NIviz

Chloé Nater, Anders Kolstad, Matthew Grainger, Markus Fjelstad Israelsen

2022-11-30

Contents

Chapter 1

Introduction

As part of the Nature Index' "vedlikeholdsavtale" for 2022, we organized a workshop for mapping out potential improvements for the graphical presentation of the Nature Index on its website (www.naturindeks.no) and developing suggestions and prototypes for new plots, figures, and maps. The following documentation constitutes a report of the work that has been done under this workshop. The report is set up as a bookdown document. This means that at the core it consists of Rmarkdown files (.Rmd) that contain written text, raw R code, and figures. Rmarkdown files can be rendered into a variety of formats including HTML and PDF. The former is deployed to a webpage: (https://ninanor.github.io/NIviz/index.html) and provides the best reading experience. The PDF version is suitable for archiving, but the formatting – in particular of code snippets – can be suboptimal in places. The work was conducted in an open manner, and all code that was developed is publicly available in a dedicated repository on GitHub: https://github.com/NINAnor/NIviz.

1.1 Today's presentation and need for improvements

The website www.naturindeks.no presents the results from the most recent update of the Nature Index. Specifically, it visualizes the calculated values for all individual indicators, for the Nature Index for each ecosystem type, and for a variety of thematic indices as both area-weighted time-series (Figure 1.1) and as maps (Figure 1.2). Additionally, other key information about indicators, such as the importance of disturbance factors and the relative amounts of data from different sources, are displayed as plain text and/or tables (they can be found on the top-right of the indicator pages on www.naturindeks.no, Figure 1.3).

1.1.1 Time series plots

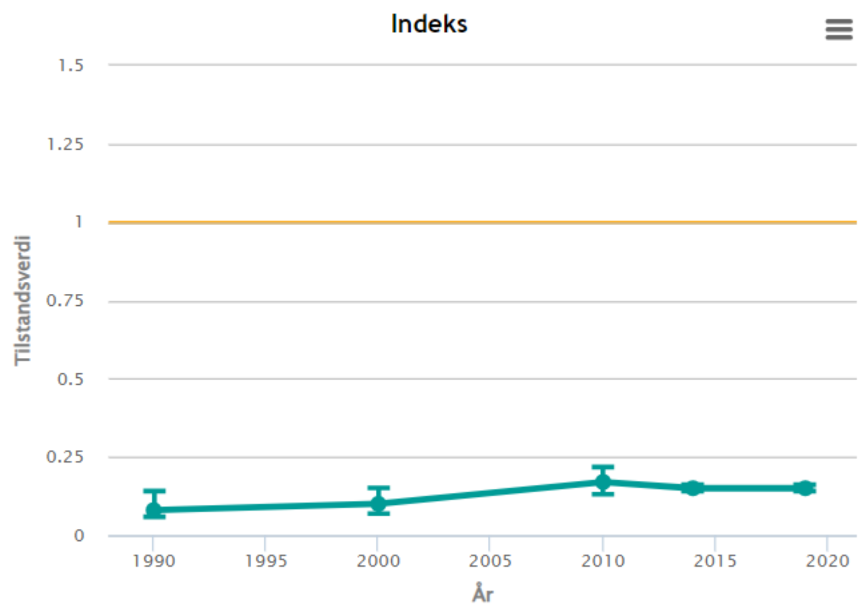


Figure 1.1: Area-weighted time-series for wolverine indicator as presented in today's version of the Nature Index webpage.

Today's time series plots show the area-weighted indicator and index values, including uncertainty, over time. The representation is well suited for displaying index change over time, but several improvements could be made when used for single indicators:

1. **Clarifying definition:** today's graphs show the area-weighted averages of indicator values. This means that they do NOT display "how well populations of an indicator species are doing within Norway" but "how well areas within Norway combined are doing with respect to an indicator species". This definition is not intuitive and not clearly stated as of now, which makes it easy for people to misunderstand what they are looking at.

2. **Displaying raw data:** only the scaled indicator values are shown, but raw data (= un-scaled population estimates stored in the Nature Index database and underlying indicator calculation) is not visible. Omitting raw data is not only in disagreement with principles of transparency, but also a lost opportunity in several ways. Displaying raw data next to scaled indicator data may help people to better understand both, and also show-

case what kind of data (including its spatial and temporal resolution) is available and forms the backbone of the indicator calculations.

3. **Toggle option for different areas:** for both scaled indicator values and underlying raw data, it could be informative to be able to look at time-series from different areas and compare them. That would, for example, be able to show that some species may have declining and increasing trends in different areas, or that trends in population changes may differ in strength across space.

4. **Review criteria for NOT including time-series plots:** for many indicators, the time-series plots are not displayed on the website at all. Instead, one is greeted with a paragraphs outlining potential scenarios in which the graphs may not be shown. The scenarios include: data only available for a small area, large proportion of missing data, data showing opposing trends in different areas, high uncertainty in trends estimates, and indicator values that have been very good for the entire time series. This has a scent of cherry-picking, and can not be considered transparent or good practice. In many cases, finding appropriate ways to present results and providing good explanations of graphs should be preferable to omitting information and this should be revisited befor publishing NI2025.

1.1.2 Maps

Today's maps show the average indicator / index value per area within Norway. The maps have some basic interactive elements: 1) hovering over an area will highlight it and display a pop-up box that shows the area name and exact average indicator values and 2) the sliding bar at the bottom allows to toggle between years.

The main drawback with the current map solution is that it does not display uncertainty in the indicator / index values at all. Uncertainty visualization is a very active field of research at the moment, not in the least because there is an increasing focus in management and politics on maps of (= spatially explicit information on) biodiversity, ecosystem condition, ecosystem services, etc. Innovative approaches to visualizing uncertainty in maps are consequently popping up, and it is worth investigating the potential use of at least some of these here. Additionally, there are some smaller quality of life improvements that could be made to the current maps. For example, having the option to zoom in would greatly help with visualization of indicators that have data provided at fine spatial scales (e.g. municipality-level: https://www.naturindeks.no/Indicators/dikesoldogg).

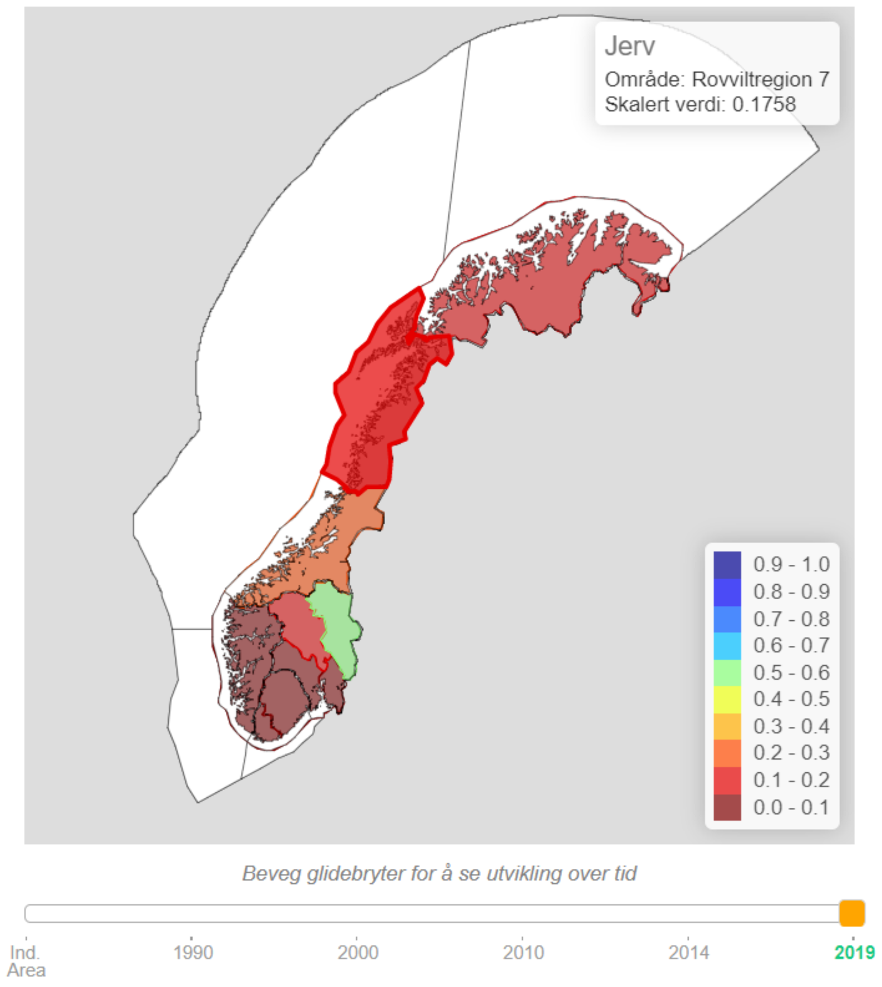
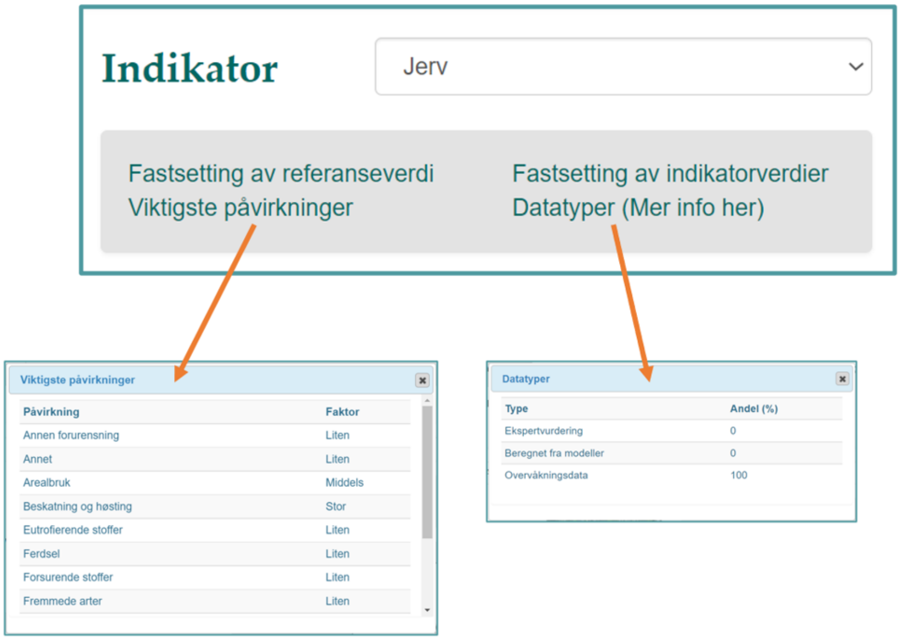


Figure 1.2: Map of wolverine indicator as presented in today's version of the Nature Index webpage. Note that when hovering over one area, it highlights and a pop-up box appears on the top right displaying the area name and exact average indicator value. The sliding bar on the bottom allows to toggle between years.


Indikator
Jerv
Fastsetting av referanseverdi
Viktigste påvirkninger
Fastsetting av indikatorverdier
Datatyper (Mer info her)

Viktigste påvirkninger

Påvirkning	Faktor
Annen forurensning	Liten
Annet	Liten
Arealbruk	Middels
Beskatning og høsting	Stor
Eutrofierende stoffer	Liten
Ferdsel	Liten
Forsurende stoffer	Liten
Fremmede arter	Liten

Datatyper

Type	Andel (%)
Ekspertvurdering	0
Beregnet fra modeller	0
Overvåkningsdata	100

Figure 1.3: Display of additional key information on the indicator pages on the website (top right corner).

1.1.3 Other key information

There's a range of other key information tied to each indicator and index. Some of it is presented on today's website but could be presented better, and some is not presented on the website at all but maybe should be.

In the top-right corner of the pages of all indicators is a box with four hyperlinks to different pieces of key information. The first notable thing here is that these links look as if they will take one away from the page when clicking, and are therefore not very inviting to click on. Using a different presentation that is more in line with modern web design could remedy this.

If one does dare to click on one of the links, a small box with the information will pop up. While text seems appropriate to describe how indicator and reference values are set up, the tables presenting the relative importance of disturbance factors and percentage of data from different sources are bland and not engaging in the least. There are many alternative ways of presenting such information in more inviting ways.

The same approaches would also work well for displaying some bits of key information that are not included on the webpage yet such as an indicator's ecosystem fidelity (i.e. which ecosystem it belongs/contributes to), the proportion of missing/imputed datapoints in the time-series, and various aspects of weighting (e.g. weights of different indicators in one index, weights of one indicator in different indices, area weights within an indicator, etc.)

1.2 About the suggestions and prototypes developed in this pilot project

The remainder of this documentation contains code for generating and presentation of some preliminary graphs and figures that could potentially address the short-comings of today's presentation outlined above. They should be regarded as drafts or prototypes; they are by no means ready products and implementation would require some further development, including optimisation, proper styling according to a common theme, and – in some cases – adaptation of the R code into the source code for the website. We also outline some additional ideas that we were considering, but did not have time to investigate further at this stage.

The majority of graphs were developed for raw data and/or scaled values of a selection of single-species indicators. The indicators used in the pilot study were selected to represent both terrestrial and aquatic habitats, and to cover different spatial resolutions and degrees of uncertainty:

- Oblong-leaved sundew ("Dikesoldogg")

- Wolverine (“Jerv”)
- Moose (“Elg)
- Common guillemot (“Lomvi”)
- White-tailed eagle (“Havørn”)
- Common ling (“Lange”)

Nonetheless, most of the graphs developed for indicator data are perfectly applicable also to the presentation of composite indices (Nature Index, various thematic indices, etc.). Similarly, the graphical tools suggested here are just as relevant to the “Fagsystem for økologisk tilstand” as they are to the Nature Index.

Both the code and documentation for this pilot project are collaborative work and were written by different people (Chloé Nater, Anders Kolstad, Matthew Grainger, Markus Fjelstad Israelsen). For that reason, writing style, coding style, and relative amounts of full-text explanations are variable throughout this document. We are happy to provide additional information on any section on request.

Chapter 2

Downloading and preparing example data

This section describes how to download the relevant indicator data from the Nature Index Database. It is completely reproducible, and can be run by anyone with valid login credentials to the database. The downloaded data is then stored locally for use in the remainder of the code.

```
if(!require(NIcalc)){
  devtools::install_github("NINAnor/NIcalc", build_vignettes = F)
}
library(NIcalc)
```

First, you have to fill in your username (NINA email) and password.

```
myUser <- "user@nina.no" # insert NINA email
myPwd  <- "" # secret password
```

Then, choose which indicator(s) you want, use the NIcalc "importDatasetApi" function to retrieve data from the database, and save the dataset locally.

```
indicator <- c("Dikesoldogg",
               "Jerv",
               "Elg",
               "Lomvi",
               "Havørn",
               "Lange")
```

```
for(i in indicator){
indicatorImport <- NULL
indicatorImport <- NIcalc::importDatasetApi(
  username = myUser,
  password = myPwd,
  indic = i,
  year = c("1990","2000","2010","2014","2019"))

assign(paste0(i, "_import"), indicatorImport)
}
```

```
path <- "P:/41201612_naturindeks_2021_2023_database_og_innsynslosning/temp/"

for(i in indicator){
  temp <- get(paste0(i, "_import"))
  saveRDS(temp, paste0(path, i, "_import.rds"))
}

for(i in indicator){
  temp <- paste0(path, i, "_import.rds")
  assign(i, readRDS(temp))
}
```

Next, we need to assemble the data set. This shouldn't be necessary since all the data is already present. But one thing we noticed was that for some indicators (e.g. jerv), the distribution family and parameters only appear after assembling.

```
# Spesify all of Norway incl the five regions, som NIunits:
myNIunits <- c(allArea = T, parts = T, counties = F)
# Include all BSunits (kommuner) irrespective of the proportion of the main ecosystems
myPartOfTotal <- 0

for(i in indicator){

  temp <- get(paste0(i, "_import"))
  assemeble <- NULL
  assemeble <- NIcalc::assembleNiObject(
    inputData = temp,
    predefNIunits = myNIunits,
    partOfTotal = myPartOfTotal,
    indexType = "thematic",
    part = "ecosystem",
    total = "total")

  # I dont se the output changing if I for example chose total = marine. Perhaps 'part
  assign(paste0(i, "_assemble"), assemeble)

}
```

We then save the assembled data files.

```
for(i in indicator){
  temp <- get(paste0(i, "_assemble"))
  saveRDS(temp, paste0("data/", i, "_assemebled.rds"))
}
```

If necessary, we load the datafiles back into R.

```
for(i in indicator){
  temp <- paste0("data/", i, "_assemebled.rds")
  assign(i, readRDS(temp))
}
```

Since we are particularly concerned with visualizing uncertainty alongside means, we next need to simulate raw data and indicator values from the uncertainty distributions stored in the database.

```
myYears <- as.character(c(1990,2000,2010,2014,2019))


for(j in indicator){
print(j)

  temp <- get(j)
  temp2 <- get(paste0(j, "_import"))
  temp_comb <- data.frame(NULL)
  myMat2 <- NULL
  myMat2_comb <- NULL
  obstype <- NULL


  obstype <- temp$referenceValues$distributionFamilyName
  obstype[!is.na(obstype)] <- "tradObs"
  obstype[is.na(obstype)]  <- "customObs"

myMatr <- NIcalc::sampleObsMat(
  ICunitId           = temp$referenceValues$ICunitId,
  value              = temp$referenceValues$expectedValue,
  distrib            = temp$referenceValues$distributionFamilyName,
  mu                 = temp$referenceValues$distParameter1,
  sig                = temp$referenceValues$distParameter2,
  customDistribution = temp$referenceValues$customDistribution,
  obsType            = obstype,
  nsim =1000
        )

myMatr <- as.data.frame(myMatr)
myMatr <- myMatr %>%
  tibble::add_column(.before=1,
    ICunitID = row.names(myMatr))

myMatr <- myMatr %>%
  tibble::add_column(.after = 1,
      year = NA)

for(i in 1:length(myYears)){
print(i)

obs <- NULL
  obs <- temp$indicatorValues[[i]]$distributionFamilyName
  obs[!is.na(obs)] <- "tradObs"
  obs[is.na(obs)]  <- "customObs"


myMat <- NIcalc::sampleObsMat(
  ICunitId           = temp$indicatorValues[[i]]$ICunitId,
  value              = temp$indicatorValues[[i]]$expectedValue,
  distrib            = temp$indicatorValues[[i]]$distributionFamilyName,
  mu                 = temp$indicatorValues[[i]]$distParameter1,
  sig                = temp$indicatorValues[[i]]$distParameter2,
  customDistribution = temp$indicatorValues[[i]]$customDistribution,
  obsType            = obs,
  nsim               = 1000

)


myMat2 <- as.data.frame(myMat)

myMat2 <- myMat2 %>%
  tibble::add_column(.before=1,
    ICunitID = row.names(myMat))
```

```
myMat2 <- myMat2 %>%
  tibble::add_column(.after = 1,
    year = myYears[i])

myMat2_comb <- rbind(myMat2_comb, myMat2)

 }

comb <- rbind(myMatr, myMat2_comb)

comb <- comb %>%
  tibble::add_column(.after = 1,
    ICunitName = temp2$ICunits$name[match(
      comb$ICunitID, temp2$ICunits$id)])

comb2 <- comb[!is.na(comb$year),]
comb3 <- comb[is.na(comb$year),]
comb3$ref_mean <- rowMeans(comb3[,-c(1:3)])

combScaled <- comb2 %>%
  tidyr::pivot_longer(cols = starts_with("V"))

combScaled <- combScaled %>%
  tibble::add_column(ref = comb3$ref_mean[
    match(combScaled$ICunitID, comb3$ICunitID)])
combScaled$scaledIndicator <- combScaled$value/combScaled$ref
combScaled <- dplyr::select(combScaled,
                     -name,
                     -value,
                     -ref)

assign(paste0(j, "_bootstrapped_raw"), comb)
assign(paste0(j, "_bootstrapped_scaled"), combScaled)

}
```

Note that in this case, we also bootstrapped the reference values. These are coded as year = NA. This means that the simulation also accounts for reported uncertainty in reference values. For the purpose of this work, this makes sense as it leads to somewhat higher uncertainty in indicator values and hence a dataset that is more suitable to for testing out visualizations of uncertainty. However, up to and including the last iteration of the Nature Index (NI2020), uncertainty in reference values was NOT taken into account.

Finally, we save the simulated datasets.

```
for(i in indicator){
  temp <- get(paste0(i, "_bootstrapped_raw"))
    temp <- get(paste0(i, "_bootstrapped_scaled"))

  saveRDS(temp, paste0("data/", i, "_bootstrapped_raw.rds"))
    saveRDS(temp, paste0("data/", i, "_bootstrapped_scaled.rds"))

}
```

Chapter 3

Time series

This chapter presents prototypes for upgraded representations of time-series information, both raw data and scaled indicator values.

We had conceptually outlined two major ideas here (Figure 3.1):

1) Graphs with toggle options allowing to display data and indicator values with different levels of aggregation, and for the different areas they are reported for in the database
2) Side-by-side presentation of scaled indicator data and raw data

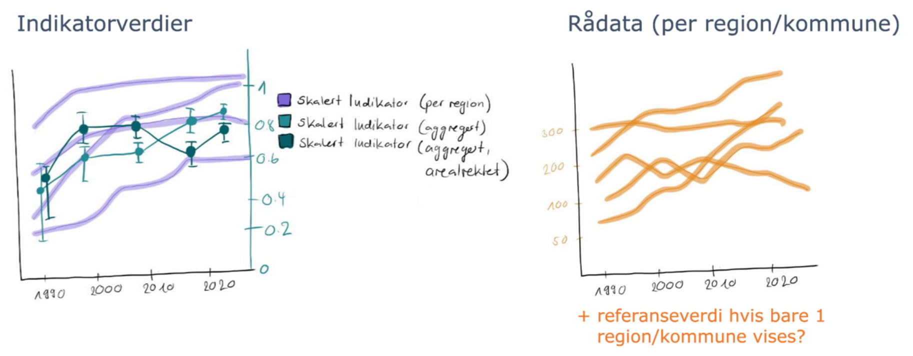
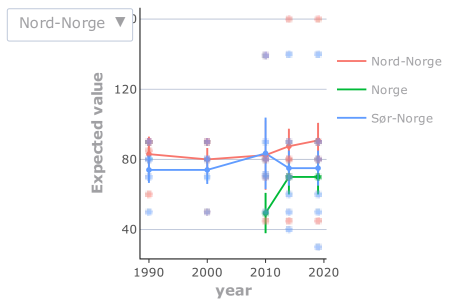


Figure 3.1: Early sketch of ideas for improving / adding additional information to time-series plots.

Time was not sufficient to implement all of these aspects in great detail and some are still missing in the presentation below (e.g. the overlay of averaged indicator values over area-specific indicator values).

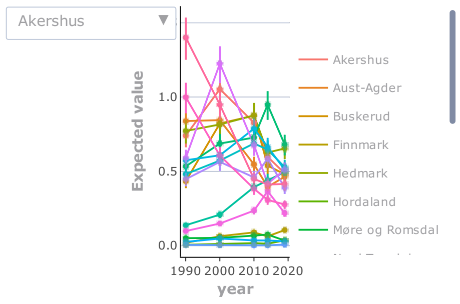
3.1 Raw data

```
library(plotly)
## TODO change to correct data source
passerinesImport <- readRDS("data/passerinesImport.rds")

dat=passerinesImport$indicatorObservations$indicatorValues

dat$yearName=as.numeric(dat$yearName) # convert character vector to numeric years
sum_dat=dat %>%
  group_by(ICunitName, yearName) %>%
  summarise(mnExpected=mean(expectedValue, na.rm=TRUE),
            mnUpper=mean(upperQuantile, na.rm=TRUE),
            mnLower=mean(lowerQuantile, na.rm=TRUE)) # summerise the data to mean valu

p=sum_dat %>%
  ggplot(aes(as.numeric(yearName), mnExpected, col=ICunitName))+
  geom_line()+
  geom_pointrange(aes(x=as.numeric(yearName), y=mnExpected, ymin=mnLower, ymax=mnUpper)
  geom_point(data=dat, aes(as.numeric(yearName), expectedValue, alpha=0.2))+
  labs(x="year", y="Expected value")+
  theme_NIseries()
p2=ggplotly(p)
p2 %>% layout(
  updatemenus = list(
    list(
      type = "list",
      label = 'Category',
      buttons = list(
        list(method = "restyle",
             args = list('visible', c(TRUE, FALSE, FALSE)),
             label = "Nord-Norge"),
        list(method = "restyle",
             args = list('visible', c(FALSE, TRUE, FALSE)),
             label = "Norge"),
        list(method = "restyle",
             args = list('visible', c(FALSE, FALSE, TRUE)),
             label = "Sør-Norge")
      )
    )
  )
) # Add drop down menus for the data
```



3.2 Scaled data

```r
Elg_assemebled<- readRDS("data/Elg_assemebled.rds")
mycols=c("ICunitName" ,"yearName", "expectedValue","lowerQuantile", "upperQuantile")

data_list<-lapply(Elg_assemebled$indicatorValues, function(x) x%>% select(mycols))

dat<-bind_rows(data_list, .id = "column_label")

# plot
sum_dat=dat %>%
  group_by(ICunitName, yearName) %>%
  summarise(mnExpected=mean(expectedValue, na.rm=TRUE),
            mnUpper=mean(upperQuantile, na.rm=TRUE),
            mnLower=mean(lowerQuantile, na.rm=TRUE)) # summerise the data to mean values
source("R/ggplotTheme.R")
p=sum_dat %>%
  ggplot(aes(as.numeric(yearName), mnExpected, col=ICunitName))+
  geom_line()+
  geom_pointrange(aes(x=as.numeric(yearName), y=mnExpected, ymin=mnLower, ymax=mnUpper))+
  geom_point(data=dat, aes(as.numeric(yearName), expectedValue, alpha=0.2))+
  labs(x="year", y="Expected value")+
  theme_NIseries()
p2=ggplotly(p)
p2 %>% layout(
  updatemenus = list(
    list(
      type = "list",
      label = 'Category',
      buttons = list(
        list(method = "restyle",
             args = list('visible', c(TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F
             label = unique(dat$ICunitName)[2]),
        list(method = "restyle",
             args = list('visible', c(FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, F
             label = unique(dat$ICunitName)[8]),
        list(method = "restyle",
             args = list('visible', c(FALSE, FALSE,TRUE, FALSE, FALSE, FALSE, FALSE, FA
             label = unique(dat$ICunitName)[5]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,TRUE, FALSE, FALSE, FALSE, F
             label = unique(dat$ICunitName)[18]),
        list(method = "restyle",
             args = list('visible', c(FALSE, FALSE, FALSE,FALSE, TRUE, FALSE, FALSE, FA
             label = unique(dat$ICunitName)[3]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, TRUE, FALSE, F
             label = unique(dat$ICunitName)[11]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, TRUE, F
             label = unique(dat$ICunitName)[13]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE,
             label = unique(dat$ICunitName)[15]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE,
             label = unique(dat$ICunitName)[16]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE,
             label = unique(dat$ICunitName)[4]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE,
             label = unique(dat$ICunitName)[10]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE,
             label = unique(dat$ICunitName)[12]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE, FALSE, FAL
             label = unique(dat$ICunitName)[14]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE, FALSE, FAL
             label = unique(dat$ICunitName)[7]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE, FALSE, FAL
             label = unique(dat$ICunitName)[17]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE, FALSE, FAL
             label = unique(dat$ICunitName)[9]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE, FALSE, FAL
             label = unique(dat$ICunitName)[6]),
        list(method = "restyle",
             args = list('visible', c( FALSE, FALSE, FALSE,FALSE, FALSE, FALSE, FALSE, FALSE, FAL
             label = unique(dat$ICunitName)[1])
      )
    )
  )
) # Add drop down menus for the data
```



Chapter 4

Maps

This chapter presents a prototype for a map representation of both average indicator values and of uncertainty therein. Most of it is focused on a side-by-side presentation of two zoomable maps: one for average values and one for the coefficient of variation (= a measure of uncertainty).

At the end, we also mention some additional ideas that we did not implement (fully) yet but may be worth trying out.

4.1 Raw data

We have not written code yet for making raw data maps, but this would entail just small changes to the proposed solutions for maps of scaled data (below).

4.2 Scaled data

4.2.1 Jerv

4.2.1.1 Prepare NI data

The jerv (wolverine) data was downloaded using the `R/singleIndicator.R` script and the importDatasetApi() function, and subsequently the assemble-NiObject() function, so now we can simply import it.

```
jerv <- readRDS("data/Jerv_assemebled.rds")
```

This data file contains the raw data in the form of expected values for each BSunits (municipalities). But we actually want to keep the original geometries of the eight "rovviltregioner", and so we need to focus in the ICunits instead.

```
par(mar=c(9,5,1,1))
barplot(jerv$indicatorValues$`2019`$expectedValue,
        names.arg = jerv$indicatorValues$`2019`$ICunitName,
        las=2,
        ylab = "Estimated number of\nwolverine in 2019")
```

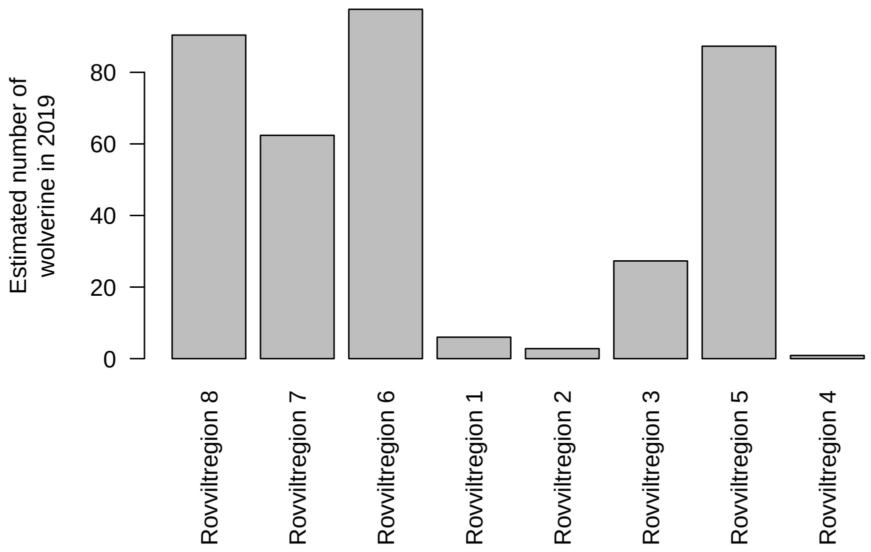


Figure 4.1: Estimated number of wolverine in 2019, exported from the NI database.

The data also contains upper and lower quantiles, but we can also get the full probability distribution and sample from it to get standard deviations, but also as probability functions that we can sample from:

```
# bruker tradOb siden custumDist er NA. Dette er ikke en generisk løsning.
obstype <- rep("tradObs", nrow(jerv$indicatorValues$'2019'))

#myYears <- as.character(c(1990,2000,2010,2014,2019))
myYears <- as.character(c(2019))

for(i in 1:length(myYears)){
```

```
# print(i)

myMat <- NIcalc::sampleObsMat(
  ICunitId           = jerv$indicatorValues[[i]]$ICunitId,
  value              = jerv$indicatorValues[[i]]$expectedValue,
  distrib            = jerv$indicatorValues[[i]]$distributionFamilyName,
  mu                 = jerv$indicatorValues[[i]]$distParameter1,
  sig                = jerv$indicatorValues[[i]]$distParameter2,
  customDistribution = jerv$indicatorValues[[i]]$customDistribution,
          obsType = obstype,
          nsim = 1000

)
assign(paste0("myMat", myYears[i]), myMat)
}

par(mfrow = c(1,2))
hist(myMat2019[1,], main = "Rovviltregion 1", xlab = "")
hist(myMat2019[8,], main = "Rovviltregion 8", xlab = "")
```

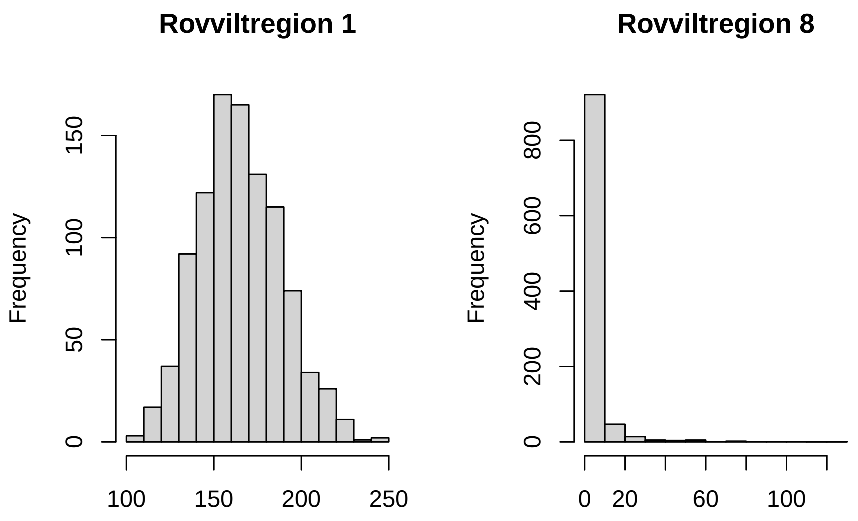


Figure 4.2: Probability distribuition for the number of wolverine, resamlped using data from the NI database and R function in the NIcalc-package.

For some reason the expected values are far from the mean of these distributions. Anders did this exercise once before, and did not get this problem then. We think the difference is that we use eco = NULL this time, in the `importDatasetApi()`, and this cause the output to somehow split into forest and alpine ecosystems. We will ignore this here for this example.

We can also get the reference values in the same way, and then divide one by the other to get scaled values

```
myMatr <- NIcalc::sampleObsMat(
            jerv$referenceValues$ICunitId,
            jerv$referenceValues$expectedValue,
            jerv$referenceValues$distributionFamilyName,
            mu = jerv$referenceValues$distParameter1,
            sig = jerv$referenceValues$distParameter2,
            customDistribution = jerv$referenceValues$customDistribution,
            obsType = obstype,
            nsim =1000
        )

temp <- colSums(myMat2019)/colSums(myMatr)
hist(temp, xlab = "Scaled indicator value for wolverine",main="")
```

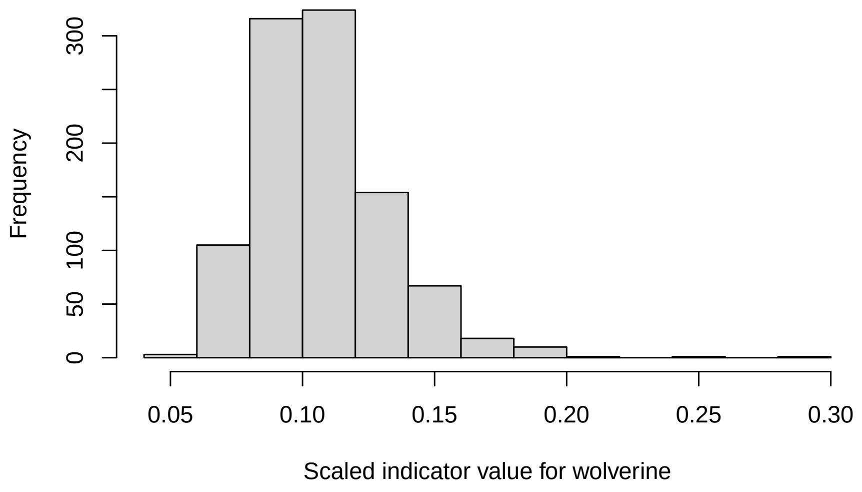


Figure 4.3: Example distribution of scaled indicator values for wolverine.

Then we will create a data frame with the mean indicator values and the SD.

```r
library(matrixStats)
jerv_tbl <- data.frame("raw2019" = round(rowMeans(myMat2019), 2),
                       "sd2019"  = round(matrixStats::rowSds(myMat2019), 2),
                       "ref"     = round(rowMeans(myMatr), 2))
jerv_tbl$scaled <- round(jerv_tbl$raw2019/jerv_tbl$ref, 2)
jerv_tbl$cv <- round(jerv_tbl$sd2019/jerv_tbl$raw2019, 2)
jerv_tbl$region <- jerv$indicatorValues$`2019`$ICunitName
DT::datatable(jerv_tbl)
```

raw2019	sd2019	ref	scaled	cv	region

This is a special case maybe, because the sd is often larger than the mean.

Note that we could use inbuilt NIcalc functions to get the indicator value, like we do below, but that will aggregate to regions, and we want to keep the original geometry.

```r
jervComp <- NIcalc::calculateIndex(
  x       = jerv,
  nsim     = 1000,
  awBSunit = "terrestrialArea",
  fids     = F,    # should fidelities be ignored in
                   # the calculation of Wi?
  tgroups  = F, # should grouping of indicators
                   # into trophic and key indicator
                   # groups be ignored
  keys     = "specialWeight", #"ignore",
)
#> Indices for NIunits 'wholeArea', 'E', 'S', 'W', 'C', 'N'
#> and years '1990', '2000', '2010', '2014', '2019' will be calculated.
#> The 30 index distributions will each be based on  1000 simulations.
#> There are 8 ICunits with observations in data set 'jerv'.
#>
#> Calculating weights that are the same for all years .....
#>
#> Sampling reference values .....
#>
#> Sampling and scaling indicator observations from  1990 .....
#>
#> Sampling and scaling indicator observations from  2000 .....
#>
#> Sampling and scaling indicator observations from  2010 .....
#>
#> Sampling and scaling indicator observations from  2014 .....
#>
#> Sampling and scaling indicator observations from  2019 .....
plot(jervComp$wholeArea)
```

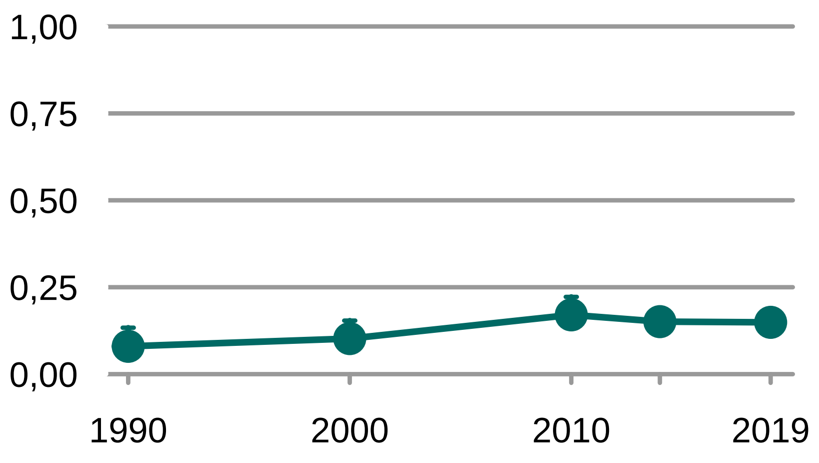


Figure 4.4: The scaled indicator values for wolverine across Norway.

4.2.1.2 Get geometries

Then we can get the spatial geometries associated with the data. These are the so-called “rovviltregioner”. There are eight of them. They are actually linked to the BS-units (municipalities), but we don’t want to plot the outlines of the municipalities. The geometries for the appropriate spatial units of each indicator can be downloaded in .json format via a previously created API for the nature index database: https://ninweb08.nina.no/NaturindeksAPI/index.html To get the file for a specific indicator, one needs to enter the numerical indicator id under “/api/Indicator/{id}/Areas” and then click download. We then converted the .json file to shapefiles for use in R.

```
path <- "P:/41201612_naturindeks_2021_2023_database_og_innsynslosning/Pilot_Forbedring_Innsynsløs
```

```
library(sf)
rov <- sf::read_sf(path)
rov <- sf::st_make_valid(rov)
rov <- rov[rov$area!="DEF jerv",]
```

Clip it against the outline of Norway to make it look more pretty

```
path <- "data/outlineOfNorway_EPSG25833.shp"
nor <- sf::read_sf(path)
nor <- st_transform(nor, crs=st_crs(rov))
```

```
rov <- st_intersection(rov, nor)
#> Warning: attribute variables are assumed to be spatially
#> constant throughout all geometries
```

4.2.1.3 Link data and geometries

Here we copy the data from the table into the geo-file.

```
rov$scaledIndicator <- jerv_tbl$scaled[match(rov$area, jerv_tbl$region)]
rov$cv <- jerv_tbl$cv[match(rov$area, jerv_tbl$region)]
rov$raw <- jerv_tbl$raw2019[match(rov$area, jerv_tbl$region)]
```

Then we can attempt to create some nice example maps where the indicator uncertainty is shown parallel to the scaled values.

```
# colour palette with 10 colours
pal <- grDevices::colorRampPalette(NIviz_colours[["IndMap_cols"]])(10)
```

```
one <- tm_shape(rov)+
  tm_polygons(col="scaledIndicator",
              border.col = "white",
              style = "cont",
              breaks = seq(0,1, length.out = 11),
              palette = pal)


two <- tm_shape(rov)+
  tm_polygons(col="cv",
              border.col = "black")

#three <- tm_shape(rov)+
#  tm_polygons(col="raw",
#               border.col = "white")


tmap_mode("view")
#> tmap mode set to interactive viewing

tmap_arrange(one, two,
             sync=T,
             widths = c(.75, .25),
             heights = c(1, 0.5)
             )
```

4.3 Additional ideas

4.3.1 Gradient density plots for interactive maps

Another idea we pursued is to build on the current map setup (Figure 1.2), but include a visualization of the uncertainty distribution in the pop-up window that appears in the top-right corner when hovering over a region in the map. This idea is sketched out below in Figure 4.5, and we've written code to visualize corresponding distributions using the same color palette as the map in Chapter 5.1.

It's important to note though that reading distributions is often unintuitive for untrained people. If such an approach were adopted, it would be important to have some very short, concise intructions on how to read and interpret a density plot of a distribution.

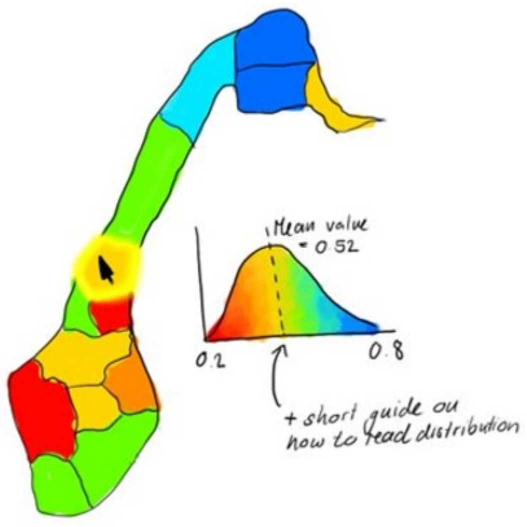


Figure 4.5: Sketch of a map with uncertainty distribution popping up for active area.

4.3.2 Other approaches

Uncertainty visualization in maps is a very prolific research topic at present, and having side-by-side maps of average values and (relative) uncertainty is just one approach that has been proposed.

There are, for example, a wide variety of ways for including information about uncertainty via additional layers in a map.[1] Visualizing uncertainty as transparency (i.e. more transparent/lighter colors for more uncertain values) may potentially work well with the red to blue rainbow scale and the relatively coarse spatial resolution of indicators and indices in the Nature Index work. An illustrative example can be found in Figures 2 and 3 in Petr Kubíček and Čeněk Šašinka[2].

There are also highly innovative approaches that are worth considering. Lydia R Lucchesi and Christopher K Wikle[3], for example, propose to use pixelation to represent uncertainty. Specifically, each pixel within a spatial unit is assigned a color based on a value randomly drawn from the statistical distribution of the quantity of interest. The spatial unit as a whole therefore has pixels sporting a range of different colors, which will result in areas with more certain values

[1] Alan M MacEachren et al., "Visual Semiotics & Uncertainty Visualization: An Empirical Study," *IEEE Transactions on Visualization and Computer Graphics* 18, no. 12 (2012): 2496–2505.

[2] "Thematic Uncertainty Visualization Usability–Comparison of Basic Methods," *Annals of GIS* 17, no. 4 (2011): 253–63.

[3] "Visualizing Uncertainty in Areal Data with Bivariate Choropleth Maps, Map Pixelation and Glyph Rotation," *Stat* 6, no. 1 (2017): 292–302.

seeming more “single-colored” while areas with very uncertain values will be a patchwork of different colors. Lucchesi and Wikle[4] went a step further even to animate their map and having it show the random draws in real-time.

4

Chapter 5

Other figures

This final chapter presents a range of other figures and charts that may help with improving the display of key information on the website.

5.1 Gradient density plots for interactive maps

The maps presented on the webpage do not include any representation of uncertainty. One way of including that information without having to add additional (layers to the) maps would be to build on the interactive functions included so far and present a probability distribution for the given region and year. This could be displayed in the same box that currently appears when hovering over an area and displays area name and average indicator value.

To make these density plots, we use the previously simulated bootstrap samples, using Jerv as an example:

```
i <- "Jerv"
bootStrp <- readRDS(paste0("data/", i, "_bootstrapped_scaled.rds"))
head(bootStrp)
#> # A tibble: 6 x 4
#>   ICunitID ICunitName      year  scaledIndicator
#>   <chr>    <chr>           <chr>           <dbl>
#> 1 1302     Rovviltregion 8 1990            0.194
#> 2 1302     Rovviltregion 8 1990            0.233
#> 3 1302     Rovviltregion 8 1990            0.190
#> 4 1302     Rovviltregion 8 1990            0.207
#> 5 1302     Rovviltregion 8 1990            0.199
#> 6 1302     Rovviltregion 8 1990            0.164
```

We have considered forcing all indicator values > 1 to display as 1, but this messes up when plotting density functions. Still, this conversion is required for calculating the point estimate (median) and we therefore make a copy of the data in which no values are larger than 1.

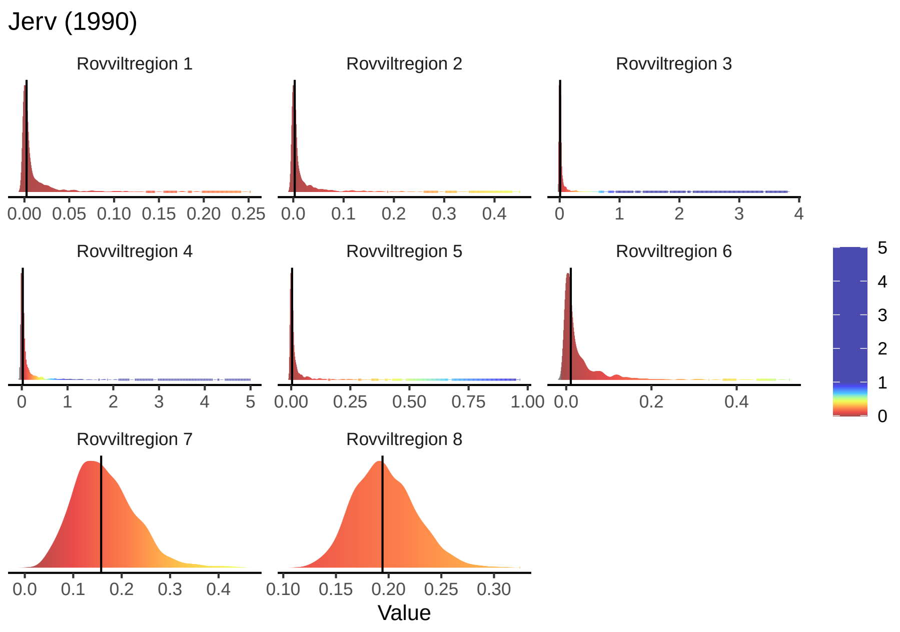
```
bootStrp1 <- bootStrp
bootStrp1$scaledIndicator[which(bootStrp1$scaledIndicator > 1)] <- 1
```

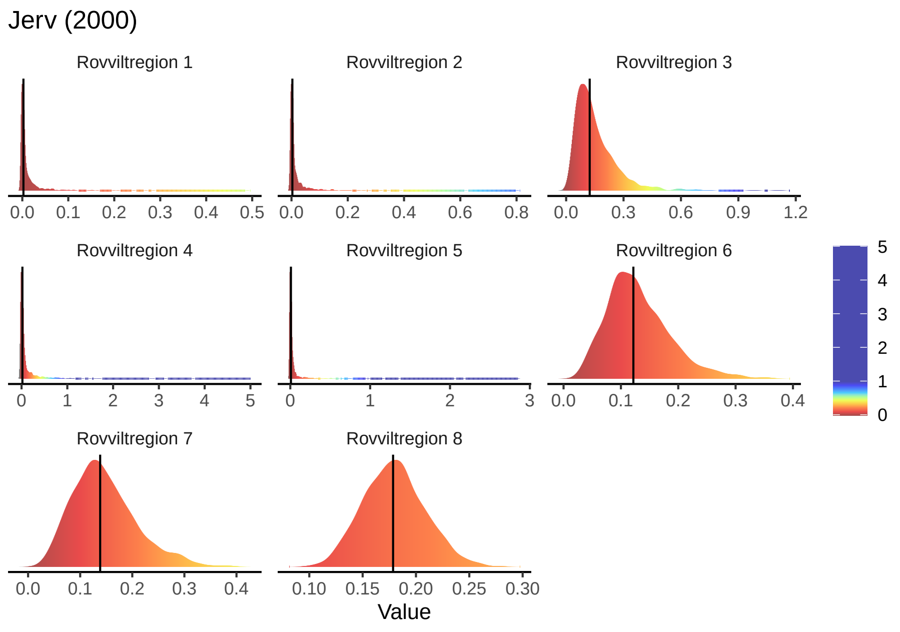
Next, we need to manually calculate the probability densities for the indicator values in each year and area. This is necessary for making density plots with a color gradient fill (but see further below for an alternative using the "ggridges" package which does not require this intermediate step).

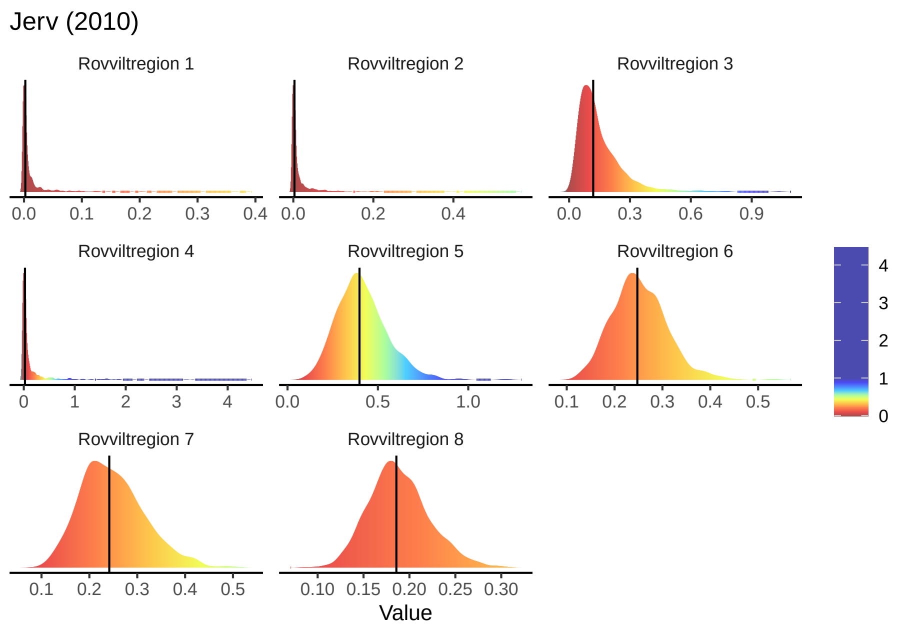
```
years <- unique(bootStrp$year)
areas <- unique(bootStrp$ICunitName)

pDens <- data.frame()
for(t in years){
  for(a in areas){

    bootStrp_sub <- subset(bootStrp, year == t & ICunitName == a)

    pDens_a_t <- data.frame(
      ICunitName = a,
      year = t,
      x = density(bootStrp_sub$scaledIndicator)$x,
      y = density(bootStrp_sub$scaledIndicator)$y
    )

    pDens <- rbind(pDens, pDens_a_t)
  }
}
```

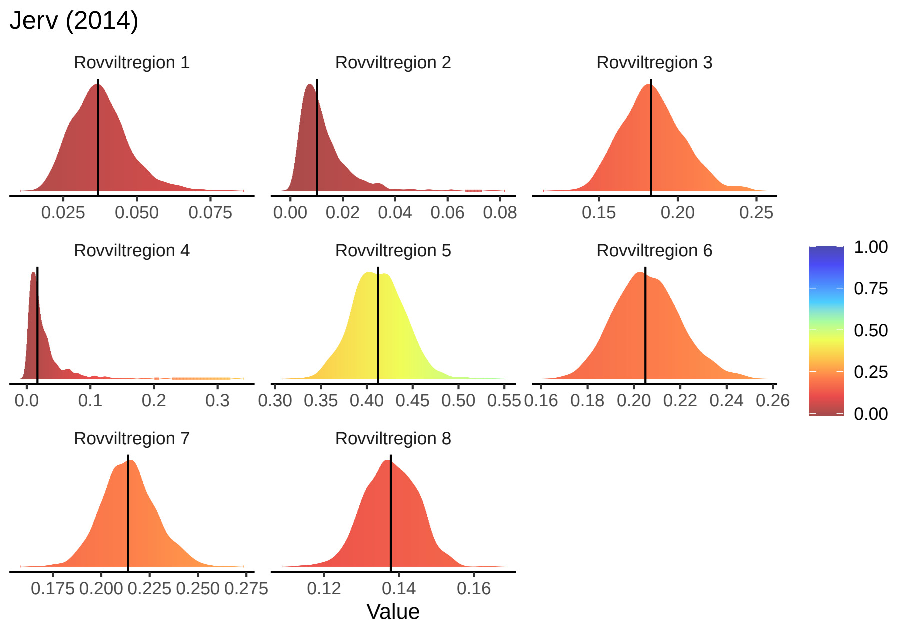
We can then proceed to plotting the probability density functions with a color gradient under the line:

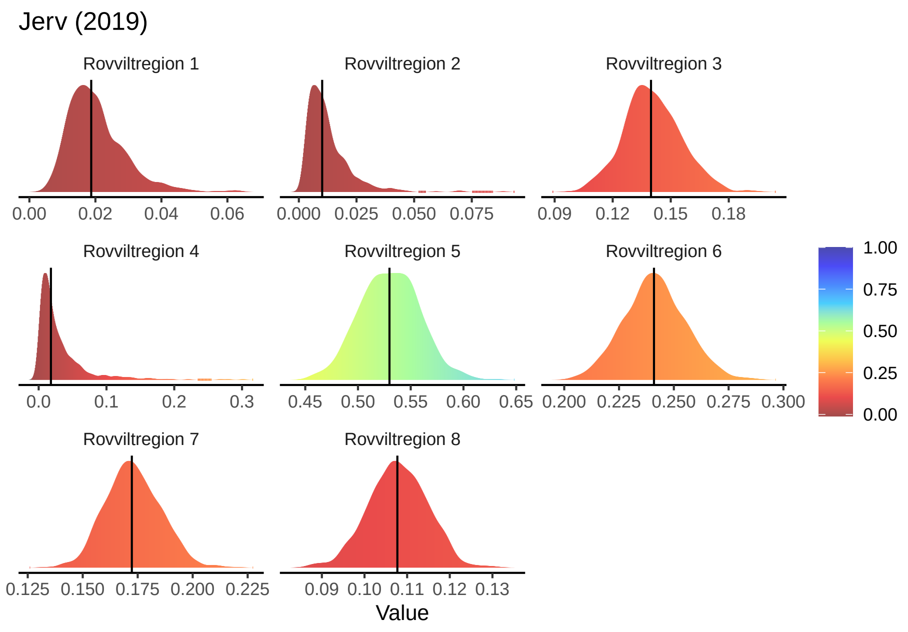
```
# Set maximum value (we will not plot beyond 4)

for(t in years){

  # Take data subsets for a given year
  bootStrp_yr <- bootStrp[which(bootStrp$year == t),]
  bootStrp1_yr <- bootStrp1[which(bootStrp1$year == t),]
  pDens_yr <- pDens[which(pDens$year == t),]
  # Extract distribution medians
  sum_values <- bootStrp1_yr %>%
    group_by(ICunitName) %>%
    summarise(sumStat = median(scaledIndicator))

  # Set maximum plotting value (never > 5) and mapping for custom color scale
  maxVal <- ifelse(max(pDens_yr$x) > 5, 5, max(pDens_yr$x))

  if(maxVal < 1+1/9){
    valuesMap <- c(-0.1, seq(0, 1, length.out = 10))
    colorMap <- c("#1F8C81", NIviz_colours$IndMap_cols)
  }else{
    valuesMap <- c(-0.1, c(seq(0, 1, length.out = 10), maxVal)/maxVal)
    colorMap <- c("#1F8C81", NIviz_colours$IndMap_cols, "#4B4BAF")
  }

  # Plot densities
  print(
    ggplot(subset(pDens_yr, x <= maxVal), aes(x, y)) +
      geom_segment(aes(xend = x, yend = 0, colour = x)) +
      #scale_color_NIviz_c(name = "IndMap_cols") +
      scale_colour_gradientn(colours = colorMap,
                             values = valuesMap,
                             limits = c(-0.01, ifelse(maxVal < 1, 1, maxVal))) +
      ggtitle(paste0(i, " (", t, ")")) +
      xlab("Value") +
      geom_vline(data = sum_values, aes(xintercept = sumStat)) +
      facet_wrap(~ ICunitName, scales = 'free') +
      theme_classic() +
      theme(strip.background = element_blank(),
            legend.title = element_blank(),
            axis.line.y = element_blank(), axis.ticks.y = element_blank(),
            axis.text.y = element_blank(), axis.title.y = element_blank())
  )
}
```


Jerv (1990)
Rovviltregion 1
0.00 0.05 0.10 0.15 0.20 0.25
Rovviltregion 2
0.0 0.1 0.2 0.3 0.4
Rovviltregion 3
0 1 2 3 4
Rovviltregion 4
0 1 2 3 4 5
Rovviltregion 5
0.00 0.25 0.50 0.75 1.00
Rovviltregion 6
0.0 0.2 0.4
Rovviltregion 7
0.0 0.1 0.2 0.3 0.4
Rovviltregion 8
0.10 0.15 0.20 0.25 0.30
Value
5 4 3 2 1 0


Jerv (2000)
Rovviltregion 1
0.0 0.1 0.2 0.3 0.4 0.5
Rovviltregion 2
0.0 0.2 0.4 0.6 0.8
Rovviltregion 3
0.0 0.3 0.6 0.9 1.2
Rovviltregion 4
0 1 2 3 4 5
Rovviltregion 5
0 1 2 3
Rovviltregion 6
0.0 0.1 0.2 0.3 0.4
Rovviltregion 7
0.0 0.1 0.2 0.3 0.4
Rovviltregion 8
0.10 0.15 0.20 0.25 0.30
Value
5 4 3 2 1 0

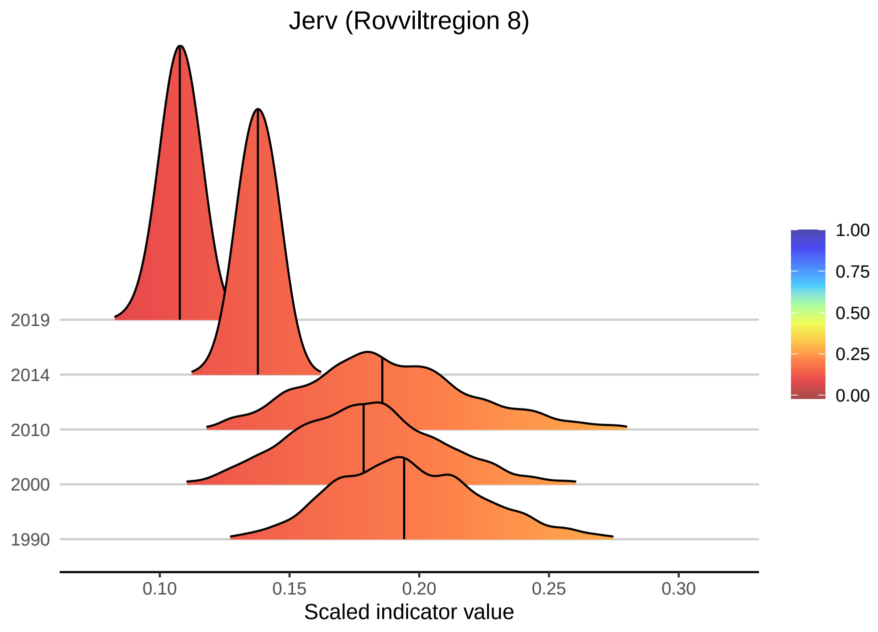

Jerv (2010)
Rovviltregion 1
Rovviltregion 2
Rovviltregion 3
Rovviltregion 4
Rovviltregion 5
Rovviltregion 6
Rovviltregion 7
Rovviltregion 8
Value
0
1
2
3
4


Jerv (2014)
Rovviltregion 1
Rovviltregion 2
Rovviltregion 3
Rovviltregion 4
Rovviltregion 5
Rovviltregion 6
Rovviltregion 7
Rovviltregion 8
Value
0.00
0.25
0.50
0.75
1.00

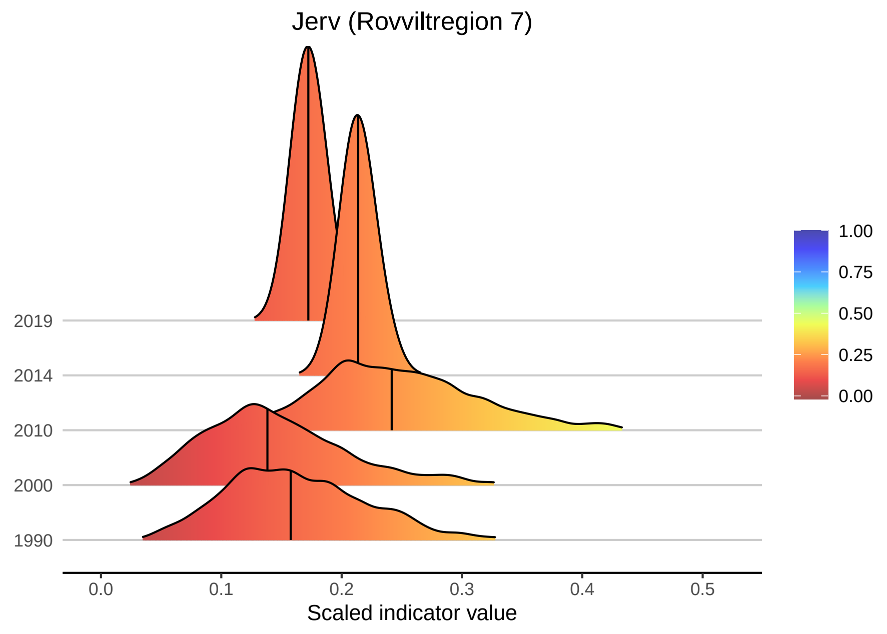


Gradient density plots might also be useful for other types of visualization of indicator and index data. For example, there is a very attractive way of using ridgeplots for visualizing time series inclusing uncertainty. Could look something like this:

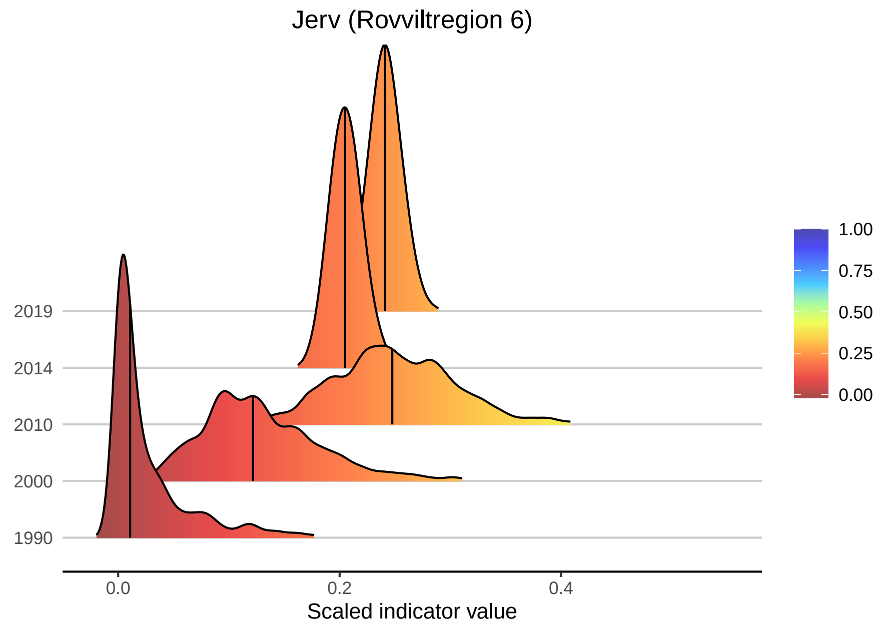
```
for(a in areas){

  # Take data subsets for a given area
  bootStrp_ar <- bootStrp[which(bootStrp$ICunitName == a),]
  pDens_ar <- pDens[which(pDens$ICunitName == a),]

  # Set maximum plotting value (never > 5) and mapping for custom color scale
  maxVal <- ifelse(max(pDens_ar$x) > 5, 5, max(pDens_ar$x))

  if(maxVal < 1+1/9){
    valuesMap <- c(-0.1, seq(0, 1, length.out = 10))
    colorMap <- c("#1F8C81", NIviz_colours$IndMap_cols)
  }else{
    valuesMap <- c(-0.1, c(seq(0, 1, length.out = 10), maxVal)/maxVal)
    colorMap <- c("#1F8C81", NIviz_colours$IndMap_cols, "#4B4BAF")
  }

  # Plot densities
  print(
  ggplot(subset(bootStrp_ar, scaledIndicator <= maxVal), aes(x = scaledIndicator, y = year, fill
    geom_density_ridges_gradient(scale = 5, rel_min_height = 0.01, quantile_lines = TRUE, quantil
    scale_fill_gradientn(colours = colorMap,
                         values = valuesMap,
                         limits = c(-0.02, ifelse(maxVal < 1, 1, maxVal))) +
    ggtitle(paste0(i, " (", a, ")")) +
    xlab("Scaled indicator value") + ylab("") +
    theme_classic() +
    theme(legend.title = element_blank(), plot.title = element_text(hjust = 0.5),
          axis.line.y = element_blank(), axis.ticks.y = element_blank(),
          panel.grid.major.y = element_line(color = "grey80"))
  )

}
#> Picking joint bandwidth of 0.00457
```



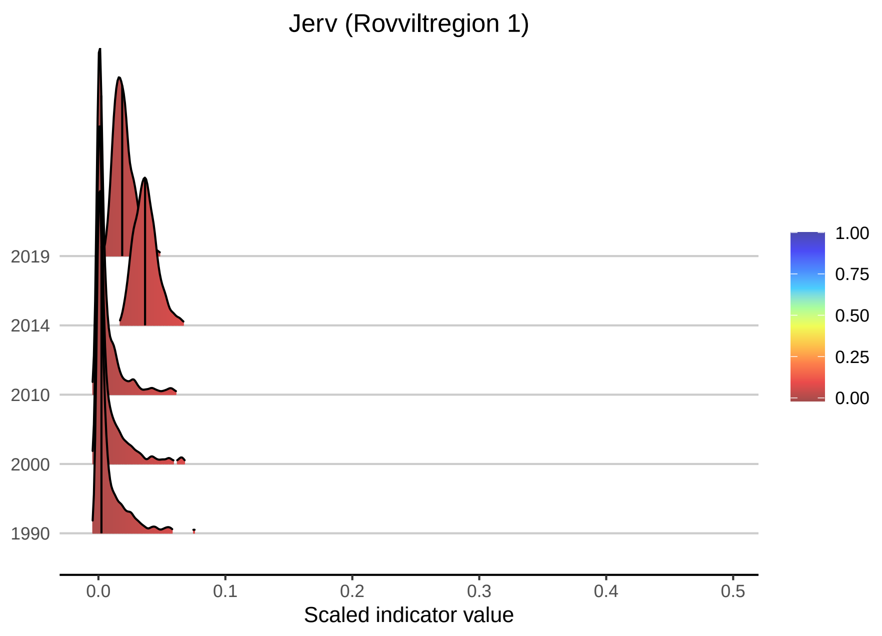
```
#> Picking joint bandwidth of 0.00947
```



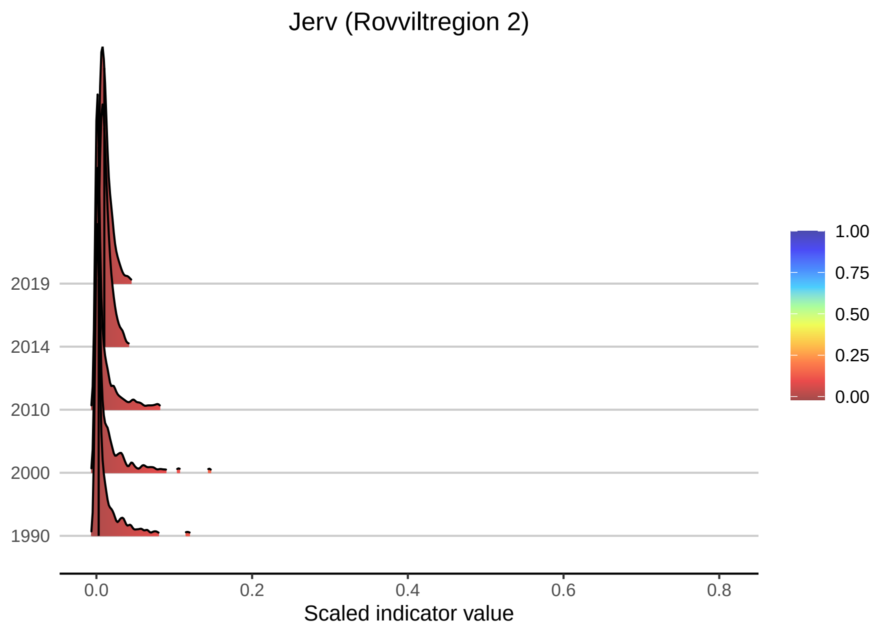
```
#> Picking joint bandwidth of 0.00715
```



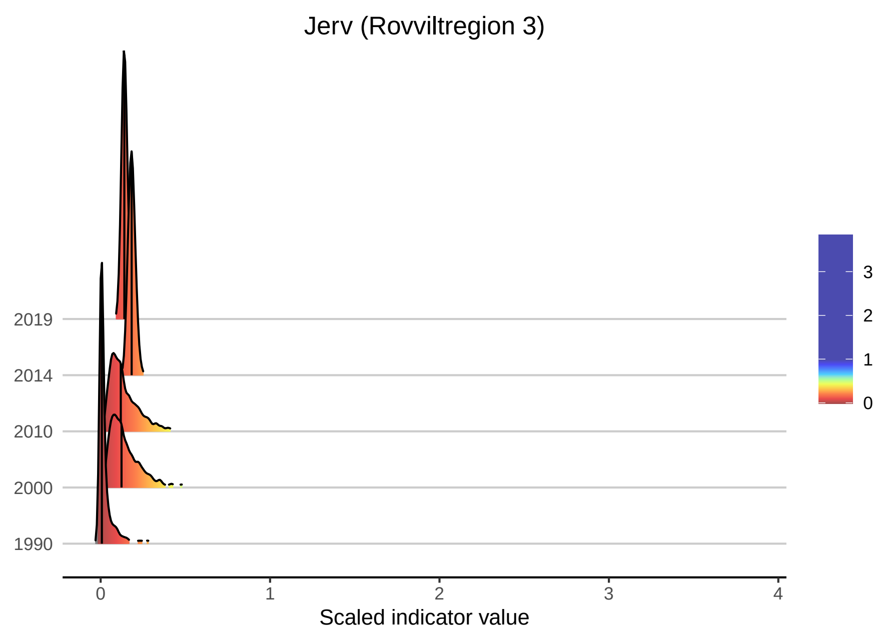
```
#> Picking joint bandwidth of 0.00184
```



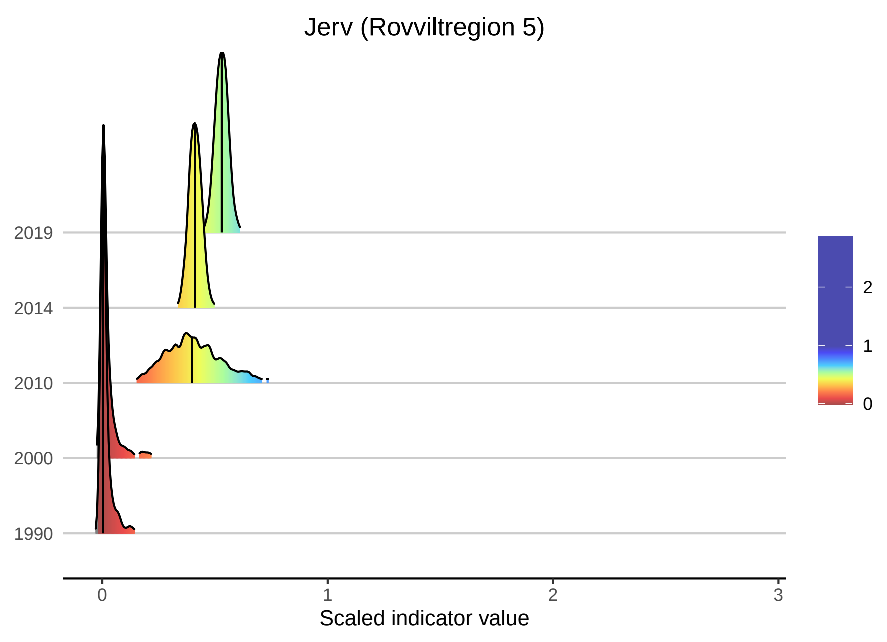
```
#> Picking joint bandwidth of 0.00215
```



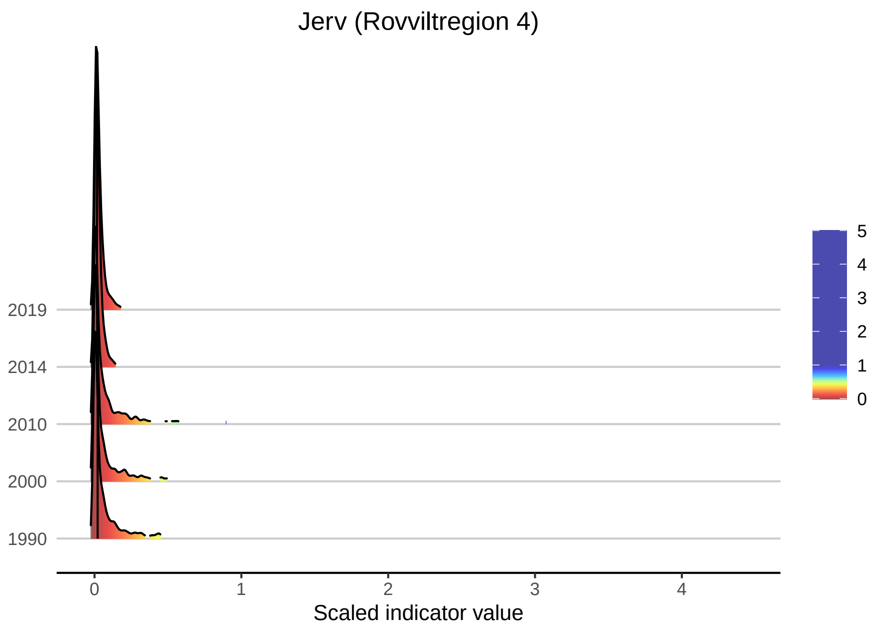
```
#> Picking joint bandwidth of 0.0101
```



```
#> Picking joint bandwidth of 0.00966
```



```
#> Picking joint bandwidth of 0.0112
```



5.2 Ecosystem fidelity

All indicators are assigned to at least one ecosystem, but a fair number of them are assigned to multiple ecosystems by means of proportions. Wolverine (Jerv), for example, is assigned with 25% to forest and and 75% to mountain. This basic information could easily be displayed on each indicator's page on naturindeks.no.

The relevant information is found in the assembled indicator data under $indicators:

```
i <- "jerv"
indexData <- readRDS(paste0("data/", i, "_assemebled.rds"))

str(indexData$indicators)
#> 'data.frame':    1 obs. of  9 variables:
#>  $ id                : num 88
#>  $ name              : chr "Jerv"
#>  $ keyElement        : logi FALSE
#>  $ functionalGroup   : chr "Topp-predator generalist"
#>  $ functionalGroupId: num 8
#>  $ scalingModel      : chr "Low"
#>  $ scalingModelId    : num 1
#>  $ Fjell               : num 75
#>  $ Skog                : num 25
```

Any ecosystem type relevant to a specific indicator appears as a separate column in this dataframe, and contains a value representing the % fidelity to that ecosystem type.

Using separately stored information on available ecosystem types, we can assemble this data for all of our example indicators:

```
# Load ecosystem info
EcoSysInfo <- readRDS("data/EcosystemInfo.rds")

# Indicator list
indicator <- c("Dikesoldogg",
               "Jerv",
               "Elg",
               "Lomvi",
               "Havørn",
               "Lange")

# Assemble fidelity data
fidData <- data.frame()

for(i in 1:length(indicator)){

  indexData <- readRDS(paste0("data/", indicator[i], "_assemebled.rds"))

  ColIdx <- which(names(indexData$indicators) %in% EcoSysInfo$ecosystem)

  fidDataI <- data.frame(
    indicator = indicator[i],
    ecosystem = names(indexData$indicators)[ColIdx],
    fidelity = unname(as.numeric(indexData$indicators[,ColIdx])))

  fidData <- rbind(fidData, fidDataI)
}
```

This gives us a dataframe with all indicators and their fidelity to different ecosystems:

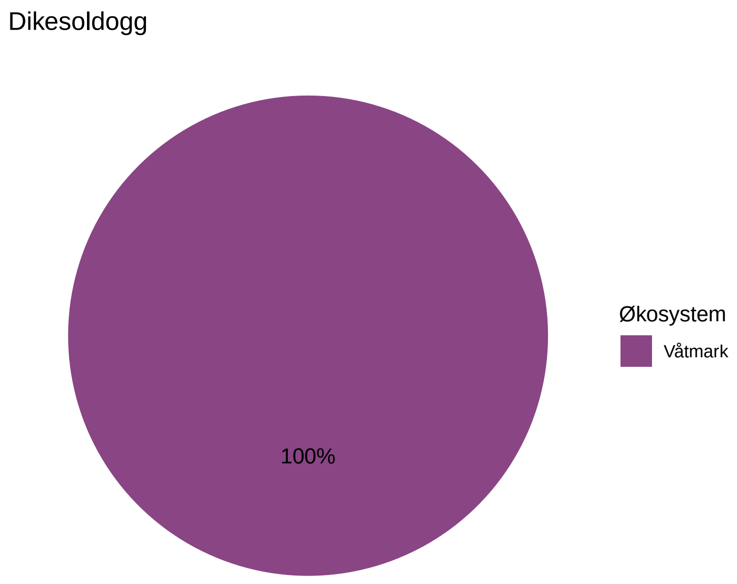
```
print(fidData)
#>     indicator         ecosystem fidelity
#> 1 Dikesoldogg           Våtmark      100
#> 2        Jerv             Fjell       75
#> 3       Jerv              Skog       25
#> 4        Elg              Skog      100
#> 5      Lomvi      Hav-pelagisk       67
#> 6      Lomvi Kystvann-pelagisk       33
#> 7     Havørn Kystvann-pelagisk      100
#> 8      Lange           Havbunn       80
#> 9      Lange     Kystvann-bunn       20
```

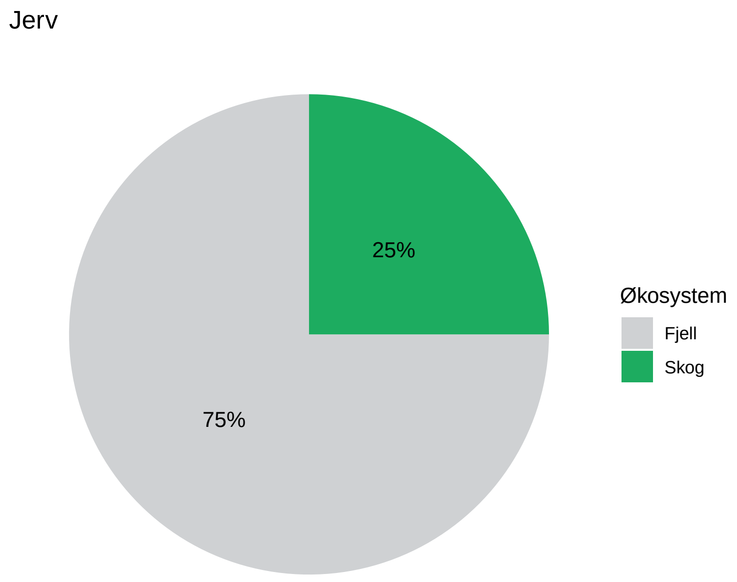
Before plotting, we match the integer ecosystem IDs to make sure the colour mapping works correctly:

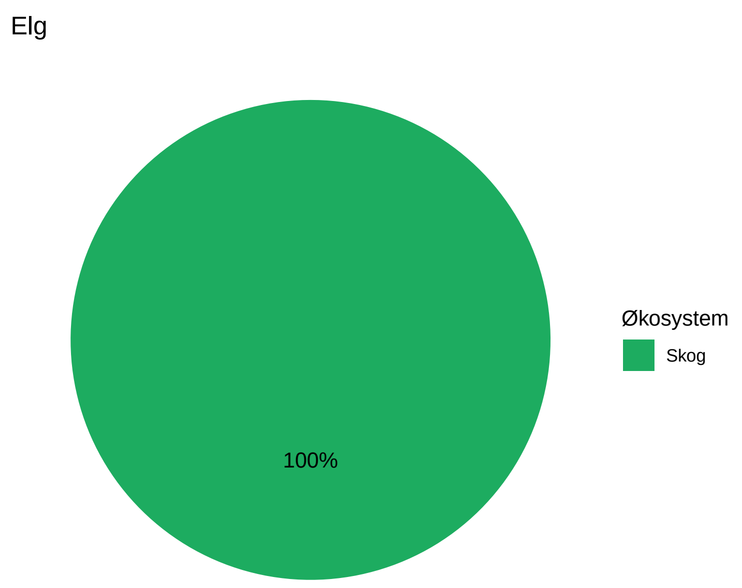
```
fidData <- merge(fidData, EcoSysInfo, all.x = TRUE)
```

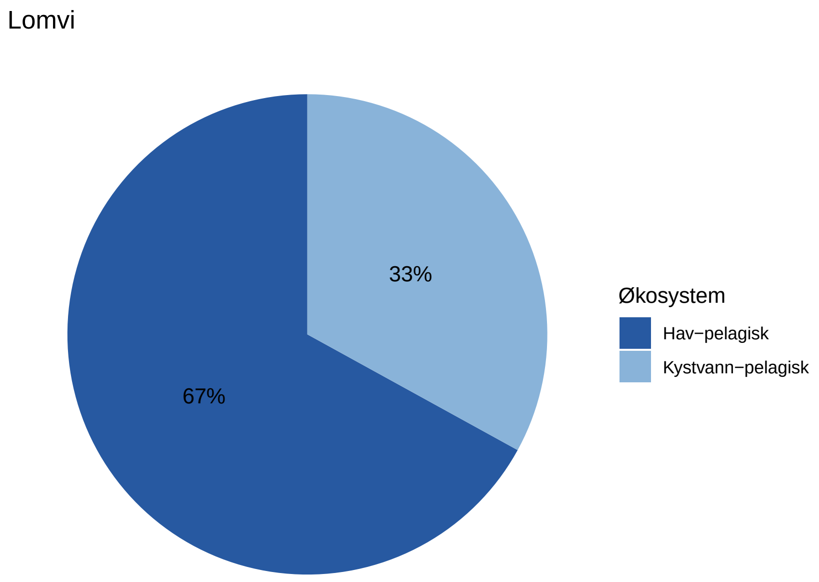
Next, we'll visualize this information for each indicator by means of pie charts.

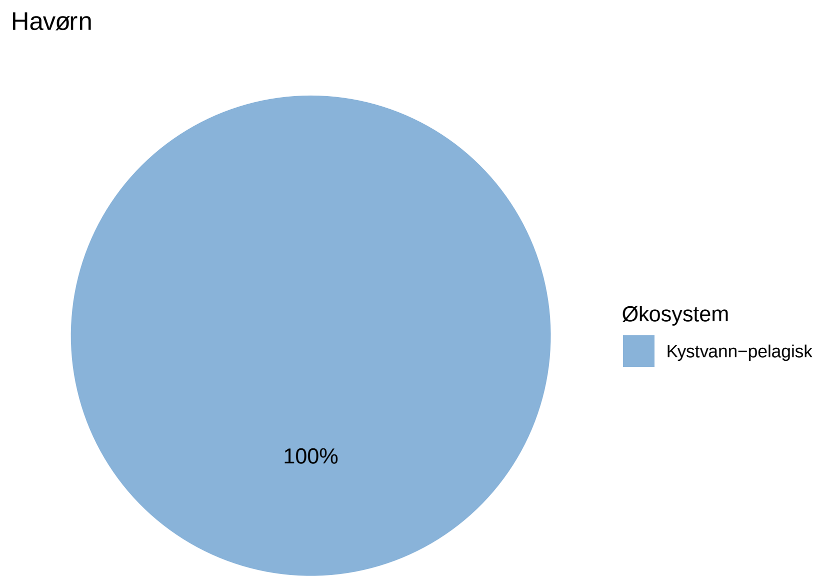
```
EcoSys_cols <- NIviz_colours$EcoSys_cols[1:11]
names(EcoSys_cols) <- EcoSysInfo$ecosystem

for(i in 1:length(indicator)){

  sub_fidData <- fidData[which(fidData$indicator == indicator[i]),]

  print(
    ggplot(sub_fidData, aes(x = "", y = fidelity, fill = fct_inorder(ecosystem))) +
    ggtitle(indicator[i]) +
    geom_bar(stat = "identity") +
    geom_text(aes(label = paste0(fidelity, "%")),
              position = position_stack(vjust = 0.5)) +
    coord_polar(theta = "y") +
    scale_fill_manual(name = "Økosystem", values = EcoSys_cols[which(names(EcoSys_cols) %in% sub_
    theme_void()
  )

}
```

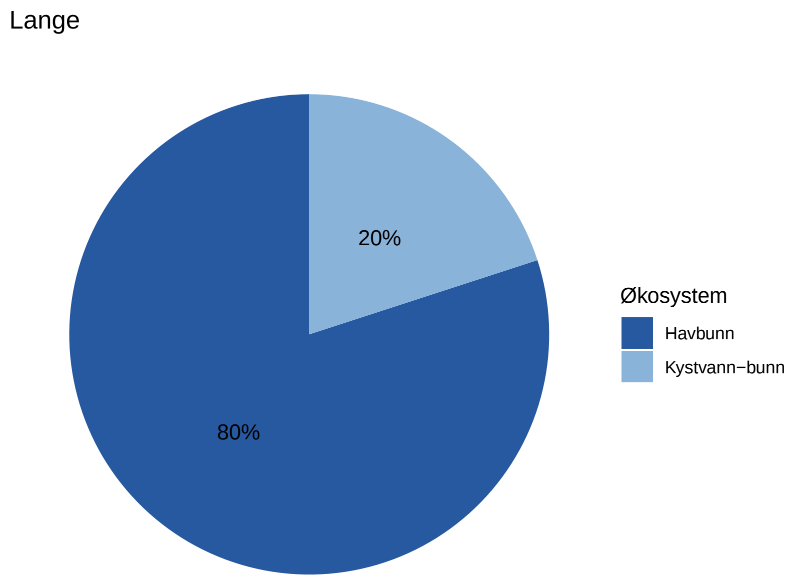

Dikesoldogg
100%
Økosystem
Våtmark


Jerv
25%
75%
Økosystem
Fjell
Skog


Elg
100%
Økosystem
Skog

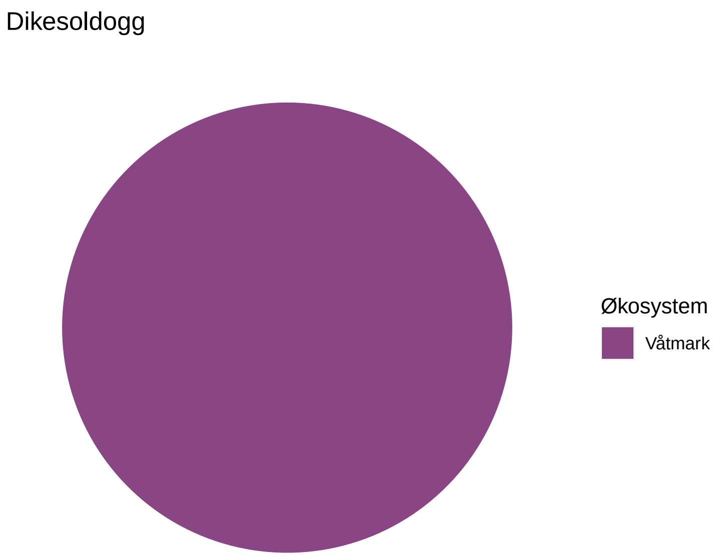

Lomvi
33%
67%
Økosystem
Hav−pelagisk
Kystvann−pelagisk

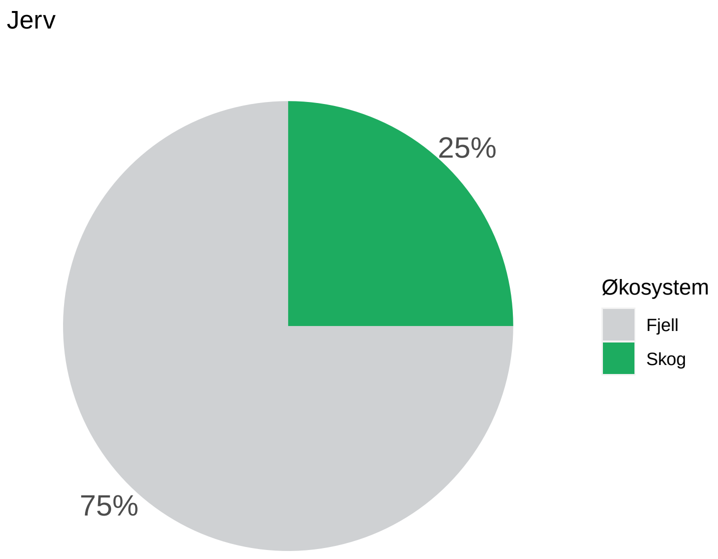


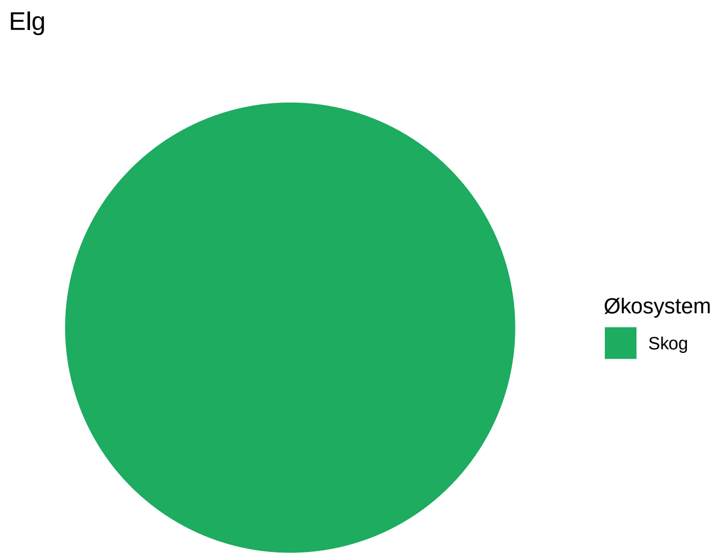


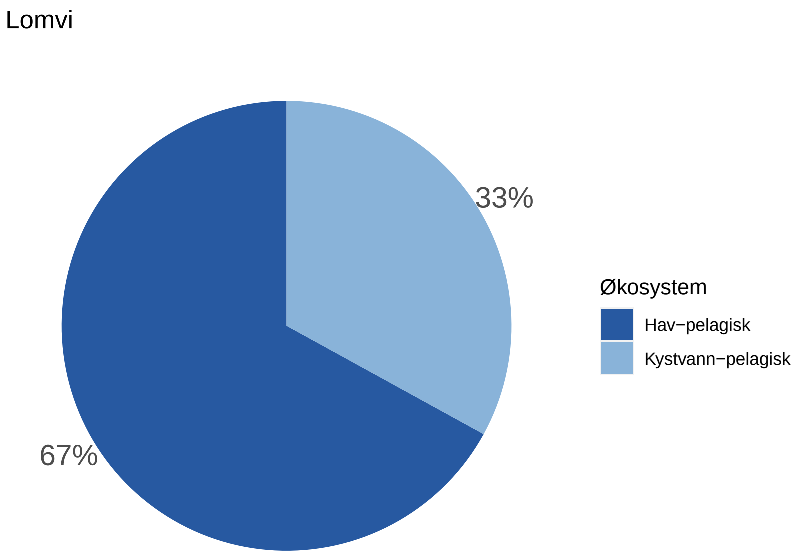
Depending on how large these pie charts should appear on the website in the end, it may be necessary to move the percentage labels outside the pies. Doing that is a bit more cumbersome, but works with the code below unless the indicator belongs 100 % to one dataset. When that is the case, the solution below does not print the percentage label at all (see indicator Lange) and we have not figured out how to fix that yet.

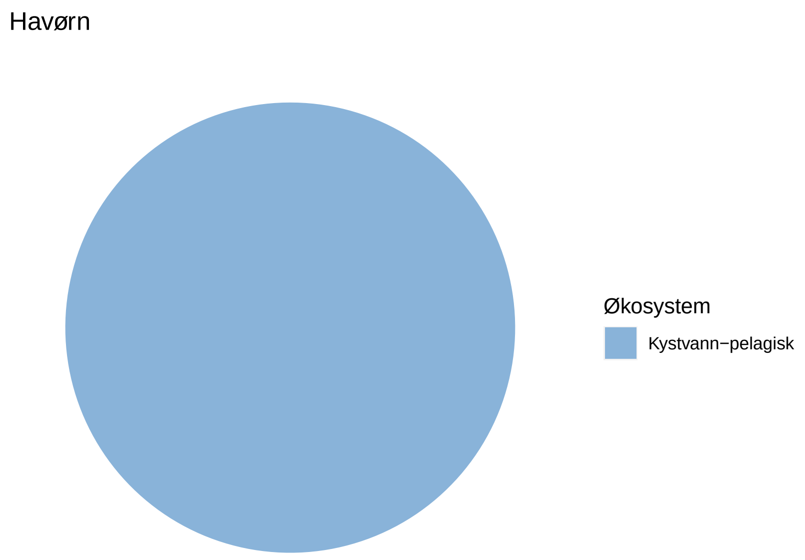
```
for(i in 1:length(indicator)){

  sub_fidData <- fidData[which(fidData$indicator == indicator[i]),]

  posData <- sub_fidData %>%
  mutate(csum = rev(cumsum(rev(fidelity))),
         pos = fidelity/2 + lead(csum, 1),
         pos = if_else(is.na(pos), fidelity/2, pos))

  print(
  ggplot(sub_fidData, aes(x = "", y = fidelity, fill = ecosystem)) +
    ggtitle(indicator[i]) +
    geom_bar(stat = "identity") +
    coord_polar(theta = "y") +
    #scale_fill_manual(values = EcoSys_cols) +
    scale_fill_manual(name = "Økosystem", values = EcoSys_cols[which(names(EcoSys_cols) %in% sub_
    scale_y_continuous(breaks = posData$pos, labels = paste0(sub_fidData$fidelity, "%")) +
    theme(axis.ticks = element_blank(),
          axis.title = element_blank(),
          axis.text = element_text(size = 15),
          panel.background = element_rect(fill = "white"))
  )

}
```

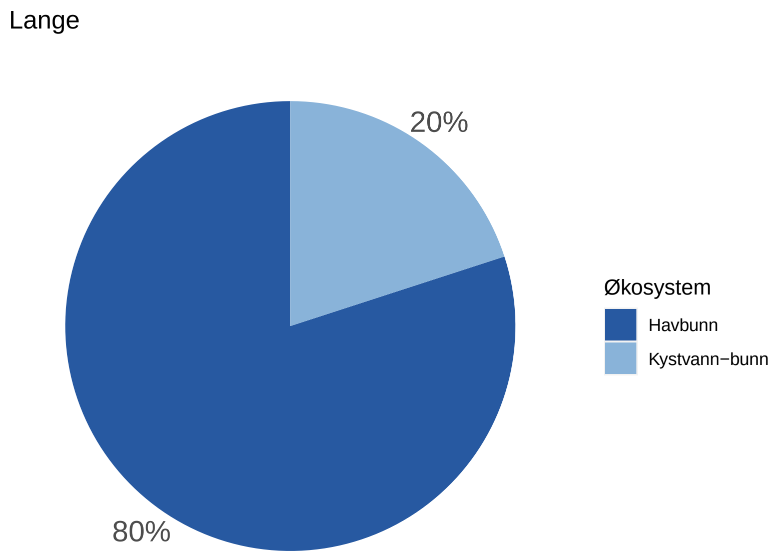

Dikesoldogg
Økosystem
Våtmark


Jerv
25%
75%
Økosystem
Fjell
Skog


Elg
Økosystem
Skog


Lomvi
33%
67%
Økosystem
Hav−pelagisk
Kystvann−pelagisk


Havørn
Økosystem
Kystvann−pelagisk


Lange
20%
80%
Økosystem
Havbunn
Kystvann−bunn

5.3 Impact factor wordclouds

```r
library(wordcloud)
library(NIcalc)
library(wordcloud2)
library(RColorBrewer)
library(readxl)
library(dplyr)
library(tm)
```

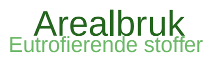
As per now, the website shows the most important impact factors for each indicator as a list in a pop-up window on the top-right. This is information that is likely of great interest to the general public, and would benefit from being placed more visible on the website and in a more engaging way than as a list. One attractive alternative way of presenting is via wordclouds.

The following wordcloud figures show the pressure factors for each indicator, where both the color saturation and text size represent the importance of each pressure factor. Small text size and low saturation means low importance, while large text size represent pressure factors with high importance to the indicator.

```r
# Load dataset
pressure = read_excel("P:/41201612_naturindeks_2021_2023_database_og_innsynslosning/Pilot_Forbedr

# Filter for species of interest
pressure = pressure %>% filter(navn_norsk == "Elg" | navn_norsk == "Dikesoldogg" | navn_norsk ==
pressure = arrange(pressure, by = navn_norsk)
pressure = pressure %>% rename(PressureFactor = Paavirkningsfaktor, PressureValue = FK_Paavirknin

# Remove all instances of "Ikke rel/ukjent" category" and increase the value of pressure factors
pressure = pressure %>% filter(PressureValue != 7) %>% mutate(PressureValue = PressureValue*2)

# Create separate datasets for the different species
dikesoldogg = pressure %>% filter(navn_norsk == "Dikesoldogg") %>% dplyr::select(PressureFactor,

elg = pressure %>% filter(navn_norsk == "Elg") %>% dplyr::select(PressureFactor, PressureValue)

havørn = pressure %>% filter(navn_norsk == "Havørn") %>% dplyr::select(PressureFactor, PressureVa

lange = pressure %>% filter(navn_norsk == "Lange") %>% dplyr::select(PressureFactor, PressureValu

lomvi = pressure %>% filter(navn_norsk == "Lomvi") %>% dplyr::select(PressureFactor, PressureValu
# Create a custom color gradient (green in this case)
pressureColors = c("#bde4aa", "#addb9d", "#9cd28f", "#8cc982", "#7dc275", "#6cb967", "#
```

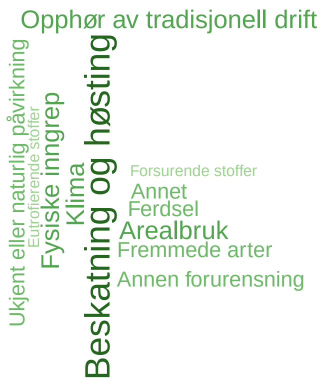
5.3.1 Dikesoldogg (oblong-leaved sundew)

```
# Dikesoldogg
wordcloud(words = dikesoldogg$PressureFactor, freq = dikesoldogg$PressureValue, min.fr
```



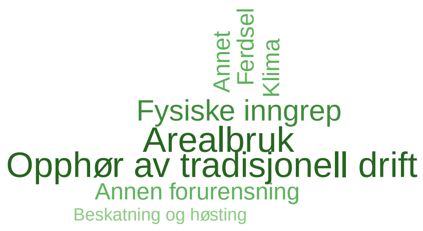
5.3.2 Elg (Moose)

```
wordcloud(words = elg$PressureFactor, freq = elg$PressureValue, min.freq = 1, max.word
```



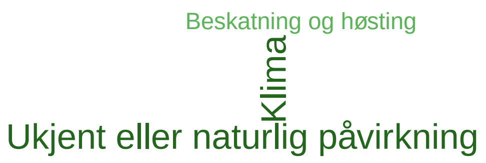
5.3.3 Havørn (White-tailed eagle)

```
wordcloud(words = havørn$PressureFactor, freq = havørn$PressureValue, min.freq = 1, max.words = 2
```



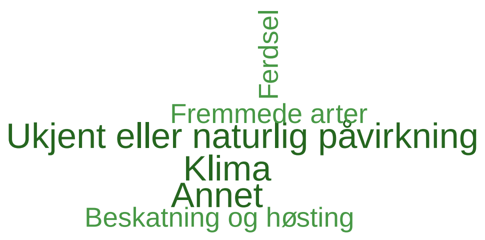
5.3.4 Lange (Common ling)

```
wordcloud(words = lange$PressureFactor, freq = lange$PressureValue, min.freq = 1, max.
```



5.3.5 Lomvi (common guillemot)

```
wordcloud(words = lomvi$PressureFactor, freq = lomvi$PressureValue, min.freq = 1, max.words = 20
```



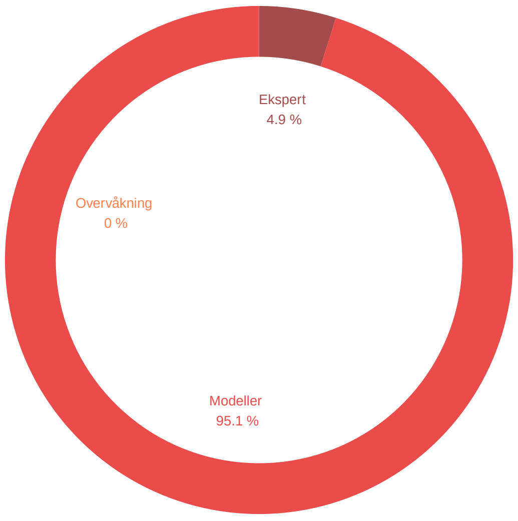
5.4 Data type

For data type, pie charts can also be a good option for visualization. The same would also hold for other pieces of key information that are not currently displayed, such as the proportion of imputed data points.

```
source("R/colorPalettes.R")
```

```
#Elg datatyper
data <- data.frame(
  category=c("Ekspert",
             "Modeller",
             "Overvåkning"),
  count=c(4.9, 95.1, 0)
)

# load library
library(tidyverse)
# Compute percentages
data$fraction <- data$count / sum(data$count)

# Compute the cumulative percentages (top of each rectangle)
data$ymax <- cumsum(data$fraction)
```

```
# Compute the bottom of each rectangle
data$ymin <- c(0, head(data$ymax, n=-1))

# Compute label position
data$labelPosition <- (data$ymax + data$ymin) / 2
data$labelPosition[3]<-0.8
# Compute a good label
data$label <- paste0(data$category, "\n ", data$count, " %")
library(ggrepel)
# Make the plot
ggplot(data, aes(ymax=ymax, ymin=ymin, xmax=4, xmin=3, fill=category)) +
  geom_rect() +
  geom_text(x=2, aes(y=labelPosition, label=label, color=category), size=2.5) + # x here controls
  scale_fill_NIviz_d("IndMap_cols") +
  scale_colour_NIviz_d("IndMap_cols") +
  coord_polar(theta="y") +
  xlim(c(-1, 4)) +
  theme_void() +
  theme(legend.position = "none")
```



5.5 Weights

Finally, today's website also does not present the indicator- or area-weights, even though they are absolutely central in the calculation of spatially aggregated indicators and indices. Visualization of indicator weights for indices (Nature Index for different ecosystems and thematic indices) are routinely made during the updating of the Nature Index and can be found in the reports (e.g. Figure 5.1). It might also be worth considering to make information on weighting at different levels available. This could, for example, include the area weights applied when making both composite indices and averaged indicators, or the weights given to any specific indicator in the Nature Index for different ecosystems and for different thematic indices.

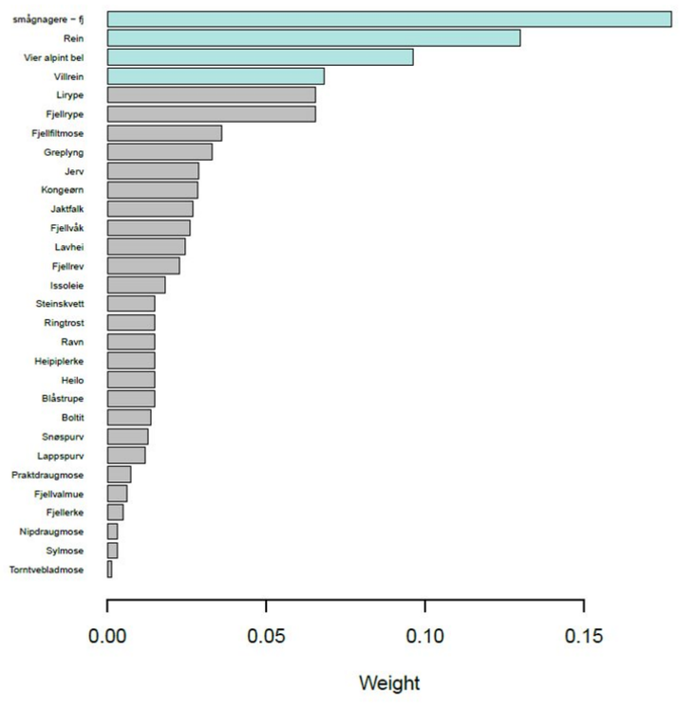


Figure 5.1: Indicator weights for the Nature Index for mountains, as presented in the Nature Index report.

Kubíček, Petr, and Čeněk Šašinka. "Thematic Uncertainty Visualization Usability–Comparison of Basic Methods." *Annals of GIS* 17, no. 4 (2011): 253–63.

Lucchesi, Lydia R, and Christopher K Wikle. "Visualizing Uncertainty in Areal Data with Bivariate Choropleth Maps, Map Pixelation and Glyph Rotation." *Stat* 6, no. 1 (2017): 292–302.

MacEachren, Alan M, Robert E Roth, James O'Brien, Bonan Li, Derek Swing-

ley, and Mark Gahegan. "Visual Semiotics & Uncertainty Visualization: An Empirical Study." *IEEE Transactions on Visualization and Computer Graphics* 18, no. 12 (2012): 2496–2505.