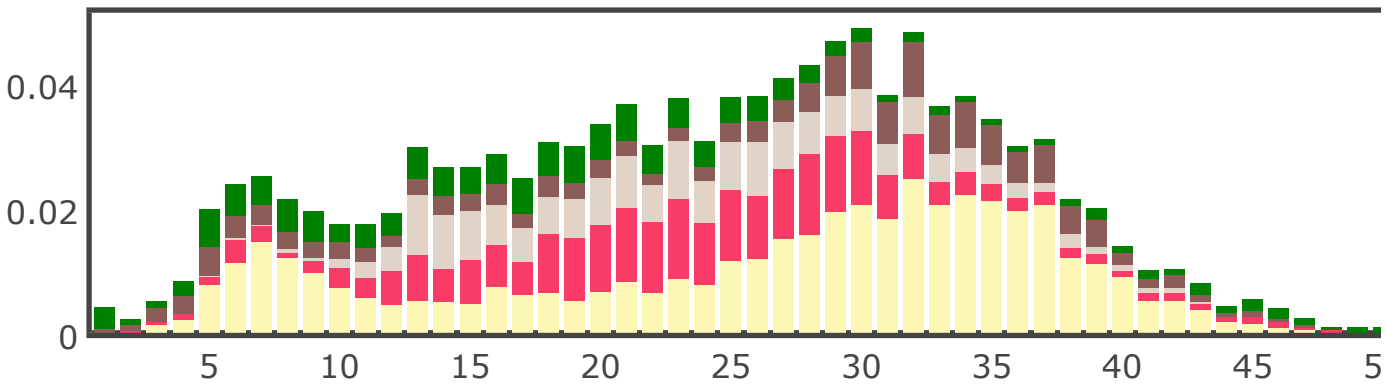
Phycis blennoides (Greater forkbeard)



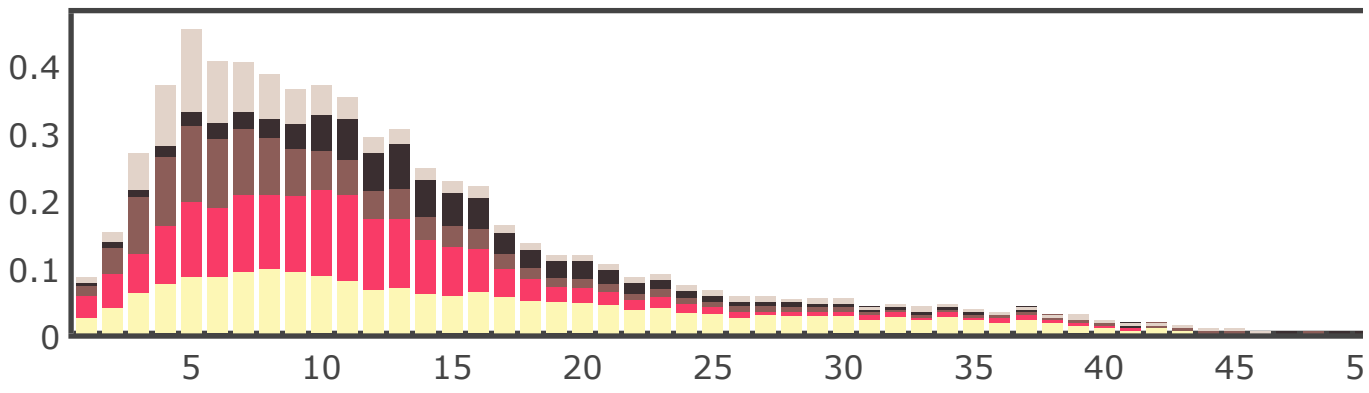
Lamprogrammus exutus (Legless cusk eel)



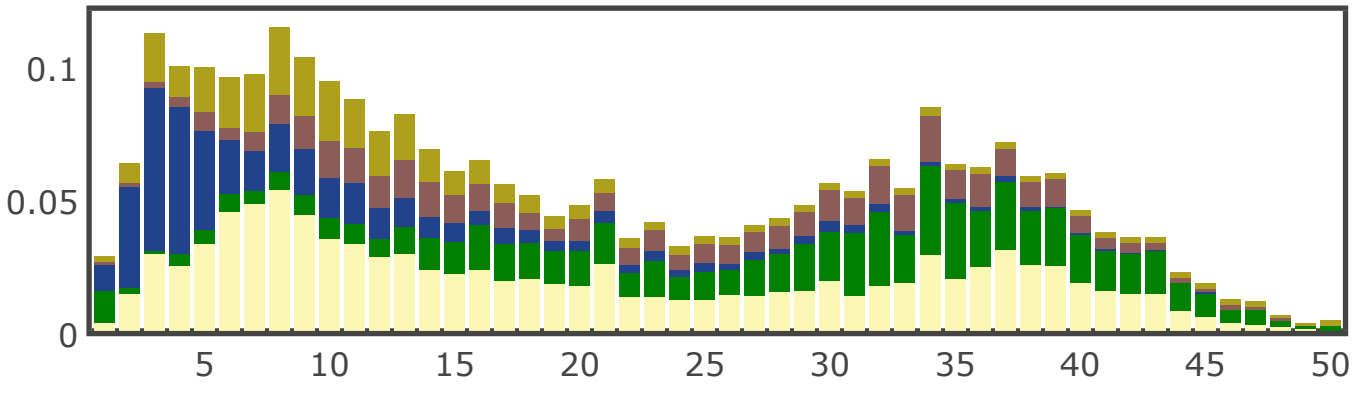
Neoniphon sammara (Sammara squirrelfish)



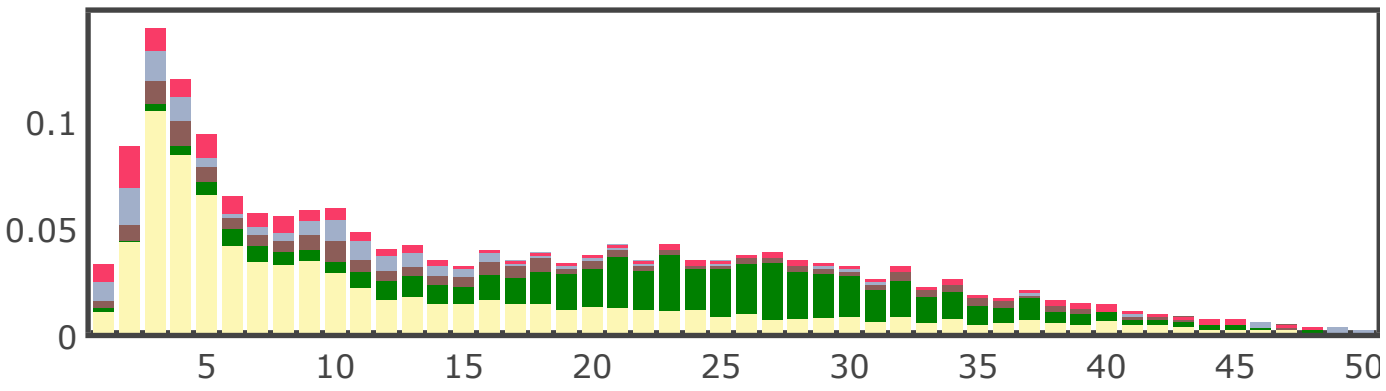
Lesueurigobius cf. sanzoi (Sanzo's goby)



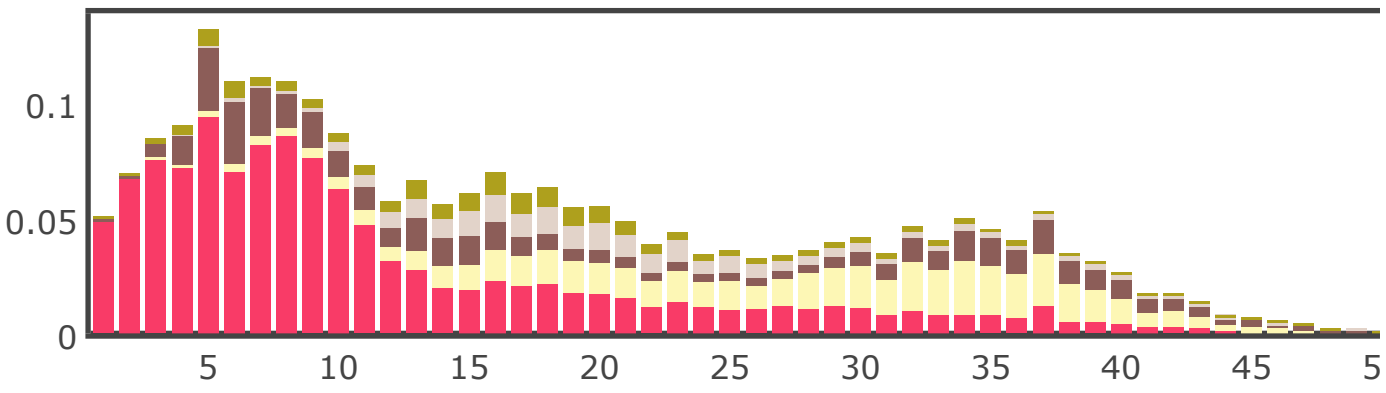
Merluccius merluccius (Hake)



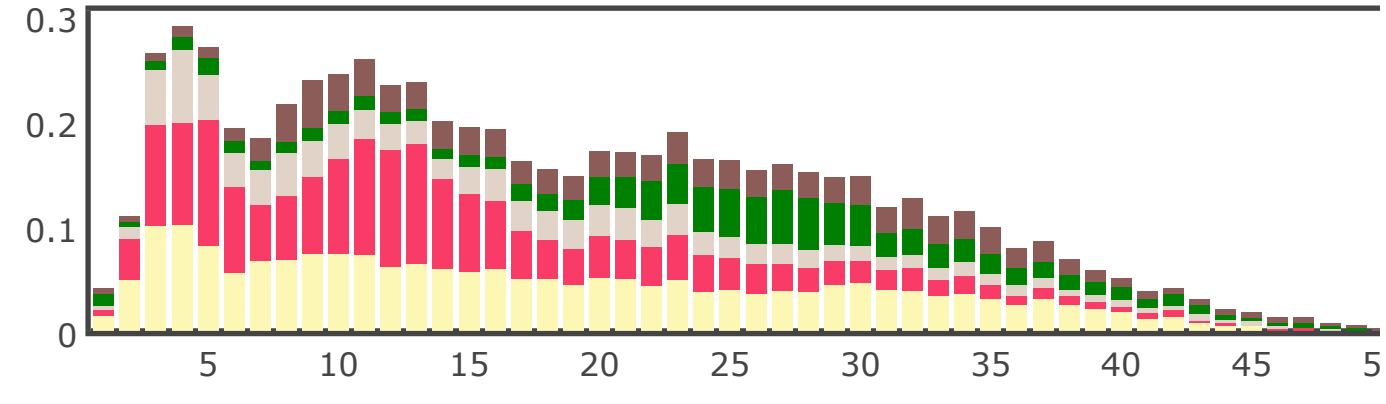
Mora moro (Common mora)



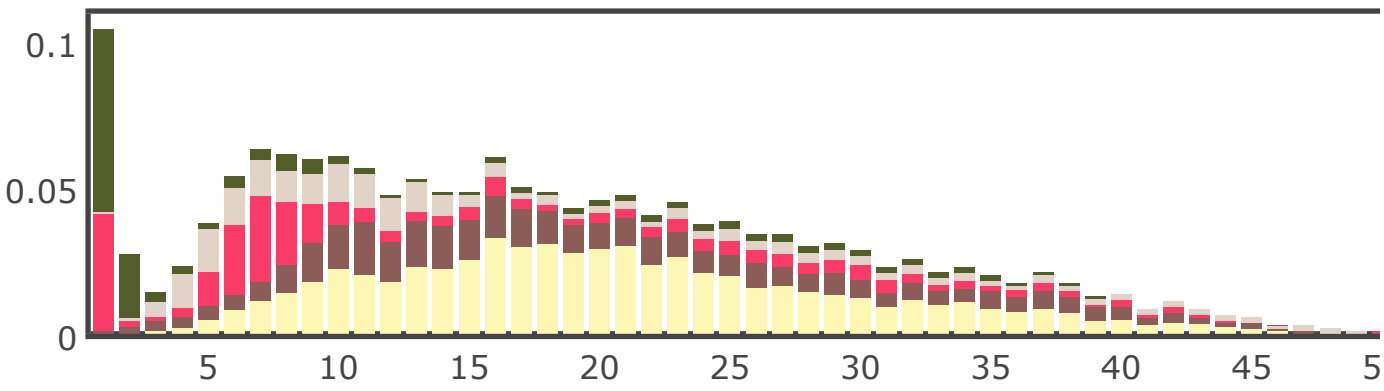
Myripristis jacobus (Blackbar soldierfish)



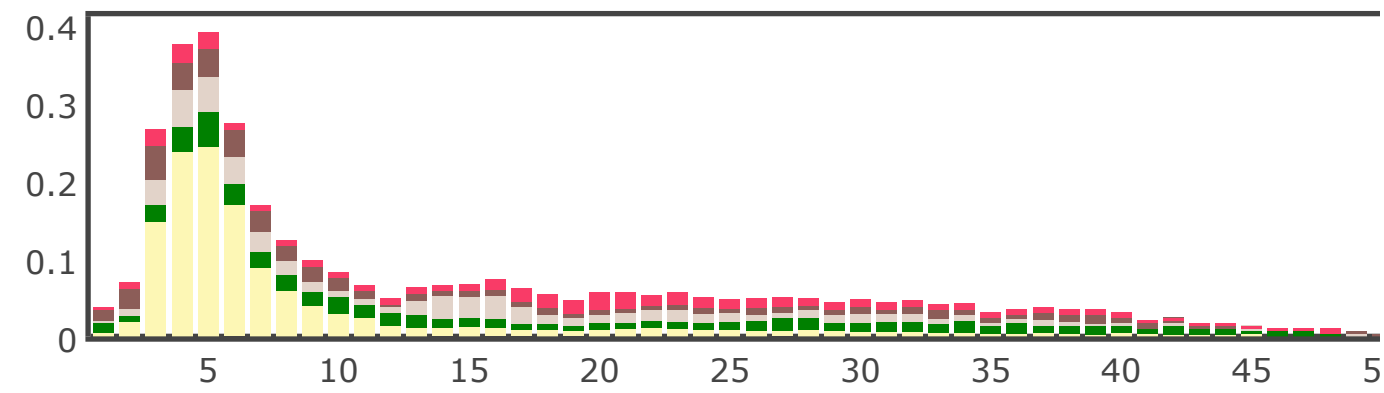
Oryzias latipes (Medaka)



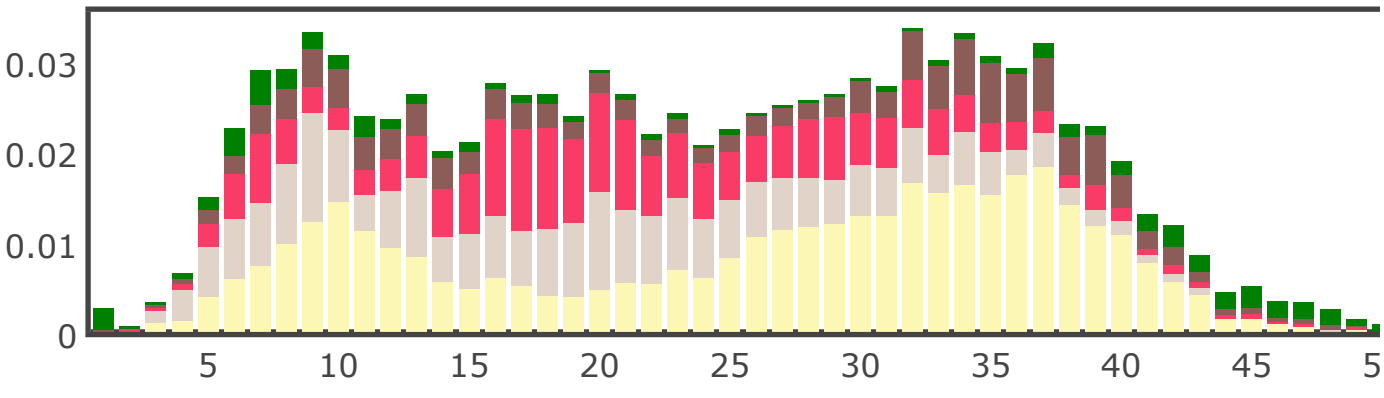
Symphodus melops (Corkwing wrasse)



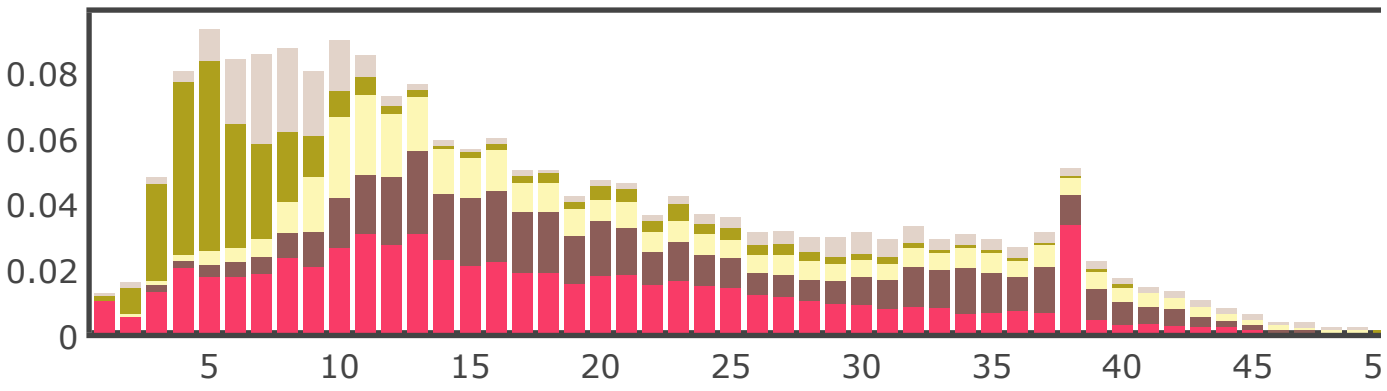
Takifugu rubripes (Japanese puffer)



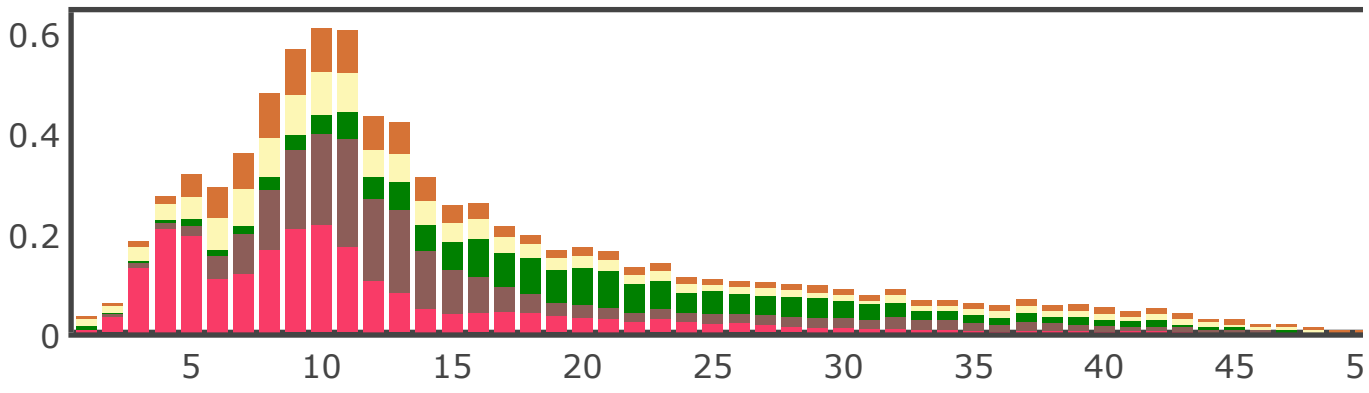
Holocentrus rufus (Longspine squirrelfish)



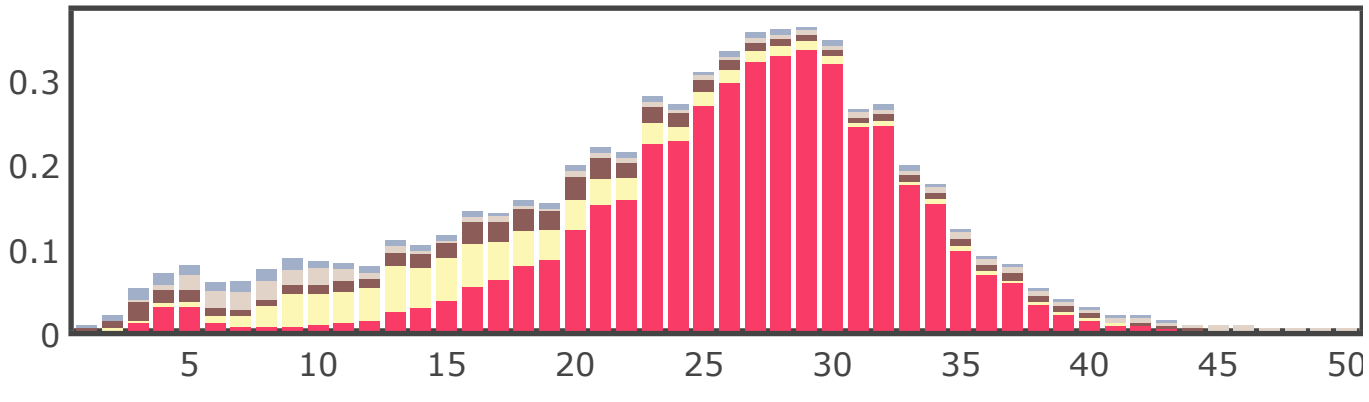
Spondyliosoma cantharus (Black seabream)



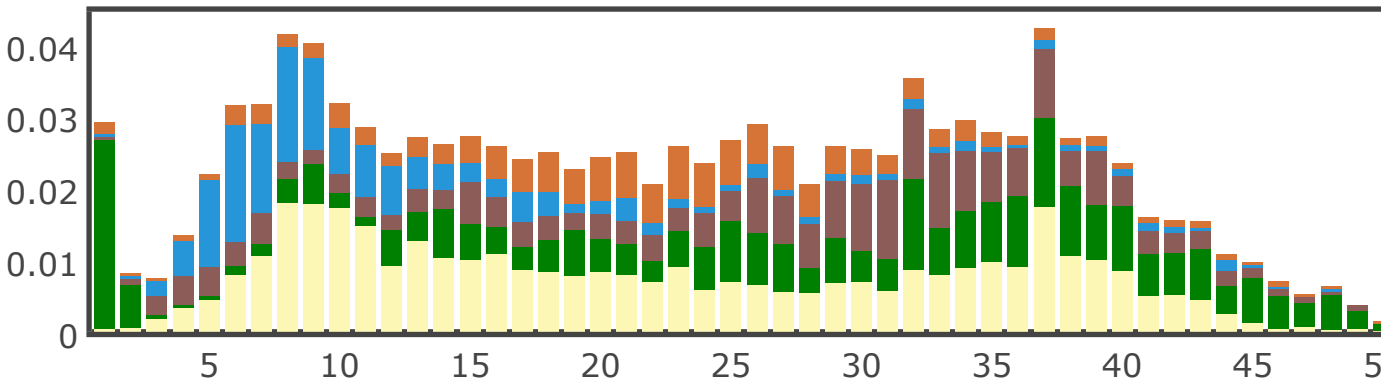
Zeus faber (John dory)



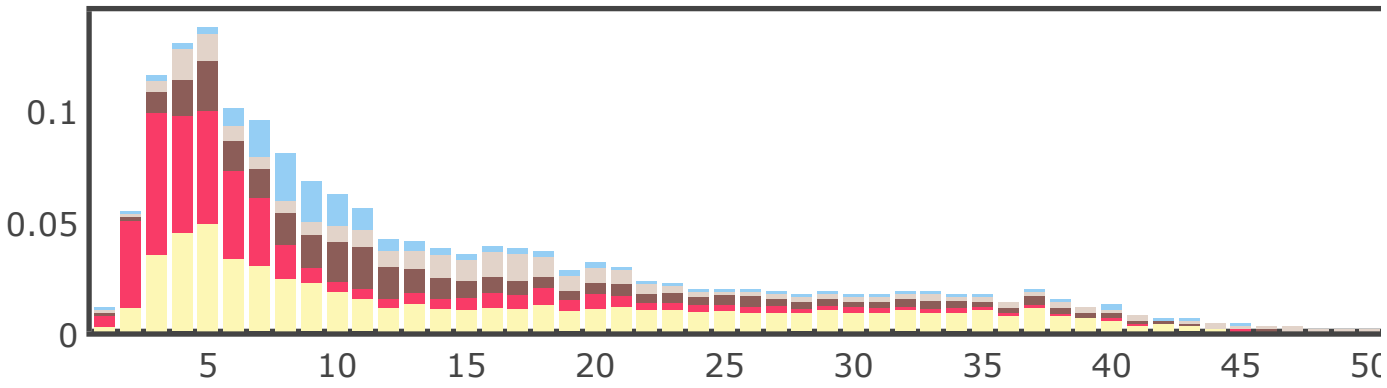
Xiphophorus maculatus (Southern platyfish)



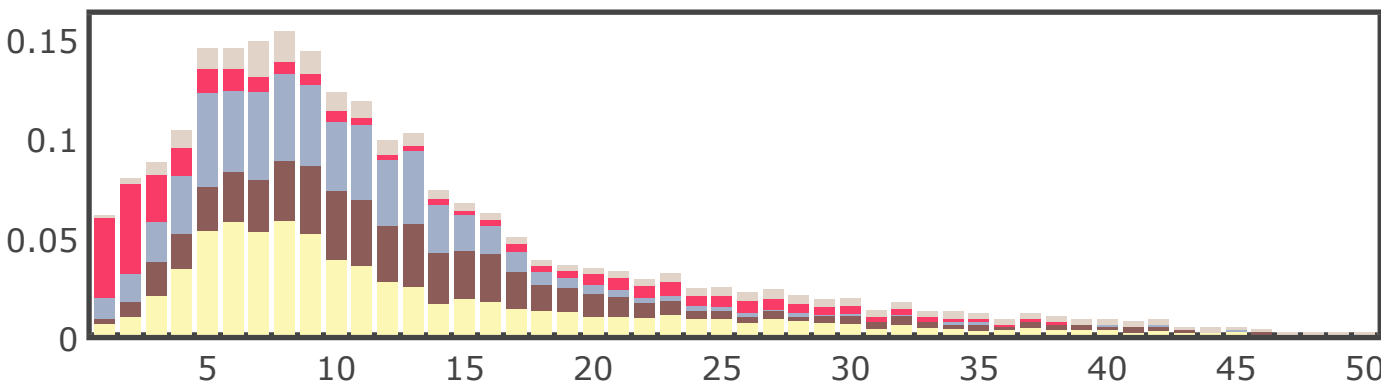
Brosme brosme (Cusk)



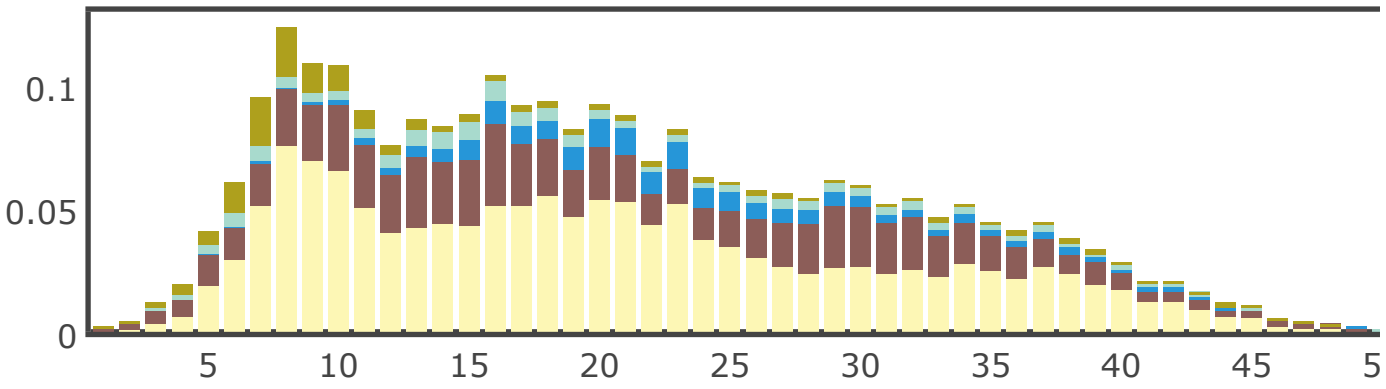
Sebastes norvegicus (Golden redfish)



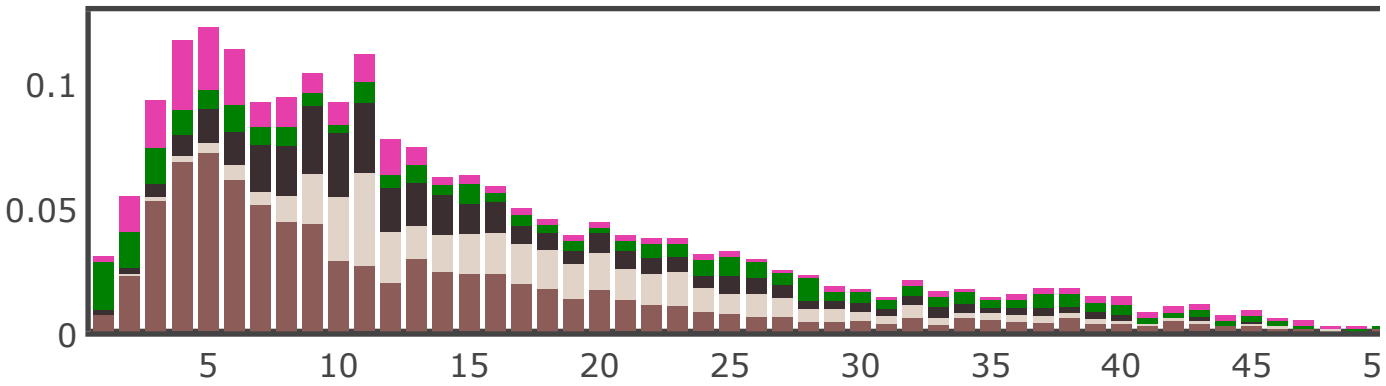
Myoxocephalus scorpius (Shorthorn sculpin)



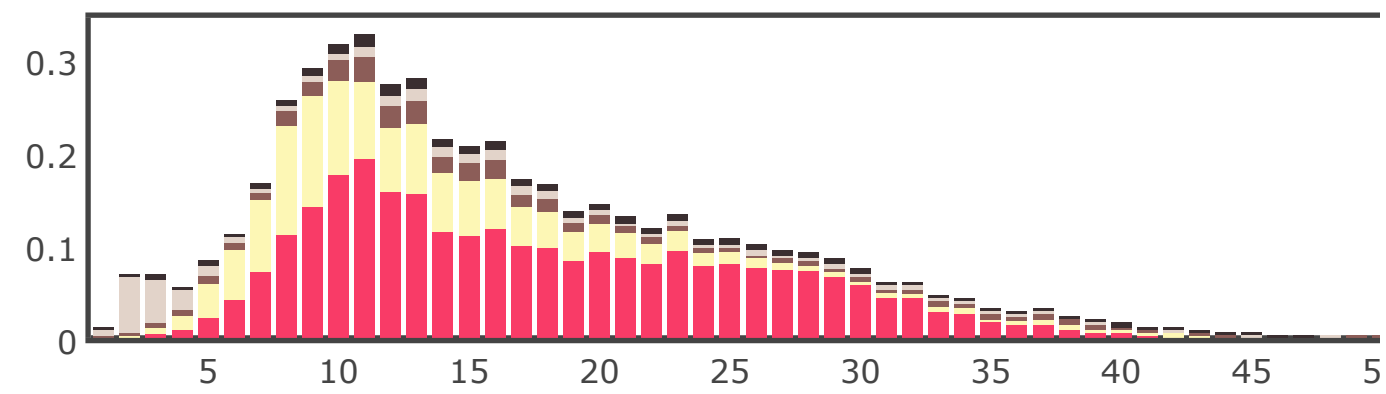
Parasudis fraserbrunneri (Tripodfish)



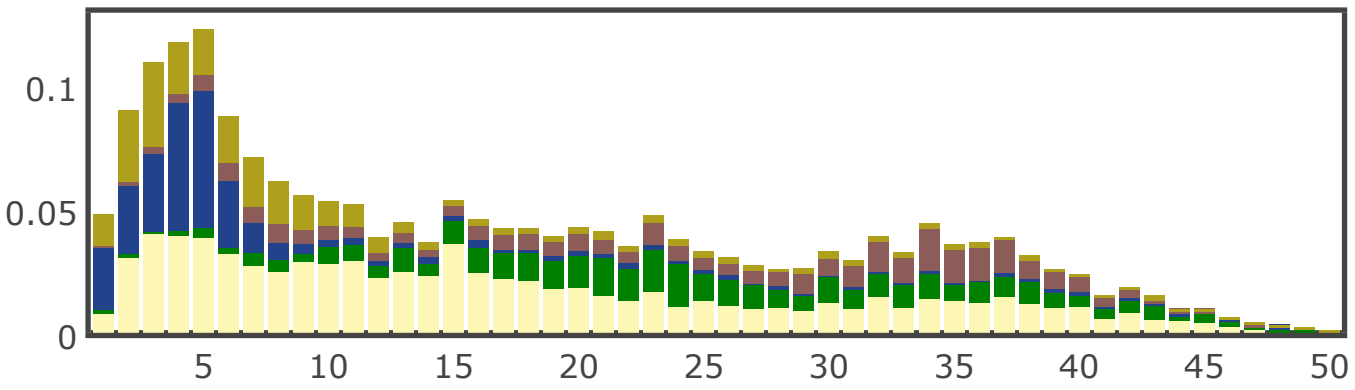
Tetraodon nigroviridis (Green spotted puffer)



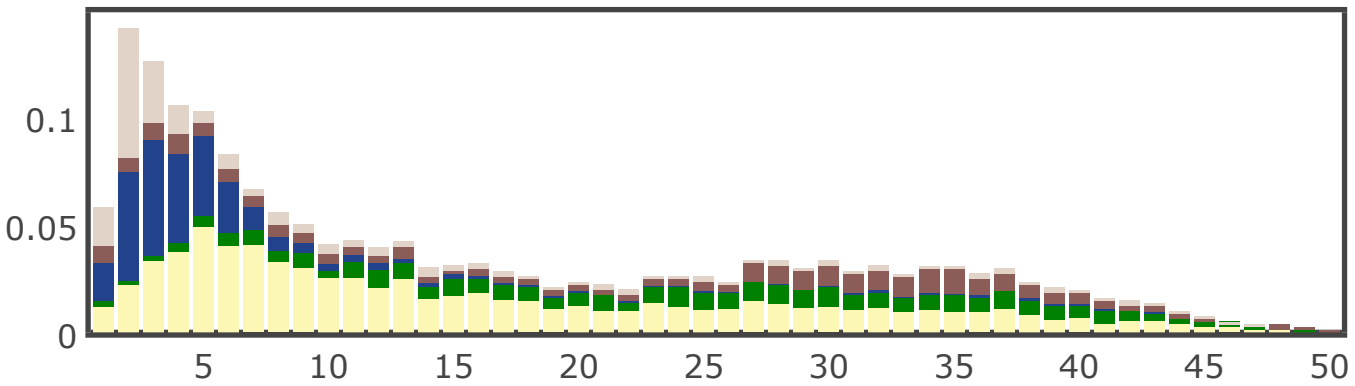
Anabas testudineus (Climbing perch)



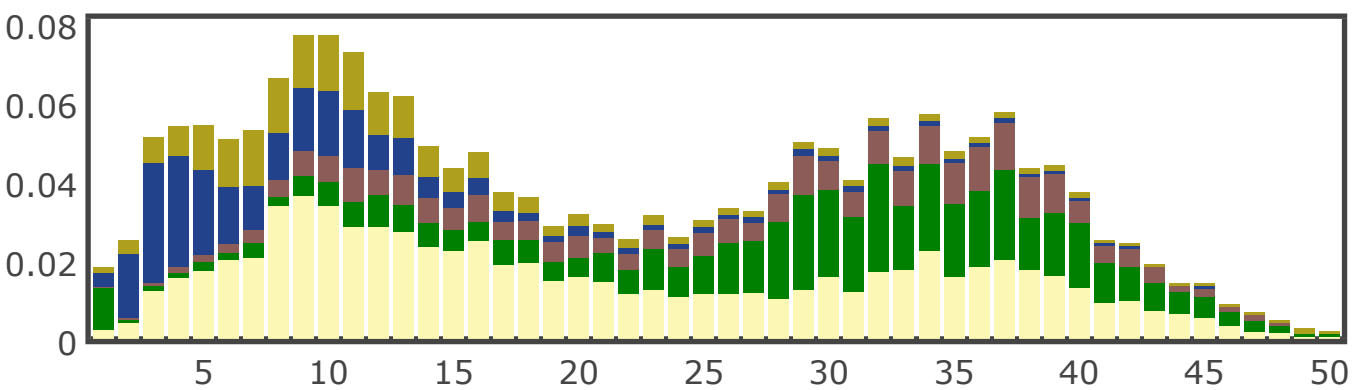
Theragra chalcogramma (Alaska pollock)



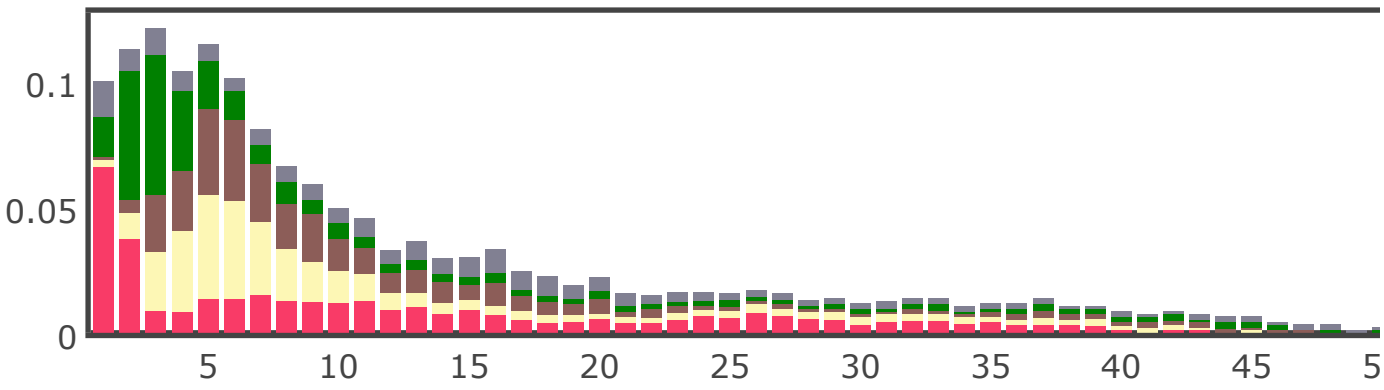
Boreogadus saida (Polar cod)



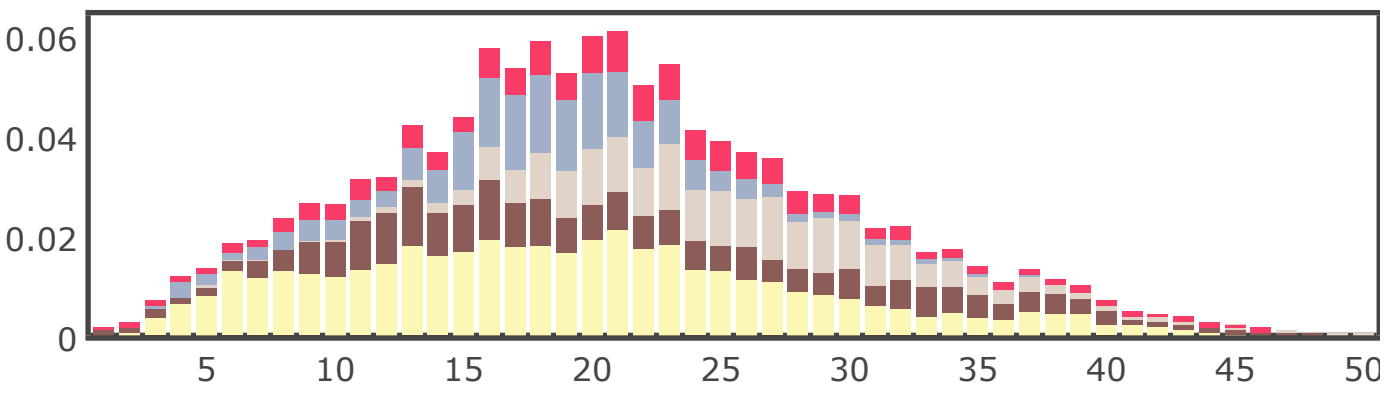
Merluccius capensis (Shallow-water Cape hake)



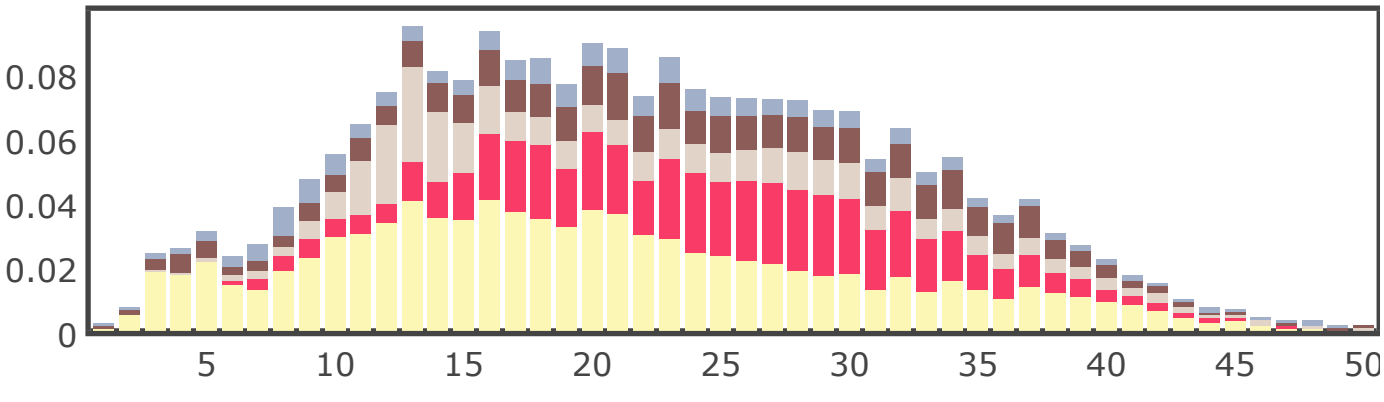
Osmerus eperlanus (European smelt)



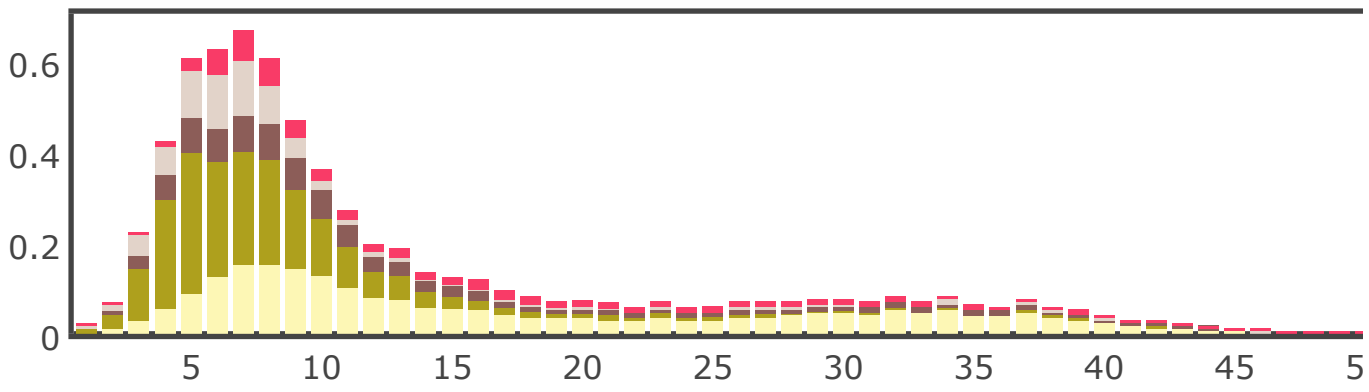
Brotula barbata (Bearded brotula)



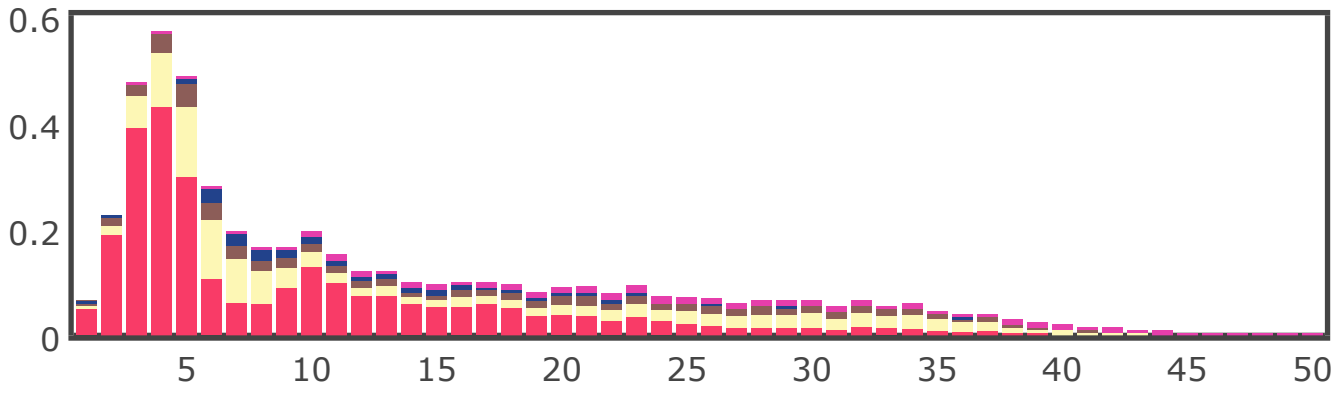
Parablennius parvicornis (Rock-pool blenny)



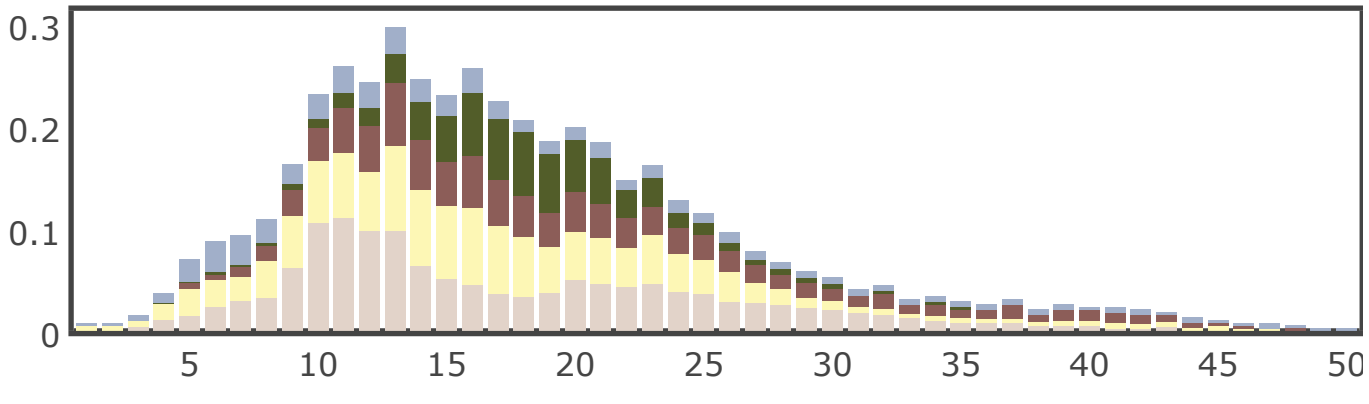
Acanthochaenus luetkenii (Pricklefish)



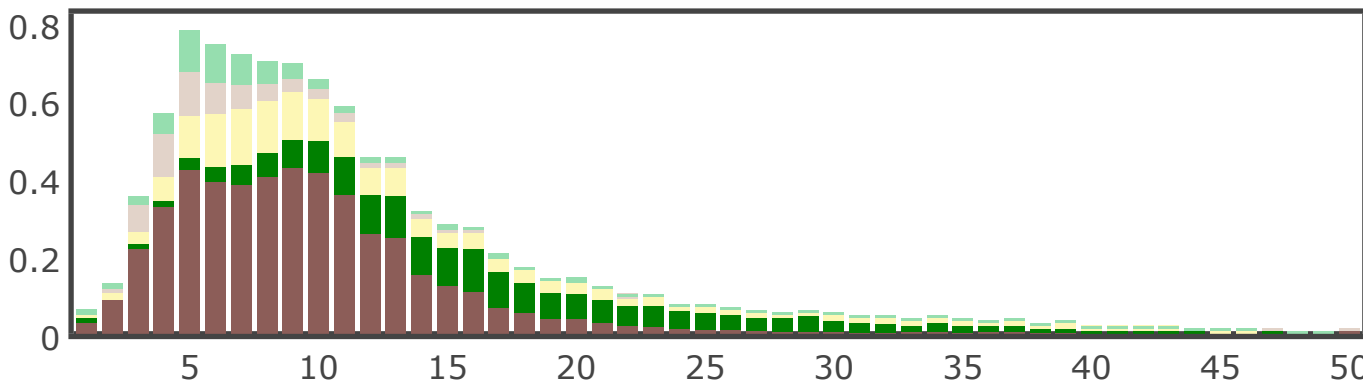
Monocentris japonica (Pinecone fish)



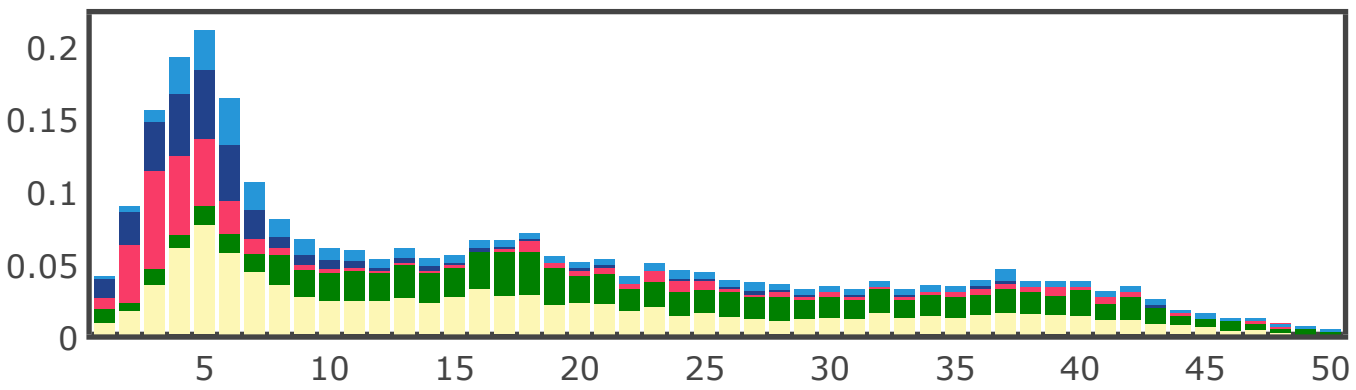
Carapus acus (Pearl fish)



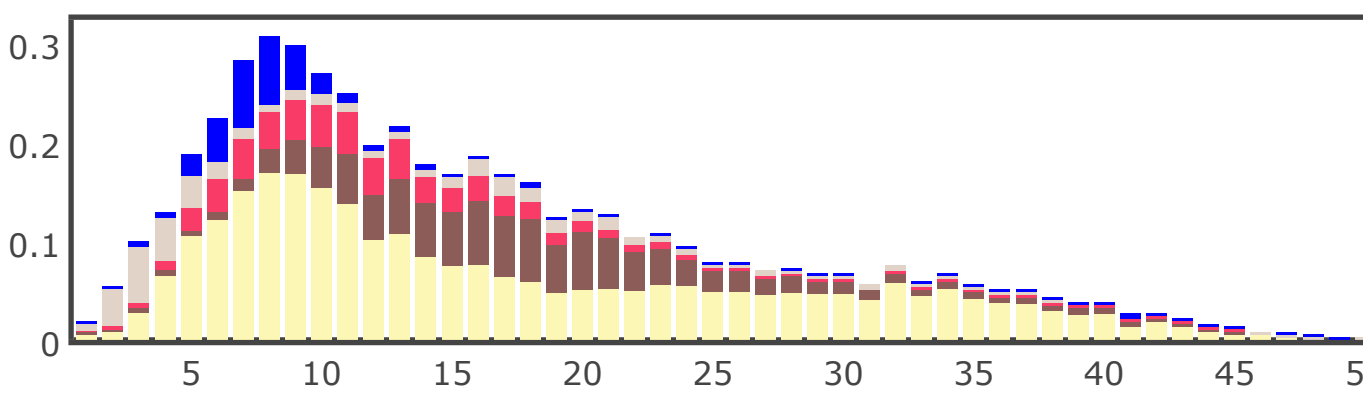
Muraenolepis marmoratus (Marbled moray cod)

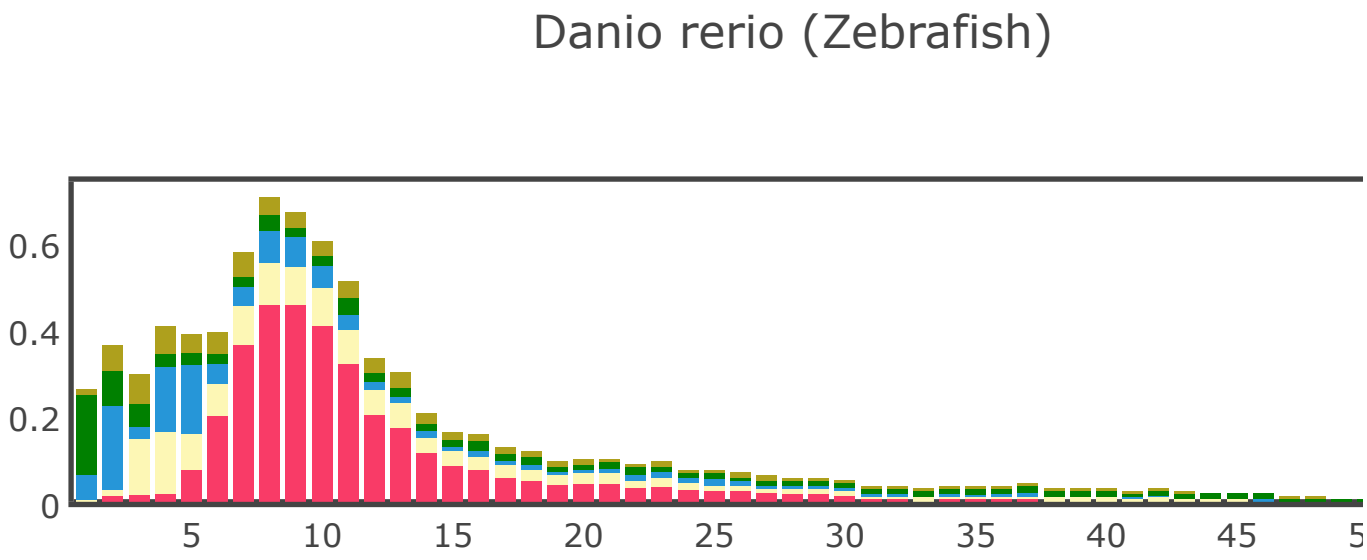


Merlangius merlangus (Whiting)



Borostomias antarcticus (Snaggletooth)





Connecting each landscape to the phylogenetic tree

In [19]:

```
from ete3 import Tree, faces, TreeStyle, TextFace, NodeStyle

def make_a_cool_figure(tree, order, filename):
    # Importing full phylogeny
    t = Tree(tree)

    # Deleting missing data that I do not have a RepeatLandscape for.

    fish_94 = t.search_nodes(name="fish_94")[0]
    fish_94.delete()
    fish_107 = t.search_nodes(name="fish_107")[0]
    fish_107.delete()

    """# Include only this order:

    exclude = set(fam_plot[fam_plot['ORDER']!=order].ALIAS)

    mylist = ['Merluccius polli',
              'Merluccius merluccius', 'Merluccius capensis']

    exclude = set(fam_plot[~fam_plot['SPECIES'].isin(mylist)].ALIAS)"""

    # For every node label, get the names
    fish = []
    for node in t.traverse():
        if 'fish' in node.name:
            fish.append(node.name)

    # For every element not being the chosen order, and if it actually is in t
he phylogeny, delete the node.
    """for element in exclude:
        if element in fish:
            delfish = t.search_nodes(name=element)[0]
```

```
            delfish.delete(preserve_branch_length = True)"""

    # Create the layout
    def mylayout(node):
        if node.is_leaf():
            for i in fish:
                if node.search_nodes(name = i):
                # Add the repeatlandscape
                    node.img_style["size"] = 1
                    faces.add_face_to_node(faces.ImgFace('../figures/RLfigures
/%s.png' % i),
                                                node, column=0, aligned=True)
                    #if i in set(fam_plot.ALIAS):
                    #    faces.add_face_to_node(faces.TextFace('%s' % fam_plot
[fam_plot['ALIAS']==i].SPECIES.unique()[0], fsize = 10),
                    #                           node, column=0)

    # Display the tree
    ts = TreeStyle()
    #ts.complete_branch_lines_when_necessary = True
    ts.show_leaf_name = False
    ts.show_branch_length = False
    ts.show_scale = False
    ts.layout_fn = mylayout
    #ts.scale = None
    ts.tree_width = 1000
    #ts.title.add_face(TextFace("%s" % order, fsize=76), column = 0)

    for n in t.traverse():
        nstyle = NodeStyle()
        nstyle["fgcolor"] = "Black"
        nstyle["size"] = 2
        n.set_style(nstyle)

    t.render(filename, tree_style = ts)
    return "Done"
```

In [20]:

```
make_a_cool_figure('../phylogeny/final_tree.tre', '', '../figures/OrderScapes/
%s_tree.pdf' % '3.may_test_RL_tree')
```

Out[20]:

```
'Done'
```