Oversetting av ANO-data fra NiN 2.3 NiN 3.0

Ida M. Mienna, Lise Tingstad

15.12.2025

Innholdsfortegnelse

Prosjektinformasjon

Tilgjengelighet: Åpen
Prosjektleder: Lise Tingstad
Forskningsleder: Jørgen Rosvold
Oppdragsgiver: Miljødirektoratet
Oppdragsreferanse: M-3044 - 2025
Kontaktperson hos oppdragsgiver: Lars Lindsø

1. Laste inn og forberede ANO-data

1.1 Last ned ANO-data

ANO-data kan lastes ned fra Miljødirektorates kartkatalog.

Dataene i dette prosjektet er lastet ned som en filgeodatabase (gdb) med siste eksportdato 21.05.2025. Disse dataene inneholder kun data samlet inn mellom 2018-2024.

For å også inkludere data for 2025 fikk vi tilsendt disse dataene fra Miljødirektoratet. De tilsendte dataene inneholdt også bilder av ruter, og fordi dette gjør datasettets størrelse for stort til å kunne lagres på Github, er bildene fjernet fra fila (ikke vist her). Den originale filen var en filgeodatabase, men den er gjort om til en Geopackage (gpkg).

1.2 Last inn data

Før dataene lastes inn må de pakkes ut (ikke vist her).

ANO 2020-2024 er en filgeodatabase (gdb) og ANO 2025 er en geopackage (gpkg). Begge inneholder flere lag hvor det kun er "surveyPoint" som er relevant til videre prosessering.

```
# Last inn 2020-2024 data
ano2024   <- st_read("data/naturovervaking_eksport.gdb", layer="ANO_SurveyPoint")

# Last inn 2025 data
ano2025   <- st_read("data/ANO2025.gpkg", layer="surveypoint")
```

1.3 Forbered data

Begge datasettene består av flere kolonner som ikke er nødvendig å ha med videre. De kolonnene vi trenger er:

- Unik ID for hver observasjon. Hvert ANO-punkt skal ha en unik ID (ano_punkt_id), men pga feilregistreringer i felt vil noen av disse være duplisert. Derfor vil den unike ID-en være globalid som er hver observasjons unike ID. Denne kan brukes for å koble NiN-data mot f.eks. artsfunn i ANO-punktet.
- Kolonner med som viser til NiN kartleggingsenhetkode. Disse er "kartleggingsenhet_1m2", "kartleggingsenhet_250m2".

Vi lager derfor ett nytt datasett hvor de to datasettene først forenkles og deretter slås sammen til ett datasett.

```
valgte_kolonner <- c("globalid", "kartleggingsenhet_1m2", "kartleggingsenhet_250m2")

df_ano2024  <- ano2024 %>%
                # Selekter aktuelle kolonner
                select(all_of(valgte_kolonner)) %>%
                # Fjern geometrien ettersom dette ikke er noe vi trenger å ha med videre
                st_drop_geometry()

df_ano2025  <- ano2025 %>%
                # Selekter aktuelle kolonner
                select(all_of(valgte_kolonner)) %>%
                # Fjern geometrien ettersom dette ikke er noe vi trenger å ha med videre
                st_drop_geometry()

# Slå sammen datasett
df_ano      <- rbind(df_ano2024, df_ano2025)
```

For å kunne oversette dataene først til NiN 2.3 grunntyper (se 2_NiN2.3_KE_til_GT) må de skrives som en kode (f.eks. V1-C-1) og ikke med navn (V1-C-1 svært og temmelig kalkfattige myrflater). Undersøk datasettet nærmere for å sjekke om formatet er standardisert for oversetting til NiN 3.0.

```
unique_vals   <- df_ano %>%
                  select(-globalid) %>%
                  map(unique)
```

Begge NiN kolonnene har varierende skrivemåter. Oppdaterer derfor begge kolonnene slik at de kun inneholder NiN 2.3 kode for kartleggingsenhet (KE).

```r
df_ano        <- df_ano %>%
                 # Fjerner all tekst etter første mellomrom slik at f.eks. "T27-C-3 kalkrik
                 ↪  blokkmark" blir til "T27-C-3"
                   mutate(
                     kartleggingsenhet_1m2 = sub(" .*", "", kartleggingsenhet_1m2),
                     kartleggingsenhet_250m2 = sub(" .*", "", kartleggingsenhet_250m2)
                     )
```

1.4 Eksporter data

Eksporter det prosesserte datasettet og gå videre til neste skript.

```r
df_ano  %>% write.csv("data/df_ano.csv", row.names = F)
```

2. Oversett ANO NiN2.3 kartleggingsenheter til grunntyper

(Forklaring)

2.1 Last inn det forberedede ANO-datasettet

```
df_ano  <- read.csv("data/df_ano.csv")
```

2.2 Oversett fra kartleggingsenhet til grunntyper

For å finne koblingen mellom NiN 2.3 KE og GT bruker vi Artsdatabankens API for NiN 2.3.

Først lager vi en funksjon som henter ut NiN 2.3 GT ved å bruke NiN 2.3 KE-kode.

```
get_grunntyper      <- function(nin_kode) {


  # Lag riktig URL
  url <- paste0(
    "https://nin-kode-api.artsdatabanken.no/v2.3/koder/hentkode/",
    URLencode(paste0("NA ", nin_kode))
  )

  # Hent JSON – prøv, og returner tom vektor hvis feiler
  res <- tryCatch(fromJSON(url), error = function(e) return(NULL))

  if (is.null(res) || length(res$Grunntyper) == 0) {
    return(NA)  # ingen grunntyper funnet
  }
```

```
  res$Grunntyper$Kode$Id

  # Returnér grunntypekode
  return(res$Grunntyper$Kode$Id)

}
```

Siden funksjonen bruker omtrent ett sekund per type kjører vi dette kun per unike type og ikke for hele datasettet.

```
ano_KE            <- df_ano %>%
                     select(kartleggingsenhet_1m2, kartleggingsenhet_250m2) %>%
                     unlist(use.names = FALSE) %>%
                     unique()
```

Deretter kjører vi funksjonen for listen med KE. Dette tar et par minutter.

```
ano_KE_GT         <- tibble(ano_KE = ano_KE) %>%
                     # Kjør funksjonen "get_grunntyper"
                     mutate(ano_GT = purrr::map(ano_KE, get_grunntyper)) %>%
                    # Hent ut alle grunntypene og lag én rad for hver. Dette dupliserer globalid
                     tidyr::unnest(ano_GT)
```

Noen grunntyper ligger ikke inne i API-en og legges derfor til her manuelt.

```
ano_KE_GT <- ano_KE_GT %>%
  mutate(ano_GT = case_when(
    ano_KE == "I1-C-1" ~ "I1-1",
    ano_KE == "T40-C-1" ~ "T40-1",
    ano_KE == "T41-C-1" ~ "T41-1",
    ano_KE == "T42-C-1" ~ "T42-1",
    ano_KE == "T43-C-1" ~ "T43-1",
    ano_KE == "T44-C-1" ~ "T44-1",
    TRUE ~ ano_GT
  ))
```

Koble det oversatte datasettet til df_ano-datasettet. Her blir det mest ryddig å jobbe med 1m2 og 250m2 hver for seg.

```r
df_ano_GT_1m2     <- df_ano %>%
                     # Fjern 250m2 kolonnen
                     select(-kartleggingsenhet_250m2) %>%
                     # Legg til GT for kartleggingsenhet_1m2
                     left_join(ano_KE_GT %>%
                                 # Endre navn slik at de kan slås sammen
                                 rename(kartleggingsenhet_1m2=ano_KE)
                               )

df_ano_GT_250m2   <- df_ano %>%
                     # Fjern 1m2 kolonnen
                     select(-kartleggingsenhet_1m2) %>%
                     # Legg til GT for kartleggingsenhet_250m2
                     left_join(ano_KE_GT %>%
                                 # Endre navn slik at de kan slås sammen
                                 rename(kartleggingsenhet_250m2=ano_KE)
                               )
```

2.3 Eksporter datasettene

```r
df_ano_GT_1m2     %>% write.csv("data/df_ano_GT_1m2.csv", row.names = F)

df_ano_GT_250m2   %>% write.csv("data/df_ano_GT_250m2.csv", row.names = F)
```

3. Oversett ANO NiN 2.3 grunntyper til NiN 3.0 grunntyper

3.1 Hent oversettelsesnøkkel fra Artsdatabanken

Oversettelsesnøkkel mellom NiN 2.3 grunntyper og NiN 3.0 grunntyper hentes fra Artsdatabankens API.

```
# Last ned Excel-fil med oversikt over NiN3
download.file(url = "https://nin-kode-api.artsdatabanken.no/v3.0/rapporter/exceldata",
              destfile = "data/nin3.xlsx")

# Les inn oversettelsesnøkkelen mellom grunntyper
GT_konvertering   <- readxl::read_excel("data/nin3.xlsx", sheet = "GT_Konvertering")
```

3.2 Oversett NiN 2.3 GT til NiN 3.0 GT

For å oversette mellom NiN 2.3 og NiN 3.0 brukes den nedlastede GT_konvertering-tabellen og de to ANO-datasettene for 1m2 og 250m2 nivå.

```
# Funksjon for å oversette
convert_GTs   <- function(df_ano, GT_konvertering) {

  df_ano %>%
    left_join(GT_konvertering %>%
                select(ForrigeKode, Kode),
              by = c("ano_GT" = "ForrigeKode"))
}
```

Kjør funksjonen over på de to datasettene.

```
# Last inn data
df_ano_GT_1m2         <- read.csv("data/df_ano_GT_1m2.csv")
df_ano_GT_250m2       <- read.csv("data/df_ano_GT_250m2.csv")

# Kjør funksjonen på hvert datasett
df_ano_NiN3_GT_1m2    <- convert_GTs(df_ano_GT_1m2, GT_konvertering)
df_ano_NiN3_GT_250m2  <- convert_GTs(df_ano_GT_250m2, GT_konvertering)
```

Undersøk de to oversatte datasettene.

```
df_ano_NiN3_GT    <- df_ano_NiN3_GT_1m2 %>%
                     select(-globalid, -kartleggingsenhet_1m2) %>%
                     bind_rows(
                       df_ano_NiN3_GT_250m2 %>%
                         select(-globalid, -kartleggingsenhet_250m2)
                     ) %>%
                     unique()
```

```
Antall NiN3 GT med NA:  274
```

```
Antall NiN3 GT med ikke-NA:  185
```

Flertallet av grunntypene ble ikke oversatt. Ved nærmere undersøkelse i Artsdatabankens API Excel-ark for konvertering ser man at denne ikke er fullstendig. Må derfor bruke en annen løsning.

3.3 Løsning 2 - bruk oversettelsesnøkkel fra NiN 3.0 systemdokumentasjon

I Halvorsen (2025) Vedlegg 6 finnes det en oversettelsesnøkkel mellom NiN 2.3 og NiN 3 grunntyper. Oversettelsesnøkkelen splitter grunntypekoden i to kolonner (hovedtype (HT) og grunntype (GT)) hvor grunntype kan bestå av flere typer. Vil ha én grunntype per rad med kode i én kolonne. Det er noen NiN 2.3 grunntyper som oversettes til flere hovedtyper, men dette håndteres senere under manuell oversetting.

```
nin3_oversettelsnokkel    <- readxl::read_excel("data/NiN3SD2_Vedlegg.xlsx",
                                                sheet="V6_NATSYST_FRA_2.3_TIL_3.0",
                                                # Hopp over de to første radene
                                                skip = 2)
```

Datasettet må forberedes litt før den kan brukes som oversettelsesnøkkel. HT kan ha flere typer samtidig som GT har det. Dette er i Excel-arket håndtert ved å legge til spesiell whitespace ("vognretur" og linjeskift). Vi bruker dette her for å fordele GT til riktig HT.

```
# Lag funksjon som fordeler GT til HT
pair_safe <- function(ht, gt) {

  n_ht <- length(ht)
  n_gt <- length(gt)

  if (n_ht == n_gt) {
    tibble(HT = ht, GT = gt)
  } else if (n_gt > n_ht) {
    tibble(HT = ht, GT = gt[1:n_ht])     # truncate GT
  } else {
    tibble(HT = ht, GT = c(gt, rep(NA, n_ht - n_gt)))  # pad GT
  }
}

# Bruk funksjonen over til å håndtere det at HT og GT kan ha flere typer i seg hvor disse skal
 ↪  fordeles
nin3_oversettelsnokkel  <- nin3_oversettelsnokkel %>%

                           mutate(
                           # Del opp i flere rader basert på spesiell whitespace ("vognretur" og
                             ↪  linjeskift)
                            HT_split_raw = str_split(HT, "[\\r\\n]+"),
                            GT_split_raw = str_split(GT, "[\\r\\n]+"),

                            # Fjern whitespace
                            HT_split_raw = map(HT_split_raw, ~ str_trim(.x)),
                            GT_split_raw = map(GT_split_raw, ~ str_trim(.x)),

                           # Fjern "'" fra GT. Disse ser ut til å kun brukes for å vise når det
                            ↪  er et skille mellom GT til forskjellige HT
                            GT_split_raw = map(GT_split_raw, ~ gsub("'", "", .x))
                            ) %>%

                           # Kjør funksjonen over for de nye HT og GT kolonnene
                           mutate(pair = map2(HT_split_raw, GT_split_raw, pair_safe)) %>%
                           select(-HT_split_raw, -GT_split_raw) %>%
                           unnest(pair, names_sep = "_paired_") %>%
                           rename(HT_old = HT,
```

```
                                GT_old = GT,
                                HT = pair_paired_HT,
                                GT = pair_paired_GT
                                )

# Fordel hovedtyper over flere rader hvor det er flere enn én hovedtype
nin3_oversettelsnokkel  <- nin3_oversettelsnokkel %>%
                            mutate(
                              # Bytt ut "." etter NA med "-"
                              HT = gsub("\\.", "-", HT),
                              # Fjern "," fra HT
                              HT = gsub(",","",HT)
                              )

# Fordel grunntyper over flere rader hvor det er flere enn én grunntype
nin3_oversettelsnokkel  <- nin3_oversettelsnokkel %>%
                           # Del opp GT i flere rader etter "," eller mellomrom
                           separate_rows(GT, sep = ",|\\s+") %>%
                           # Hvis GT er tom, men ikke NA, legg til NA
                           mutate(GT = ifelse(GT == "", NA, GT)) %>%
                           # Noen GT med verdi mellom 1-9 mangler 0 foran
                           mutate(
                             GT = if_else(str_detect(GT, "^\\d$"),
                                          paste0("0", GT),
                                          GT)
                           )

# Lag kolonne for grunntypekode (HT-GT)
nin3_oversettelsnokkel <- nin3_oversettelsnokkel %>%
                           # Sett sammen HT og GT til én kolonne. Hvis GT er NA, legg til kun HT
                           mutate(ano_GT_NiN3_0 = if_else(is.na(GT),HT,paste0(HT, "-", GT))
                                  ) %>%
                           # Hvis HT er NA, gjør ano_GT_NiN3_0 til NA
                           mutate(ano_GT_NiN3_0 = ifelse(is.na(HT), NA, ano_GT_NiN3_0)
                                  )
```

3.4 Oversett ved bruk av oversettelsesnøkkel

```
# Funksjon for å oversette
convert_GTs_H <- function(df_ano, nin3_oversettelsnokkel) {
```

```
  df_ano %>%
    # Fjern "NA " fra ano_GT
    mutate(ano_GT_NiN2_3 = str_replace_all(ano_GT, "NA ", "")) %>%
    # Legg til oversettelsesnøkkel
    left_join(nin3_oversettelsnokkel %>%
                select(Kode, ano_GT_NiN3_0, HT, GT, FP, SP),
              by = c("ano_GT_NiN2_3" = "Kode"))
}
```

Kjør funksjonen over på de to datasettene.

```
# Last inn data
df_ano_GT_1m2           <- read.csv("data/df_ano_GT_1m2.csv")
df_ano_GT_250m2         <- read.csv("data/df_ano_GT_250m2.csv")

# Kjør funksjonen på hvert datasett
df_ano_NiN3_GT_H_1m2    <- convert_GTs_H(df_ano_GT_1m2, nin3_oversettelsnokkel)
df_ano_NiN3_GT_H_250m2  <- convert_GTs_H(df_ano_GT_250m2, nin3_oversettelsnokkel)
```

Undersøk de to oversatte datasettene.

```
df_ano_NiN3_GT_H        <- df_ano_NiN3_GT_H_1m2 %>%
                             select(-globalid, -kartleggingsenhet_1m2) %>%
                             bind_rows(
                               df_ano_NiN3_GT_H_250m2 %>%
                                 select(-globalid, -kartleggingsenhet_250m2)
                             ) %>%
                             # Ikke inkluder de som er NA i GT ettersom disse håndteres senere
                             filter(!is.na(ano_GT_NiN2_3))%>%
                             unique()
```

```
Antall NiN3 GT med NA:  3
```

```
Antall NiN3 GT med ikke-NA:  482
```

Kun tre typer oversettes ikke automatisk:

```
NiN 2.3 grunntyper som ikke ble oversatt: T27-1 T27-3 T17-1
```

Disse tre typene (T27-1, T27-3 og T17-1) er beskrevet i Halvorsen (2025) for hvorfor de ikke kan oversettes.

3.5 Eksporter resultater

```
df_ano_NiN3_GT_H_1m2    %>% write.csv("data/df_ano_NiN3_GT_H_1m2.csv", row.names = F)
df_ano_NiN3_GT_H_250m2  %>% write.csv("data/df_ano_NiN3_GT_H_250m2.csv", row.names = F)
```

4. Oversett ANO NiN 3.0 grunntyper til kartleggingsenheter

4.1 Last inn ANO-data

```
# Last inn data
df_ano_NiN3_GT_H_1m2    <- read.csv("data/df_ano_NiN3_GT_H_1m2.csv")
df_ano_NiN3_GT_H_250m2  <- read.csv("data/df_ano_NiN3_GT_H_250m2.csv")
```

4.2 Hent oversettelsesnøkkel mellom grunntyper og kartleggingsenheter fra Artsdatabanken

Bruker her samme Excel-ark som lastet ned i kap. 3 (nin3.xlsx), men arkene GT_KLE_M005 og GT_KLE_M020 for å oversette fra NiN 3.0 grunntyper til kartleggingsenheter 1:5000 og 1:20 000.

```
# Les inn oversettelsesnøkkelen mellom grunntyper og kartleggingsenheter
GT_KE5_konvertering   <- readxl::read_excel("data/nin3.xlsx", sheet = "GT_KLE_M005")
GT_KE20_konvertering  <- readxl::read_excel("data/nin3.xlsx", sheet = "GT_KLE_M020")
```

Av en eller annen grunn er flere av radene duplisert, antagelig pga. oversettelsesfeil. Disse filene må derfor kvalitetssikres på forhånd. Vi henter først ut duplikater, men kvalitetssikrer kun for de som er i ANO-datasettet ettersom det er over 80 duplikater (29 i M005 og 53 i M020).

For M005 er det kun verdier for OA02, TA01 og TK01 som er duplisert. De to duplikatene for OA er reelle duplikater. Ettersom TA01 ikke vil ha en god manuell oversettelse senere (se Manuell oversettelser), så fikser vi ikke disse. TK01 er en reell duplikat. Vi gjør derfor ingen endringer i M005-datasettet utenom å fjerne duplikater.

For M020 er det flere verdier for TA01 som er duplisert. Ettersom disse ikke vil ha en god manuell oversettelse senere (se Manuell oversettelser), så fikser vi ikke disse nå. Da står vi igjen med to duplikater. VB01-M020-01 og VB01-09 er feilskriving, hvor M020_kode egentlig

skal være VB01-M020-02. VK02-M020-02 og VK02-04 er en reell duplikat, så denne gjør vi ikke noe med.

```
# M005 ------------------------------------------------------------------------------
GT_KE20_konvertering <- GT_KE20_konvertering %>%
   distinct()

# M020 ------------------------------------------------------------------------------

GT_KE20_konvertering <- GT_KE20_konvertering %>%
  group_by(Grunntype_kode) %>%
  # Legg til en id for dupliserte verdier
  mutate(dup_id = row_number()) %>%
  ungroup() %>%
  # Endre på feil, men kun for den dupliserte verdien
  mutate(
    M020_kode = case_when(
      # VB01-09 skal egentlig også oversettes til VB01-M020-02 og ikke bare VB01-M020-01
      M020_kode == "VB01-M020-01" & Grunntype_kode == "VB01-09" & dup_id == 1 ~ "VB01-M020-02",
      TRUE ~ M020_kode
    )
  ) %>%
  select(-dup_id) %>%
  # Fjern de resterende duplikatene vi valgte å ikke gjøre noe med
  distinct()
```

I tillegg er det noen feilskrivinger i oversettelsene. VB01-10 skal oversettes til VB01-M020-03.

```
# M020 ------------------------------------------------------------------------------

GT_KE20_konvertering <- GT_KE20_konvertering %>%
  # Endrer på feil
  mutate(
    M020_kode = case_when(
      # VB01-10 skal egentlig oversettes til VB01-M020-03
      M020_kode == "VB01-M020-02" & Grunntype_kode == "VB01-10" ~ "VB01-M020-03",
      TRUE ~ M020_kode
    )
  )
```

4.3 Oversett NiN 3.0 grunntyper til kartleggingsenheter

(Forklaring)

Funksjon for å oversette fra grunntyper til kartleggingsenheter.

```
# Funksjon for å oversette
convert_GT_KE <- function(df_ano, GT_KE5_konvertering, GT_KE20_konvertering) {

  df_ano %>%

    # Fjern "NA-" fra ano_GT_NiN3_0
    mutate(ano_GT_NiN3_0 = str_replace_all(ano_GT_NiN3_0, "NA-", "")) %>%

    # Legg til NiN 3.0 M0005-kode (kartleggingsenhet 1:5000)
    left_join(GT_KE5_konvertering,
              by = c("ano_GT_NiN3_0" = "Grunntype_kode")) %>%

    # Legg til NiN 3.0 M0020-kode (kartleggingsenhet 1:20000)
    left_join(GT_KE20_konvertering,
              by= c("ano_GT_NiN3_0" = "Grunntype_kode"))

}
```

Kjør funksjonen over på de to ANO-datasettene med NiN 3.0 grunntyper.

```
# Kjør funksjonen på hvert datasett
df_ano_NiN3_KE_1m2      <- convert_GT_KE(df_ano_NiN3_GT_H_1m2,
                                         GT_KE5_konvertering, GT_KE20_konvertering)
df_ano_NiN3_KE_250m2    <- convert_GT_KE(df_ano_NiN3_GT_H_250m2,
                                         GT_KE5_konvertering, GT_KE20_konvertering)
```

Undersøk de to oversatte datasettene.

```
df_ano_NiN3_KE          <- df_ano_NiN3_KE_1m2 %>%
                             select(-globalid, -ano_GT, -HT, -GT) %>%
                             rename(NiN2_3_KE = kartleggingsenhet_1m2) %>%
                             bind_rows(
                               df_ano_NiN3_KE_250m2 %>%
                                 select(-globalid, -ano_GT, -HT, -GT) %>%
                                 rename(NiN2_3_KE = kartleggingsenhet_250m2)
```

```
                              ) %>%
                              # Ikke inkluder de som er NA i GT ettersom disse håndteres senere
                              filter(!is.na(ano_GT_NiN3_0))%>%
                              unique()
```

```
Antall NiN3 M005 med NA:  10
```

```
Antall NiN3 M005 med ikke-NA:  527
```

```
Antall NiN3 M020 med NA:  10
```

```
Antall NiN3 M020 med ikke-NA:  527
```

Ti typer oversettes ikke.

```
Ikke oversatt til M005: TA01-79 TA01-80 FB03-01 FB01-01 FC01-02 FA02-04 TK01 FA02-08 FA02-10 TD01
```

```
Ikke oversatt til M020: TA01-79 TA01-80 FB03-01 FB01-01 FC01-02 FA02-04 TK01 FA02-08 FA02-10 TD01
```

TK01 og TD01 er hovedtyper og oversettes derfor ikke, men håndteres senere (se kap. 5). TA01-79 og 80, FA02-04, 08 og 10, FB01-01, FB03-01, og FC01-02 eksisterer ikke Artsdatabankens oversettelsesark mellom NiN 3.0 grunntyper og kartleggingsenheter (begge typene).

De to typene som tilhører TA01 og resten som tilhører hovedtypegruppe F (Limniske vannmassesystemer) har ikke kartleggingsenheter for M005 og M020, men kun grunntyper. Disse oversettes derfor kun til grunntyper.

```
F_typer <- df_ano_NiN3_KE %>%
  filter(str_detect(NiN2_3_KE, "F") & str_length(NiN2_3_KE) > 2)

df_ano_NiN3_KE_1m2 <- df_ano_NiN3_KE_1m2 %>%


  mutate(

    # Fikse F-typer
    M005_kode = ifelse(ano_GT_NiN3_0 %in% c(F_typer$ano_GT_NiN3_0,
                                            "TA01-79", "TA01-80"),
                            ano_GT_NiN3_0,
```

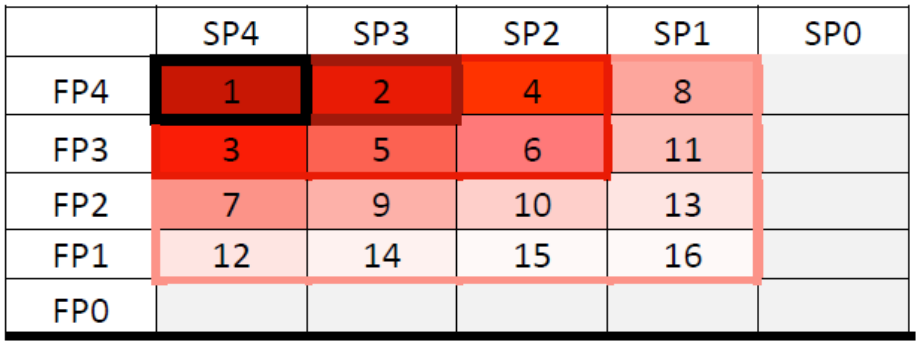
```
                            M005_kode),
    M020_kode = ifelse(ano_GT_NiN3_0 %in% c(F_typer$ano_GT_NiN3_0,
                                            "TA01-79", "TA01-80"),
                            ano_GT_NiN3_0,
                            M020_kode))

df_ano_NiN3_KE_250m2 <- df_ano_NiN3_KE_250m2 %>%
  mutate(M005_kode = ifelse(ano_GT_NiN3_0 %in% c(F_typer$ano_GT_NiN3_0,
                                            "TA01-79", "TA01-80"),
                            ano_GT_NiN3_0,
                            M005_kode),
         M020_kode = ifelse(ano_GT_NiN3_0 %in% c(F_typer$ano_GT_NiN3_0,
                                            "TA01-79", "TA01-80"),
                            ano_GT_NiN3_0,
                            M020_kode))
```

4.4 Eksporter resultater

```
df_ano_NiN3_KE_1m2    %>% write.csv("data/df_ano_NiN3_KE_1m2.csv", row.names = F)
df_ano_NiN3_KE_250m2  %>% write.csv("data/df_ano_NiN3_KE_250m2.csv", row.names = F)
```

5. Bestem usikkerhet i de automatiske oversettelsene fra NiN 2.3 til NiN 3.0

Figur 3 i Halvorsen (2025) viser rangering av oversettelser mellom NiN 2.3 og NiN 3.0 grunntyper med hensyn til oppført følsomhetspresisjon (FP) og spesifiseringsevne (SP). Vi bruker denne for å bestemme usikkerhet i oversettelsene.



	SP4	SP3	SP2	SP1	SP0
FP4	1	2	4	8	
FP3	3	5	6	11	
FP2	7	9	10	13	
FP1	12	14	15	16	
FP0					

Figur 1. Figur 3 i Halvorsen (2025).

5.1 Sett opp rangeringsmatrise som tabell

Vi bruker Fig.3 og beskrivelsene av svært gode til dårlige oversettelser i Halvorsen (2025).

```
# Sett opp Fig.3 i Halvorsen (2025) som tabell
df_usikkerhet <- tibble::tribble(
  ~FP, ~SP, ~rangering,
   4,   4,    1,
   4,   3,    2,
   4,   2,    4,
   4,   1,    8,

   3,   4,    3,
   3,   3,    5,
```

```
   3,   2,    6,
   3,   1,   11,

   2,   4,    7,
   2,   3,    9,
   2,   2,   10,
   2,   1,   13,

   1,   4,   12,
   1,   3,   14,
   1,   2,   15,
   1,   1,   16
)

# Legg til sikkerhet
df_usikkerhet <- df_usikkerhet %>%
  mutate(sikkerhet = case_when(
    rangering == 1                    ~ "Svært god oversettelse", #Svart boks i Figur 1
    rangering == 2                    ~ "God oversettelse",       #Mørkerød boks i Figur 1
    rangering >= 3 &  rangering <= 6  ~ "Akseptabel oversettelse",#Rød boks i Figur 1
    rangering >= 7 &  rangering <= 16 ~ "Dårlig oversettelse"     #Lyserød boks i Figur 1

  ))
```

5.2 Legg til usikkerhet i ANO-data

Last inn ANO data med NiN 3.0 kartleggingsenheter og legg til usikkerhet i datasettene.

```
df_ano_NiN3_KE_1m2      <- read.csv("data/df_ano_NiN3_KE_1m2.csv")
df_ano_NiN3_KE_250m2    <- read.csv("data/df_ano_NiN3_KE_250m2.csv")

df_ano_NiN3_KE_u_1m2    <- df_ano_NiN3_KE_1m2 %>%
                           left_join(df_usikkerhet, by=c("FP", "SP")) %>%
                           select(-rangering)

df_ano_NiN3_KE_u_250m2  <- df_ano_NiN3_KE_250m2 %>%
                           left_join(df_usikkerhet, by=c("FP", "SP")) %>%
                           select(-rangering)
```

5.3 Eksporter resultater

```
df_ano_NiN3_KE_u_1m2    %>% write.csv("data/df_ano_NiN3_KE_u_1m2.csv", row.names = F)
df_ano_NiN3_KE_u_250m2  %>% write.csv("data/df_ano_NiN3_KE_u_250m2.csv", row.names = F)
```

6. Oversettelse av beskrivelsesvariabler

I tillegg til å oversette kartleggingsenheter mellom NiN 2.3 og NiN 3.0 skal også beskrivelsesvariabler oversettes.

6.1 Last inn data

```
# Last inn 2018-2024 data
ano2024   <- st_read("data/naturovervaking_eksport.gdb", layer="ANO_SurveyPoint")

# Last inn 2025 data
ano2025   <- st_read("data/ANO2025.gpkg", layer="surveypoint")
```

I de to datasettene ANO-datasettene finnes disse beskrivelsesvariablene:

```
Beskrivelsesvariabler i ANO 2020-2024: bv_7gr_gi bv_7jb_ba bv_7jb_bt bv_7jb_si bv_7tk bv_7se
```

```
Beskrivelsesvariabler i ANO 2025: bv_7gr_gi bv_7jb_ba bv_7jb_bt bv_7jb_si bv_7tk bv_7se
```

6.2 Oversettelse av beskrivelsesvariabler

Vi bruker Vedlegg 9 i Halvorsen (2025) for å oversette. Her legger vi ikke til oversettelsessikkerhet ettersom oversettelsene generelt er gode.

```
ano2024_bv <- ano2024 %>%
  st_drop_geometry() %>%
    select(globalid, matches("^bv_"))

ano2025_bv <- ano2025 %>%
  st_drop_geometry() %>%
    select(globalid, matches("^bv_"))
```

7GR-GI Grøfting: Grøftingsintensitet

Er det samme som bv_7gr_gi-kolonnene i datasettet. I NiN 3.0 oversettes denne til AD-TE Forventet endringsgjeld etter påvirkning (suksesjonslengde).

Det er en usymmetrisk relasjon med god følsomhet (<) mellom NiN 2.3 og NiN 3.0 for denne variabelen, som betyr at hele NiN 2.3 variabelen fanges opp i NiN 3.0 variabelen. Variabelens trinn er derimot ikke direkte oversettbar mellom systemene ettersom variabelen i NiN 2.3 består av fem trinn og i NiN 3.0 av sju trinn pluss nulltrinn og endetrinn. Vi gjør derfor en oversettelse med forbehold om at det vil være unøyaktigheter i verdiene. Oversettelsene her tar utgangspunkt i beskrivelsene som er av hvert trinn i hvert system (NiN 2.3 og NiN 3.0)

```
bv_var <- c("7GR-GI_1"="AD-TE_0", # Intakt / Ingen endringsgjeld
            "7GR-GI_2"="AD-TE_a", # Ubetydelig / Ubetydelig endringsgjeld
            "7GR-GI_3"="AD-TE_c", # Nokså lite grøftingsinngrep / Betydelig endringsgjeld
            "7GR-GI_4"="AD-TE_d", # Omfattende grøfting / Vesentlig endringsgjeld
            "7GR-GI_5"="AD-TE_e", # Gjennomgripende grøfting / Temmelig stor endringsgjeld
            "7GR-GI_X"=NA) # Verdien for ikke registrert eksisterer ikke i NiN 3.0

ano2024_bv <- ano2024_bv %>%
  mutate(bv_ad_te = unname(bv_var[bv_7gr_gi]))
ano2025_bv <- ano2025_bv %>%
  mutate(bv_ad_te = unname(bv_var[bv_7gr_gi]))
```

7JB-BA Jordbruk: Aktuell bruksintensitet

Er det samme som bv_7jb_ba-kolonnene i datasettet. I NiN 3.0 oversettes denne til KM-AH Nåtidig høstingsintensitet.

Det er en inkongruent relasjon ( ) mellom NiN 2.3 og NiN 3.0 for denne variabelen. 7JB-BA er ikke videreført i NiN 3.0, og oversettes derfor ikke.

7JB-BT Jordbruk: Beitetrykk

Er det samme som bv_7jb_bt-kolonnene i datasettet. I NiN 3.0 oversettes denne til KM-BT Aktuelt beitetrykk.

Det er en kongruent relasjon (=) mellom NiN 2.3 og NiN 3.0 for denne variabelen, men verdiene er endret. Vi gjør derfor en direkte oversettelse med modifikasjon.

```
bv_var <- c("7JB-BT_1"="7KM-BT_0",
            "7JB-BT_2"="7KM-BT_1",
            "7JB-BT_3"="7KM-BT_2",
            "7JB-BT_4"="7KM-BT_3",
            "7JB-BT_5"="7KM-BT_4",
            "7JB-BT_6"="7KM-BT_y")

ano2024_bv <- ano2024_bv %>%
  mutate(bv_km_bt = unname(bv_var[bv_7jb_bt]))
ano2025_bv <- ano2025_bv %>%
  mutate(bv_km_bt = unname(bv_var[bv_7jb_bt]))
```

7JB-SI Jordbruk: Slåtteintensitet

Er det samme som bv_7jb_si-kolonnene i datasettet. 7JB-SI er ikke videreført i NiN 3.0, og oversettes derfor ikke.

7TK Spor etter ferdsel med tunge kjøretøy

Er det samme som bv_7tk-kolonnene i datasettet. I NiN 3.0 oversettes denne til KM-TK Ferdsel med tunge kjøretøy.

Det er en kongruent relasjon mellom NiN 2.3 og NiN 3.0 for denne variabelen. Vi oversetter derfor 7SE kolonnen direkte til KM-TK.

```
ano2024_bv <- ano2024_bv %>%
  mutate(bv_km_tk = str_replace(bv_7tk, "^7TK_", "KM-TK_"))
ano2025_bv <- ano2025_bv %>%
  mutate(bv_km_tk = str_replace(bv_7tk, "^7TK_", "KM-TK_"))
```

7SE Spor etter slitasje og slitasjebetinget erosjon

Er det samme som bv_7se-kolonnene i datasettet. I NiN 3.0 oversettes denne til KM-SE Slitasjebetinget erosjon.

Det er en kongruent relasjon mellom NiN 2.3 og NiN 3.0 for denne variabelen. Vi oversetter derfor 7SE kolonnen direkte til KM-SE.

```r
ano2024_bv <- ano2024_bv %>%
  mutate(bv_km_se = str_replace(bv_7se, "^7SE_", "KM-SE_"))
ano2025_bv <- ano2025_bv %>%
  mutate(bv_km_se = str_replace(bv_7se, "^7SE_", "KM-SE_"))
```

6.3 Eksporter resultater

```r
ano2024_bv %>% write.csv("data/ano2024_bv.csv", row.names = F)
ano2025_bv %>% write.csv("data/ano2025_bv.csv", row.names = F)
```

7. Manuelle oversettelser mellom NiN 2.3 og NiN 3.0

Det er flere NiN 2.3 som oversettes til flere NiN 3.0 KE. For flere av disse kan andre data (artslister og registrerte variabler slik som tresjiktsdekning) brukes til å bedømme hvilken KE som er mest riktig.

7.1 Hent inn data

ANO-data

Oversatte data.

```
df_ano_NiN3_KE_u_1m2 <- read.csv("data/df_ano_NiN3_KE_u_1m2.csv")
df_ano_NiN3_KE_u_250m2 <- read.csv("data/df_ano_NiN3_KE_u_250m2.csv")
```

Originale data hvor oversettelsene skal legges til.

```
# Last inn 2020-2024 data
ano2024   <- st_read("data/naturovervaking_eksport.gdb", layer="ANO_SurveyPoint")

# Last inn 2025 data
ano2025   <- st_read("data/ANO2025.gpkg", layer="surveypoint")
```

Hent også artsdata i vegetasjonsruteregistreringene.

```
ano_art2024   <- st_read("data/naturovervaking_eksport.gdb", layer="ANO_Art")
ano_art2025   <- st_read("data/ANO2025.gpkg", layer="art")
```

Vi kommer her også til å trenger Vedlegg 6 i Halvorsen (2025).

```r
nin3_oversettelsnokkel    <- readxl::read_excel("data/NiN3SD2_Vedlegg.xlsx",
                                                sheet="V6_NATSYST_FRA_2.3_TIL_3.0",
                                                # Hopp over de to første radene
                                                skip = 2)
```

Artslister

Henter inn artslister som viser artsmengder langs forskjellige LKM-er hvor disse kan brukes til å skille mellom ulike typer langs gradientene. OBS! Listene er ikke komplett og kan ikke alltid oversettes direkte til alle typer.

Kalkinnhold (KA) fastmark

Listen for arter langs KA hentes fra et generalisert artslistedatasett (GAD) for arter langs UF+KA. Denne listen ligger ikke åpent tilgjengelig, men er delt av Rune Halvorsen over epost. Vi trenger kun KA-informasjon om artene, og henter dette ut fra lista som kombinerer UF og KA sammen til tallkombinasjoner hvor tallene representerer trinn i LKM-en (a=1, b=2, c=3, d=4, e=5, f=6, g=7, h=8). For å hente ut KA per art tar vi gjennomsnitt av alle kolonner hvor tallverdien til høyre er lik (f.eks 11, 21 og 31 har samme kalktrinn). For å unngå skjevhet i verdiene tar vi kun gjennomsnitt for de verdiene som ikke er 0.

```r
KA_arter <- readxl::read_excel("data/GAD_T4_KA_UF.xlsx", skip = 1)

# 1. Identify numeric-looking columns
num_cols <- names(KA_arter)[str_detect(names(KA_arter), "^\\d+$")]

# 2. Coerce these columns to numeric
KA_arter_clean <- KA_arter %>%
  mutate(across(all_of(num_cols), ~ suppressWarnings(as.numeric(.x))))

# 3. Compute averages by rightmost digit
KA_arter <- KA_arter_clean %>%
  pivot_longer(
    cols = all_of(num_cols),
    names_to = "column",
    values_to = "value"
  ) %>%
  mutate(digit = str_sub(column, -1)) %>%
  group_by(NyRad, digit) %>%
  summarise(
    avg_value = mean(value[value != 0], na.rm = TRUE),
    .groups = "drop"
```

```
  ) %>%
  pivot_wider(
    names_from = digit,
    values_from = avg_value,
    names_prefix = "avg_"
  ) %>%
  mutate(across(starts_with("avg_"), ~ ifelse(is.nan(.x), 0, .x))) %>%
  rename(
    KA_a = avg_1,
    KA_b = avg_2,
    KA_c = avg_3,
    KA_d = avg_4,
    KA_e = avg_5,
    KA_f = avg_6,
    KA_g = avg_7,
    KA_h = avg_8
  ) %>%
  left_join(KA_arter_clean, by = "NyRad")
```

Kalkinnhold (KA) våtmark

En annen liste er spesifikk for KA i våtmark (KA_V_arter). Her er KA basistrinnene slått sammen til trinn som ikke matcher trinn i KE-er. For å håndtere dette antar vi her at basistrinnene har samme verdi som det sammenslåtte trinnet (f.eks. når bc=1 er b=1 og c=1).

```
KA_V_arter <-
↪  readxl::read_excel("data/Artstabeller_for_variasjon_langs_viktige_LKM_(Excel_format).xlsx",
↪  sheet="x7VKA", skip=2) %>%
  # Legg til kolonnenavn
  rename_with(~ c("id", "art", "NorskNavn", "KA0", "a","bc","de","fg","hi"), .cols = 1:9) %>%
  select(id:hi) %>%
  # Splitt KA kolonnene med flere basistrinn til en kolonne per basistrinn
  mutate(b = bc,
         c = bc,
         d = de,
         e = de,
         f = fg,
         g = fg,
         h = hi,
         i = hi) %>%
  # Fjern bc til hi kolonnene
  select(-c(bc:hi))
```

Vannmetning (VM) fastmark

Det er flere typer i NiN 3.0 som defineres av vannmetning (VM) og samtidig ikke var det i NiN 2.3 (se f.eks. T4 og T31). Vi trenger derfor en artstabell som viser til fuktkrevende arter. Vi bruker samme artstabell som for KA fastmark, bare at her henter vi ut uLKM-en for VM hvor vi anser arter med +-verdier til å være fuktkrevende. Vi antar i oversettelsene at tilstedeværelsen av en art med +-tegn indikerer at typen har høy nok vannmetning til å være en fukttype.

```
VM_arter <- readxl::read_excel("data/GAD_T4_KA_UF.xlsx", skip = 1) %>%
  # Fjern alle kolonner bortsett fra Art, NorskNavn og VM
  select(Art, NorskNavn, VM) %>%
  # Filtrer ut de radene hvor det er +-tegn
  filter(str_detect(VM,"\\+"))
```

Tørrleggingsvarighet (TV) våtmark

Denne listen er spesifikk for tørrleggingsvarighet (TV) i våtmark.

```
TV_V_arter <-
 ↪  readxl::read_excel("data/Artstabeller_for_variasjon_langs_viktige_LKM_(Excel_format).xlsx",
 ↪  sheet="x8VTV", skip=2) %>%
  # Legg til kolonnenavn
  rename_with(~ c("id", "art", "NorskNavn", "cd", "ef","gh","ij","k"), .cols = 1:8) %>%
  select(id:k)
```

7.2 Klargjør datasett til manuell vurdering

De oversatte dataene må klargjøres før de legges til de to ANO datasettene.

```
# Sett sammen 1 m2 og 250 m2
df_ano_NiN3_KE_u <- df_ano_NiN3_KE_u_1m2 %>%
  select(-globalid) %>%
  rename(KE_5K_NiN2_3 = kartleggingsenhet_1m2) %>%
  bind_rows(df_ano_NiN3_KE_u_250m2 %>%
              select(-globalid) %>%
               rename(KE_5K_NiN2_3 = kartleggingsenhet_250m2)) %>%
  # Fjern grunntyper ettersom det kan være flere av disse, men kun én KE på hver side
  select(-sikkerhet) %>%
```

```
  distinct()

df_ano_NiN3_KE_u_1m2 <- df_ano_NiN3_KE_u_1m2 %>%
  # Fjern kolonner som ikke er nødvendig å ha med videre
  select(-c(ano_GT, HT, GT)) %>%
  # Flytt FP og SP før sikkerhet
  relocate(FP, .before = sikkerhet) %>%
  relocate(SP, .before = sikkerhet) %>%
  # Legg til at de oversatte kolonnene gjelder for 1m2
  rename_with(
    ~ paste0(.x, "_1m2"),
    .cols = -c(globalid, kartleggingsenhet_1m2)
  ) %>%
  # Legg til i kolonnenavnet at KE-kolonnen har blitt oppdatert (kan ikke slås sammen med
  ↪  ANO-datasett slik det er)
  rename(kartleggingsenhet_1m2_oppd = kartleggingsenhet_1m2)

df_ano_NiN3_KE_u_250m2 <- df_ano_NiN3_KE_u_250m2 %>%
  # Fjern kolonner som ikke er nødvendig å ha med videre
  select(-c(ano_GT, HT, GT)) %>%
  # Flytt FP og SP før sikkerhet
  relocate(FP, .before = sikkerhet) %>%
  relocate(SP, .before = sikkerhet) %>%
  # Legg til at de oversatte kolonnene gjelder for 1m2
  rename_with(
    ~ paste0(.x, "_250m2"),
    .cols = -c(globalid, kartleggingsenhet_250m2)
  ) %>%
  # Legg til i kolonnenavnet at KE-kolonnen har blitt oppdatert (kan ikke slås sammen med
  ↪  ANO-datasett slik det er)
  rename(kartleggingsenhet_250m2_oppd = kartleggingsenhet_250m2)
```

Legg til de to datasettene med oversettelser til de to ANO-datasettene.

```
ano2024_auto_oversatt <- ano2024 %>%
  left_join(df_ano_NiN3_KE_u_1m2, by="globalid") %>%
  left_join(df_ano_NiN3_KE_u_250m2, by="globalid")

ano2025_auto_oversatt <- ano2025 %>%
  left_join(df_ano_NiN3_KE_u_1m2, by="globalid") %>%
  left_join(df_ano_NiN3_KE_u_250m2, by="globalid")
```

Her fjerner vi dupliserte globalid som kommer fra kodelinjene over ved å oppsummere rader med flere verdier. For eksempel vil M005_kode_1m2 som sier hvilke M005-typer NiN 2.3 KE-en er oversatt til gå fra å ha en type per rad til at disse oppsummeres i samme rad.

```
ano_type_list   <- list()

# Sett geometrien til sides fordi de da går fortere
ano2024_auto_oversatt_geo <- ano2024_auto_oversatt %>%
  select(globalid, shape) %>%
  unique()
ano2025_auto_oversatt_geo <- ano2025_auto_oversatt %>%
  select(globalid, geom) %>%
  unique()


# 2024 ------------------------------------------------------------------------------
ano_type_list$type2024 <- ano2024_auto_oversatt %>%
  # Fjern geometri midlertidig
  st_drop_geometry() %>%
  # Fjern dupliserte globalid ved å slå sammen rader til én rad og oppsummer verdiene.
  group_by(globalid) %>%
  summarise(across(
    .cols = everything(),
    .fns = ~ {
      # unique values
      vals <- unique(.x)
      # collapse to comma-separated string
      paste(vals, collapse = ", ")
    },
    .names = "{.col}"
  ), .groups = "drop") %>%
  # Gjør om tekst-NA til faktiske NA-verdier og manglende verdier "" til NA
  mutate(across(everything(), ~ ifelse(.x %in% c("", "NA"), NA, .x))) %>%
  # Legg til kolonne for manuelle oversettelser
  mutate(M005_mo_1m2 = as.character(NA),
         M020_mo_1m2 = as.character(NA),
         M005_mo_250m2 = as.character(NA),
         M020_mo_250m2 = as.character(NA)
         ) %>%
  # Legg til geometrien
  left_join(ano2024_auto_oversatt_geo, by="globalid") %>%
  st_as_sf(sf_column_name = "shape")
```

```r
# 2025 ------------------------------------------------------------------------------

ano_type_list$type2025 <- ano2025_auto_oversatt %>%
  # Fjern geometri midlertidig
  st_drop_geometry() %>%
  # Fjern dupliserte globalid ved å slå sammen rader til én rad og oppsummer verdiene.
  group_by(globalid) %>%
  summarise(across(
    .cols = everything(),
    .fns = ~ {
      # unique values
      vals <- unique(.x)
      # collapse to comma-separated string
      paste(vals, collapse = ", ")
    },
    .names = "{.col}"
  ), .groups = "drop") %>%
  # Gjør om tekst-NA til faktiske NA-verdier og manglende verdier "" til NA
  mutate(across(everything(), ~ ifelse(.x %in% c("", "NA"), NA, .x))) %>%
  # Legg til kolonne for manuelle oversettelser
  mutate(M005_mo_1m2 = as.character(NA),
         M020_mo_1m2 = as.character(NA),
         M005_mo_250m2 = as.character(NA),
         M020_mo_250m2 = as.character(NA)
         ) %>%
  # Legg til geometrien
  left_join(ano2025_auto_oversatt_geo, by="globalid") %>%
  st_as_sf(sf_column_name = "geom")
```

Hent også artsdata for de observasjonene som skal til manuell vurdering ettersom disse vil være relevante. Her trengs ikke geometrien så dette fjernes.

```r
ano_art_list   <- list()

ano_art_list$art2024 <- ano_art2024 %>%
  # Fjern geometri
  st_drop_geometry()

ano_art_list$art2025 <- ano_art2025 %>%
  # Fjern geometri
  st_drop_geometry()
```

7.2 Oversikt over oversettelsene mellom NiN 2.3 og NiN 3.0

Vurderingene gjøres ikke per observasjon, men per unike NiN 2.3 KE med flere NiN 3.0 KE-er. Vi lager derfor først en oversikt over de som har fått flere enn én NiN 3.0 KE (enten M005 eller M020), og bruker denne for å vite hvilke som skal oversettes manuelt.

```
n_KE_5K_NiN2_3 <- df_ano_NiN3_KE_u %>%
  select(KE_5K_NiN2_3, M005_kode) %>%
  unique() %>%
  # Fjern de hvor M005_kode er NA
  filter(!is.na(M005_kode)) %>%
  group_by(KE_5K_NiN2_3) %>%
  summarise(
    M005_kode_list = paste(sort(unique(na.omit(M005_kode))),
                           collapse = ", "),
    n_M005 = n()
  )

n_KE_20K_NiN2_3 <- df_ano_NiN3_KE_u %>%
  select(KE_5K_NiN2_3, M020_kode) %>%
  unique() %>%
  # Fjern de hvor M020_kode er NA
  filter(!is.na(M020_kode)) %>%
  group_by(KE_5K_NiN2_3) %>%
  summarise(
    # Lager en liste med kodene for NiN 3.0 i alfabetisk rekkefølge
    M020_kode_list = paste(sort(unique(na.omit(M020_kode))),
                           collapse = ", "),
    n_M020 = n()
  )

n_KE_NiN2_3 <- full_join(n_KE_5K_NiN2_3, n_KE_20K_NiN2_3) %>%
  # Kun de hvor M005 eller M020 har flere typer
  filter(n_M005 > 1 | n_M020 > 1)
```

7.3 Funksjoner til manuelle oversettelser

Flere av oversettelsene har de samme mønstrene i hvrodan de skal oversettes manuelt. Funksjonene under brukes for disse.

```
# Filter for expressions
make_filter_expr <- function(rules) {
  # Extract RHS of formulas (~ a >= 1 → a >= 1)
  exprs <- lapply(rules, rlang::f_rhs)

  # Kombiner expressions med AND
  Reduce(function(x, y) rlang::expr((!!x) & (!!y)), exprs)
}

# For å kunne sette inn verdier som kolonner eller faktiske verdier
resolve_val <- function(val, df) {
  if (is.null(val)) return(NULL)

  if (rlang::is_formula(val)) {
    col <- rlang::as_name(rlang::f_rhs(val))
    return(df[[col]])
  }

  if (rlang::is_symbol(val)) {
    col <- rlang::as_name(val)
    return(df[[col]])
  }

  if (is.character(val)) {
    return(val)
  }

  stop("Ikke støttet")
}
```

Funksjoner for å oversette ved hjelp av KA. Disse funksjonene gjelder både for KA fastmark og våtmark. For typer registrert i 1 m2 vegetasjonsruter finner vi først de typene som inneholder KA-arter fra artslistereglene (se 7.4 Regler for artslister) og registrer de som M005_high/M020_high. Deretter setter vi de som ikke matcher artslistereglene som M005_low/M020_low. For typer registrert i 250 m2 sirkel setter vi typen til å være den samme som 1 m2 hvis KE er den samme. Er den ikke det setter vi typen til å være M005_low/M020_low.

```
## M005 ------------------------------------------------------------------------
sett_M005_KA <- function(ano_df, ano_art_df,
                         KE_code,
                         M005_high, M005_low,
```

```
                        KA_arter_df,
                        KA_rules) {

  # Fiks verdier om de er kolonnenavn eller verdier
  M005_high_val <- resolve_val(M005_high, ano_df)
  M005_low_val  <- resolve_val(M005_low,  ano_df)

  # Sett opp filter
  cond_expr <- make_filter_expr(KA_rules)

  # Finn relevante globalid med KE
  id <- ano_df %>%
    sf::st_drop_geometry() %>%
    dplyr::filter(!is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE_code)
↪  %>%
    dplyr::distinct(globalid)

  # Hent ut arter som matcher filter
  KA_utvalgte <- KA_arter_df %>%
    dplyr::filter(!!cond_expr)

  # Standardiser artsnavn-kolonne
  navnkol <- intersect(c("art_norsk_navn", "norsk_navn"), names(ano_art_df))
  if (length(navnkol) == 1) {
    ano_art_df <- ano_art_df %>% dplyr::rename(navn_brukt = dplyr::all_of(navnkol))
  } else {
    ano_art_df$navn_brukt <- NA_character_
  }

  # Finn artsobservasjoner som matcher KA-artene
  utvalgte <- ano_art_df %>%
    dplyr::filter(parentglobalid %in% id$globalid) %>%
    dplyr::filter(navn_brukt %in% KA_utvalgte$NorskNavn)

  # Legg til typene
  ano_df %>%
    dplyr::mutate(
      # 1 m2 ------------------------------------------------------------
      M005_mo_1m2 = dplyr::case_when(
        globalid %in% utvalgte$parentglobalid ~ M005_high_val,

        !is.na(kartleggingsenhet_1m2_oppd) &
          kartleggingsenhet_1m2_oppd %in% KE_code &
```

```
          is.na(M005_mo_1m2) ~ M005_low_val,

        TRUE ~ M005_mo_1m2
      ),

      # 250 m2 ------------------------------------------------------------
      M005_mo_250m2 = dplyr::case_when(
        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code &
          !is.na(kartleggingsenhet_1m2_oppd) &
          kartleggingsenhet_1m2_oppd %in% KE_code ~ M005_mo_1m2,

        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code &
          (is.na(kartleggingsenhet_1m2_oppd) | !(kartleggingsenhet_1m2_oppd %in% KE_code)) ~
↪  M005_low_val,

        TRUE ~ M005_mo_250m2
      ),

      # sikkerhet ---------------------------------------------------------
      sikkerhet_mo_1m2 = dplyr::case_when(
        globalid %in% utvalgte$parentglobalid ~ "God oversettelse",

        !is.na(kartleggingsenhet_1m2_oppd) &
          kartleggingsenhet_1m2_oppd %in% KE_code ~ "God oversettelse",

        TRUE ~ sikkerhet_mo_1m2
      ),

      sikkerhet_mo_250m2 = dplyr::case_when(
        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code ~ "Akseptabel oversettelse",

        TRUE ~ sikkerhet_mo_250m2
      )
    )
}



## M020 ----------------------------------------------------------------------------
```

```r
sett_M020_KA <- function(ano_df, ano_art_df,
                         KE_code,
                         M020_high, M020_low,
                         KA_arter_df,
                         KA_rules) {

  # Fiks verdier om de er kolonnenavn eller verdier
  M020_high_val <- resolve_val(M020_high, ano_df)
  M020_low_val  <- resolve_val(M020_low,  ano_df)

  # Sett opp filter
  cond_expr <- make_filter_expr(KA_rules)

  # Finn relevante globalid med KE
  id <- ano_df %>%
    sf::st_drop_geometry() %>%
    dplyr::filter(!is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE_code)
↪  %>%
    dplyr::distinct(globalid)

  # Hent ut arter som matcher filter
  KA_utvalgte <- KA_arter_df %>%
    dplyr::filter(!!cond_expr)

  # Standardiser artsnavn-kolonne
  navnkol <- intersect(c("art_norsk_navn", "norsk_navn"), names(ano_art_df))
  if (length(navnkol) == 1) {
    ano_art_df <- ano_art_df %>% dplyr::rename(navn_brukt = dplyr::all_of(navnkol))
  } else {
    ano_art_df$navn_brukt <- NA_character_
  }

  # Finn artsobservasjoner som matcher KA-artene
  utvalgte <- ano_art_df %>%
    dplyr::filter(parentglobalid %in% id$globalid) %>%
    dplyr::filter(navn_brukt %in% KA_utvalgte$NorskNavn)

  # Legg til typene
  ano_df %>%
    dplyr::mutate(
      # 1 m2 ------------------------------------------------------------
      M020_mo_1m2 = dplyr::case_when(
        globalid %in% utvalgte$parentglobalid ~ M020_high_val,
```

```
        !is.na(kartleggingsenhet_1m2_oppd) &
          kartleggingsenhet_1m2_oppd %in% KE_code &
          is.na(M020_mo_1m2) ~ M020_low_val,

        TRUE ~ M020_mo_1m2
      ),

      # 250 m2 ------------------------------------------------------------
      M020_mo_250m2 = dplyr::case_when(
        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code &
          !is.na(kartleggingsenhet_1m2_oppd) &
          kartleggingsenhet_1m2_oppd %in% KE_code ~ M020_mo_1m2,

        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code &
          (is.na(kartleggingsenhet_1m2_oppd) | !(kartleggingsenhet_1m2_oppd %in% KE_code)) ~
↪  M020_low_val,

        TRUE ~ M020_mo_250m2
      )
    )
}



# Funksjoner for å kjøre over 2024 og 2025 sammen
sett_M005_KA_multi <- function(ano_df_list,
                               ano_art_df_list,
                               KE_code,
                               M005_high, M005_low,
                               KA_arter_df,
                               KA_rules) {

  purrr::map2(
    ano_df_list,
    ano_art_df_list,
    ~ sett_M005_KA(
        ano_df = .x,
        ano_art_df = .y,
        KE_code = KE_code,
        M005_high = M005_high,
```

```
        M005_low = M005_low,
        KA_arter_df = KA_arter_df,
        KA_rules = KA_rules
      )
  )
}



sett_M020_KA_multi <- function(ano_df_list,
                               ano_art_df_list,
                               KE_code,
                               M020_high, M020_low,
                               KA_arter_df,
                               KA_rules) {

  purrr::map2(
    ano_df_list,
    ano_art_df_list,
    ~ sett_M020_KA(
        ano_df = .x,
        ano_art_df = .y,
        KE_code = KE_code,
        M020_high = M020_high,
        M020_low = M020_low,
        KA_arter_df = KA_arter_df,
        KA_rules = KA_rules
      )
  )
}
```

Funksjoner for å filtrere for vannmetning (VM). For typer registrert i 1 m2 vegetasjonsruter finner vi først de typene som inneholder VM-arter fra artslistereglene (se 7.4 Regler for artslister) og registrer de som M005_high/M020_high. Deretter setter vi de som ikke matcher artslistereglene som M005_low_val/M020_low_val. For typer registrert i 250 m2 sirkel setter vi typen til å være den samme som 1 m2 hvis KE er den samme. Er den ikke det setter vi typen til å være M005_low_val/M020_low_val.

```
## M005 ------------------------------------------------------------------------------
sett_M005_VM <- function(ano_df, ano_art_df,
                         KE_code,
                         M005_high, M005_low,
```

```
                      VM_arter_df) {

M005_high_val <- resolve_val(M005_high, ano_df)
M005_low_val  <- resolve_val(M005_low,  ano_df)

id <- ano_df %>%
  sf::st_drop_geometry() %>%
  dplyr::filter(!is.na(kartleggingsenhet_1m2_oppd) &
                kartleggingsenhet_1m2_oppd %in% KE_code) %>%
  dplyr::distinct(globalid)

navnkol <- intersect(c("art_norsk_navn", "norsk_navn"), names(ano_art_df))
if (length(navnkol) == 1) {
  ano_art_df <- ano_art_df %>% dplyr::rename(navn_brukt = dplyr::all_of(navnkol))
} else {
  ano_art_df$navn_brukt <- NA_character_
}

utvalgte <- ano_art_df %>%
  dplyr::filter(parentglobalid %in% id$globalid) %>%
  dplyr::filter(navn_brukt %in% VM_arter_df$NorskNavn)

ano_df %>%
  dplyr::mutate(
    # 1 m2 ------------------------------------------------------------
    M005_mo_1m2 = dplyr::case_when(
      globalid %in% utvalgte$parentglobalid ~ M005_high_val,

      !is.na(kartleggingsenhet_1m2_oppd) &
        kartleggingsenhet_1m2_oppd %in% KE_code &
        is.na(M005_mo_1m2) ~ M005_low_val,

      TRUE ~ M005_mo_1m2
    ),

    sikkerhet_mo_1m2 = dplyr::case_when(
      globalid %in% utvalgte$parentglobalid ~ "God oversettelse",

      !is.na(kartleggingsenhet_1m2_oppd) &
        kartleggingsenhet_1m2_oppd %in% KE_code ~ "God oversettelse",

      TRUE ~ sikkerhet_mo_1m2
    ),
```

```
      # 250 m2 ------------------------------------------------------------
      M005_mo_250m2 = dplyr::case_when(
        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code &
          !is.na(kartleggingsenhet_1m2_oppd) &
          kartleggingsenhet_1m2_oppd %in% KE_code ~ M005_mo_1m2,

        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code &
          (is.na(kartleggingsenhet_1m2_oppd) |
           !(kartleggingsenhet_1m2_oppd %in% KE_code)) ~ M005_low_val,

        TRUE ~ M005_mo_250m2
      ),

      sikkerhet_mo_250m2 = dplyr::case_when(
        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code ~ "Akseptabel oversettelse",

        TRUE ~ sikkerhet_mo_250m2
      )
    )
}



## M020 ------------------------------------------------------------------------------

sett_M020_VM <- function(ano_df, ano_art_df,
                         KE_code,
                         M020_high, M020_low,
                         VM_arter_df) {

  M020_high_val <- resolve_val(M020_high, ano_df)
  M020_low_val  <- resolve_val(M020_low,  ano_df)

  id <- ano_df %>%
    sf::st_drop_geometry() %>%
    dplyr::filter(!is.na(kartleggingsenhet_1m2_oppd) &
                  kartleggingsenhet_1m2_oppd %in% KE_code) %>%
    dplyr::distinct(globalid)
```

```
  navnkol <- intersect(c("art_norsk_navn", "norsk_navn"), names(ano_art_df))
  if (length(navnkol) == 1) {
    ano_art_df <- ano_art_df %>% dplyr::rename(navn_brukt = dplyr::all_of(navnkol))
  } else {
    ano_art_df$navn_brukt <- NA_character_
  }

  utvalgte <- ano_art_df %>%
    dplyr::filter(parentglobalid %in% id$globalid) %>%
    dplyr::filter(navn_brukt %in% VM_arter_df$NorskNavn)

  ano_df %>%
    dplyr::mutate(
      # 1 m2 ------------------------------------------------------------
      M020_mo_1m2 = dplyr::case_when(
        globalid %in% utvalgte$parentglobalid ~ M020_high_val,

        !is.na(kartleggingsenhet_1m2_oppd) &
          kartleggingsenhet_1m2_oppd %in% KE_code &
          is.na(M020_mo_1m2) ~ M020_low_val,

        TRUE ~ M020_mo_1m2
      ),

      # 250 m2 ----------------------------------------------------------
      M020_mo_250m2 = dplyr::case_when(
        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code &
          !is.na(kartleggingsenhet_1m2_oppd) &
          kartleggingsenhet_1m2_oppd %in% KE_code ~ M020_mo_1m2,

        !is.na(kartleggingsenhet_250m2_oppd) &
          kartleggingsenhet_250m2_oppd %in% KE_code &
          (is.na(kartleggingsenhet_1m2_oppd) |
           !(kartleggingsenhet_1m2_oppd %in% KE_code)) ~ M020_low_val,

        TRUE ~ M020_mo_250m2
      )
    )
}
```

```r
# Funksjoner for å kjøre over 2024 og 2025 sammen
sett_M005_VM_multi <- function(ano_df_list,
                               ano_art_df_list,
                               KE_code,
                               M005_high, M005_low,
                               VM_arter_df) {

  purrr::map2(
    ano_df_list,
    ano_art_df_list,
    ~ sett_M005_VM(
        ano_df = .x,
        ano_art_df = .y,
        KE_code = KE_code,
        M005_high = M005_high,
        M005_low = M005_low,
        VM_arter_df = VM_arter_df
      )
  )
}



sett_M020_VM_multi <- function(ano_df_list,
                               ano_art_df_list,
                               KE_code,
                               M020_high, M020_low,
                               VM_arter_df) {

  purrr::map2(
    ano_df_list,
    ano_art_df_list,
    ~ sett_M020_VM(
        ano_df = .x,
        ano_art_df = .y,
        KE_code = KE_code,
        M020_high = M020_high,
        M020_low = M020_low,
        VM_arter_df = VM_arter_df
      )
  )
}
```

For å filtrere for tørrleggingsvarighet (TV). For typer registrert i 1 m2 vegetasjonsruter finner vi først de typene som inneholder TV-arter fra de to artslisteregelsettene (se 7.4 Regler for artslister) og registrer de som enten M005_high/M020_high eller M005_low/M020_low. Deretter setter vi de som ikke matcher artslistereglene som M005_high/M020_high. For typer registrert i 250 m2 sirkel setter vi typen til å være den samme som 1 m2 hvis KE er den samme. Er den ikke det setter vi typen til å være M005_high/M020_high.

```
# M005 ------------------------------------------------------------------------------
sett_M005_TV_for_KE <- function(ano_df, ano_art_df,
                                KE_code,
                                M005_high, M005_low,
                                TV_rules_high, TV_rules_low,
                                TV_V_arter_df) {

  high_expr <- make_filter_expr(TV_rules_high)
  low_expr  <- make_filter_expr(TV_rules_low)

  id <- ano_df %>%
    sf::st_drop_geometry() %>%
    dplyr::filter(!is.na(kartleggingsenhet_1m2_oppd) &
                  kartleggingsenhet_1m2_oppd %in% KE_code) %>%
    dplyr::distinct(globalid)

  # 2. Standardize species name column
  navnkol <- intersect(c("art_norsk_navn", "norsk_navn"), names(ano_art_df))
  if (length(navnkol) == 1) {
    ano_art_df <- ano_art_df %>% dplyr::rename(navn_brukt = dplyr::all_of(navnkol))
  } else {
    ano_art_df$navn_brukt <- NA_character_
  }

  # 3. Apply TV rules
  TV_high_species <- TV_V_arter_df %>% dplyr::filter(!!high_expr)
  TV_low_species  <- TV_V_arter_df %>% dplyr::filter(!!low_expr)

  # 4. Find observed species
  art_high <- ano_art_df %>%
    dplyr::filter(parentglobalid %in% id$globalid,
                  navn_brukt %in% TV_high_species$NorskNavn)

  art_low <- ano_art_df %>%
    dplyr::filter(parentglobalid %in% id$globalid,
                  navn_brukt %in% TV_low_species$NorskNavn)
```

```r
# 5. Update M005 + sikkerhet (NO FILTERING)
ano_df %>%
  dplyr::mutate(
    # ---- 1 m² ----------------------------------------------------
    M005_mo_1m2 = dplyr::case_when(
      globalid %in% art_high$parentglobalid ~ M005_high,
      globalid %in% art_low$parentglobalid  ~ M005_low,

      !is.na(kartleggingsenhet_1m2_oppd) &
        kartleggingsenhet_1m2_oppd %in% KE_code &
        is.na(M005_mo_1m2) ~ M005_high,

      TRUE ~ M005_mo_1m2
    ),

    sikkerhet_mo_1m2 = dplyr::case_when(
      globalid %in% c(art_high$parentglobalid, art_low$parentglobalid) ~ "God oversettelse",

      !is.na(kartleggingsenhet_1m2_oppd) &
        kartleggingsenhet_1m2_oppd %in% KE_code ~ "Dårlig oversettelse",

      TRUE ~ sikkerhet_mo_1m2
    ),

    # ---- 250 m² --------------------------------------------------
    M005_mo_250m2 = dplyr::case_when(
      !is.na(kartleggingsenhet_250m2_oppd) &
        kartleggingsenhet_250m2_oppd %in% KE_code &
        !is.na(kartleggingsenhet_1m2_oppd) &
        kartleggingsenhet_1m2_oppd %in% KE_code ~ M005_mo_1m2,

      !is.na(kartleggingsenhet_250m2_oppd) &
        kartleggingsenhet_250m2_oppd %in% KE_code ~ M005_high,

      TRUE ~ M005_mo_250m2
    ),

    sikkerhet_mo_250m2 = dplyr::case_when(
      !is.na(kartleggingsenhet_250m2_oppd) &
        kartleggingsenhet_250m2_oppd %in% KE_code ~ "Akseptabel oversettelse",

      TRUE ~ sikkerhet_mo_250m2
```

```
      )
    )
}


sett_M005_TV_for_KE_multi <- function(ano_df_list,
                                      ano_art_df_list,
                                      KE_code,
                                      M005_high, M005_low,
                                      TV_rules_high, TV_rules_low,
                                      TV_V_arter_df) {

  purrr::map2(
    ano_df_list,
    ano_art_df_list,
    ~ sett_M005_TV_for_KE(
        ano_df  = .x,
        ano_art_df = .y,
        KE_code = KE_code,
        M005_high = M005_high,
        M005_low  = M005_low,
        TV_rules_high = TV_rules_high,
        TV_rules_low  = TV_rules_low,
        TV_V_arter_df = TV_V_arter_df
      )
  )
}
```

Funksjon for å skille på typer når det er tresjiktsdekning som definerer typene. For typer registrert i 1 m2 vegetasjonsruter antar vi at tresjiktsdekningen fra sirkelen også gjelder for 1 m2 ruta. Deretter filtrerer vi både 1 m2 og 250 m2 basert på tresjikstdekningen og setter typen som enten tresatt eller ikke basert på terskelen.

```
sett_tresjikt <- function(ano_df,
                          KE,
                          M005_tresatt = NULL,
                          M005_ikketresatt = NULL,
                          M020_tresatt = NULL,
                          M020_ikketresatt = NULL,
                          tresjikt_dekning = 10) {

  # Resolve dynamic/static values (resolve_val must return NULL for NULL inputs)
```

```r
M005_tresatt_val     <- resolve_val(M005_tresatt,     ano_df)
M005_ikketresatt_val <- resolve_val(M005_ikketresatt, ano_df)
M020_tresatt_val     <- resolve_val(M020_tresatt,     ano_df)
M020_ikketresatt_val <- resolve_val(M020_ikketresatt, ano_df)

# Standardiser tresjiktsdekning-kolonnenavn (gjør det én gang)
tres_col <- intersect(c("tresjikt_dekning", "tresjikt_dekning2"), names(ano_df))
if (length(tres_col) == 1 && tres_col != "tresjikt_dekning") {
  ano_df <- ano_df %>% dplyr::rename(tresjikt_dekning = dplyr::all_of(tres_col))
}
if (!("tresjikt_dekning" %in% names(ano_df))) {
  ano_df$tresjikt_dekning <- NA_real_
}

# Helper: KE-match (exact match). If you really need pattern match, swap to str_detect.
in_KE_1m2  <- !is.na(ano_df$kartleggingsenhet_1m2_oppd)  & ano_df$kartleggingsenhet_1m2_oppd
↪  %in% KE
in_KE_250m2 <- !is.na(ano_df$kartleggingsenhet_250m2_oppd) &
↪  ano_df$kartleggingsenhet_250m2_oppd %in% KE

# --- M005 ------------------------------------------------------------
if (!is.null(M005_tresatt)) {
  ano_df <- ano_df %>%
    dplyr::mutate(
      # 1 m2
      M005_mo_1m2 = dplyr::case_when(
        in_KE_1m2 &
          !is.na(tresjikt_dekning) &
          tresjikt_dekning >= tresjikt_dekning ~ M005_tresatt_val,

        in_KE_1m2 &
          is.na(M005_mo_1m2) ~ M005_ikketresatt_val,

        TRUE ~ M005_mo_1m2
      ),

      sikkerhet_mo_1m2 = dplyr::case_when(
        in_KE_1m2 ~ "Akseptabel oversettelse",
        TRUE ~ sikkerhet_mo_1m2
      ),

      # 250 m2
      M005_mo_250m2 = dplyr::case_when(
```

```
        in_KE_250m2 & in_KE_1m2 ~ M005_mo_1m2,
        in_KE_250m2 & !in_KE_1m2 ~ M005_ikketresatt_val,
        TRUE ~ M005_mo_250m2
      ),

      sikkerhet_mo_250m2 = dplyr::case_when(
        in_KE_250m2 ~ "Akseptabel oversettelse",
        TRUE ~ sikkerhet_mo_250m2
      )
    )
  }

  # --- M020 ------------------------------------------------------------
  if (!is.null(M020_tresatt)) {
    ano_df <- ano_df %>%
      dplyr::mutate(
        # 1 m2
        M020_mo_1m2 = dplyr::case_when(
          in_KE_1m2 &
            !is.na(tresjikt_dekning) &
            tresjikt_dekning >= tresjikt_dekning ~ M020_tresatt_val,

          in_KE_1m2 &
            is.na(M020_mo_1m2) ~ M020_ikketresatt_val,

          TRUE ~ M020_mo_1m2
        ),

        # 250 m2
        M020_mo_250m2 = dplyr::case_when(
          in_KE_250m2 & in_KE_1m2 ~ M020_mo_1m2,
          in_KE_250m2 & !in_KE_1m2 ~ M020_ikketresatt_val,
          TRUE ~ M020_mo_250m2
        )
      )
  }

  ano_df
}
```

Funksjon for å sette en spesifikk M005 og M020 kode.

```r
sett_manuell_koder <- function(ano_df, KE, M005_KE = NULL, M020_KE = NULL,
                               sikkerhet_val = "Dårlig oversettelse") {

  # --- Base update (M005 + sikkerhet) ----------------------------------------
  if (!is.null(M005_KE) && !is.na(M005_KE)) {
  ano_df <- ano_df %>%
    mutate(
      # M005 ------------------------------------------------------------------
      M005_mo_1m2 = ifelse(
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE, M005_KE,
        ↪  M005_mo_1m2
      ),
            # sikkerhet -------------------------------------------------------
      sikkerhet_mo_1m2 = ifelse(
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE, sikkerhet_val,
        ↪  sikkerhet_mo_1m2
      ),
      M005_mo_250m2 = ifelse(
        !is.na(kartleggingsenhet_250m2_oppd) & kartleggingsenhet_250m2_oppd %in% KE, M005_KE,
        ↪  M005_mo_250m2
      ),
      sikkerhet_mo_250m2 = ifelse(
        !is.na(kartleggingsenhet_250m2_oppd) & kartleggingsenhet_250m2_oppd %in% KE,
        ↪  sikkerhet_val, sikkerhet_mo_250m2
      )
    )
  }

  # --- Optional update: only if M020_KE is provided --------------------------
  if (!is.null(M020_KE) && !is.na(M020_KE)) {
    ano_df <- ano_df %>%
      mutate(
        M020_mo_1m2 = ifelse(
          !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE, M020_KE,
          ↪  M020_mo_1m2
        ),
        M020_mo_250m2 = ifelse(
          !is.na(kartleggingsenhet_250m2_oppd) & kartleggingsenhet_250m2_oppd %in% KE, M020_KE,
          ↪  M020_mo_250m2
        )
      )
  }
```

```
  return(ano_df)
}
```

7.4 Regler for artslister

Regler for artslister viser til hvordan vi skal skille de ulike typene fra hverandre når de varierer langs de ulike gradientene. For eksempel vil kalkkrevende arter ikke finnes (ha verdi 0) for lave basistrinn (KA a til f er 0 og KA g og h kan være 0 eller høyere). Vi bruker arter med preferanser for kalk for å skille om en type er høy, intermediær eller lav kalk. Vi bruker for KA ikke arter som forekommer ved lave verdier av KA ettersom de fleste artene kan forekomme langs hele KA-gradienten hvis de også forekommer ved lave KA-verdier. Ett unntak her er for V10-C-3, men det forklares nærmere for denne typen om hvorfor.

For tørrleggingsvarighet har vi todelte regelsett hvor noen arter forekommer ved laver TV-verdier og noen ved høye.

Regler for KA fastmark

```
# Når en type har KA d og/eller høyere og derfor bruker vi arter med KA d eller høyere for å
 ↪  skille på typene
KA_high_d_rules <-list(KA_a  = ~ KA_a == 0,
                  KA_b  = ~ KA_b == 0,
                  KA_c  = ~ KA_c == 0,
                  KA_d  = ~ KA_d >= 0, # Større eller lik 0
                  KA_e  = ~ KA_e >= 0,
                  KA_f  = ~ KA_f >= 0,
                  KA_g  = ~ KA_g >= 0,
                  KA_h  = ~ KA_h >= 0
                  )

# Når en type har KA g og/eller høyere og derfor bruker vi arter med KA g eller høyere for å
 ↪  skille på typene
KA_high_g_rules <-list(KA_a  = ~ KA_a == 0,
                  KA_b  = ~ KA_b == 0,
                  KA_c  = ~ KA_c == 0,
                  KA_d  = ~ KA_d == 0,
                  KA_e  = ~ KA_e == 0,
                  KA_f  = ~ KA_f == 0,
                  KA_g  = ~ KA_g >= 0, # Større eller lik 0
                  KA_h  = ~ KA_h >= 0
                  )
```

Regler for KA våtmark

```
# Når en type har KA a og/eller høyere og derfor bruker vi arter med KA a eller høyere for å
 ↪  skille på typene
KA_V_high_a_rules <- list(KA0 = ~ KA0 == 0,
                     a = ~ a   >= 0, # Større eller lik 0
                     b = ~ b   >= 0,
                     c = ~ c   >= 0,
                     d = ~ d   >= 0,
                     e = ~ e   >= 0,
                     f = ~ f   >= 0,
                     g = ~ g   >= 0,
                     h = ~ h   >= 0,
                     i = ~ i   >= 0
                     )

# Når en type har KA c og/eller høyere og derfor bruker vi arter med KA c eller høyere for å
 ↪  skille på typene
KA_V_high_c_rules <- list(KA0 = ~ KA0 == 0,
                     a = ~ a   == 0,
                     b = ~ b   == 0,
                     c = ~ c   >= 0, # Større eller lik 0
                     d = ~ d   >= 0,
                     e = ~ e   >= 0,
                     f = ~ f   >= 0,
                     g = ~ g   >= 0,
                     h = ~ h   >= 0,
                     i = ~ i   >= 0)

# Når en type har KA e og/eller høyere og derfor bruker vi arter med KA e eller høyere for å
 ↪  skille på typene
KA_V_high_e_rules <- list(KA0 = ~ KA0 == 0,
                     a = ~ a   == 0,
                     b = ~ b   == 0,
                     c = ~ c   == 0,
                     d = ~ d   == 0,
                     e = ~ e   >= 0, # Større eller lik 0
                     f = ~ f   >= 0,
                     g = ~ g   >= 0,
                     h = ~ h   >= 0,
                     i = ~ i   >= 0)
```

```
# Når en type har KA g og/eller høyere og derfor bruker vi arter med KA g eller høyere for å
 ↪  skille på typene
KA_V_high_g_rules <- list(KA0 = ~ KA0 == 0,
                     a = ~ a   == 0,
                     b = ~ b   == 0,
                     c = ~ c   == 0,
                     d = ~ d   == 0,
                     e = ~ e   == 0,
                     f = ~ f   == 0,
                     g = ~ g   >= 0, # Større eller lik 0
                     h = ~ h   >= 0,
                     i = ~ i   >= 0
                    )

# Når en type har KA d og/eller lavere og derfor bruker vi arter med KA d eller lavere for å
 ↪  skille på typene
KA_V_low_d_rules <- list(KA0 = ~ KA0 >= 0,
                     a = ~ a   >= 0,
                     b = ~ b   >= 0,
                     c = ~ c   >= 0,
                     d = ~ d   >= 0, # Større eller lik 0
                     e = ~ e   == 0,
                     f = ~ f   == 0,
                     g = ~ g   == 0,
                     h = ~ h   == 0,
                     i = ~ i   == 0
                     )
```

Regler for tørrleggingsvarighet (TV)

```
# Når en type har TV ij og/eller høyere og derfor bruker vi arter med TV ij eller høyere for å
 ↪  skille på typene
TV_rules_high <- list(
  cd  = ~ cd == 0,
  ef  = ~ ef == 0,
  gh  = ~ gh == 0,
  ij  = ~ ij >= 0, # Større eller lik 0
  k   = ~ k  >= 0
)

# Når en type har TV gh og/eller lavere og derfor bruker vi arter med TV gh eller lavere for å
 ↪  skille på typene
```

```
TV_rules_low <- list(
  cd  = ~ cd >= 0,
  ef  = ~ ef >= 0,
  gh  = ~ gh >= 0, # Større eller lik 0
  ij  = ~ ij == 0,
  k   = ~ k  == 0
)
```

7.5 Anslåelse av usikkerhet i manuell oversettelse

I Halvorsen (2025) anslås usikkerhet ved å først beregne følsomhet og spesifiseringsevne hvor det tas hensyn til kjente kartleggingsareal av både NiN 2.3 grunntypen og den oversatte grunntypen i NiN 3.0. I de tilfellene hvor vi har gjort manuelle oversettelser vil vi kun ha informasjon om NiN 2.3, og vi kan derfor ikke beregne usikkerhet som det Halvorsen (2025) har gjort. I tillegg kan ikke usikkerhetene av de automatiske oversettelsene brukes direkte inn i de manuelle oversettelsene ettersom at flere grunntyper som slås sammen til kartleggingsenheter ha ulike usikkerheter.

Vi vil sette fire typer usikkerheter i de manuelle oversettelsene: Svært god oversettelse, God oversettelse, Akseptabel oversettelse og Dårlig oversettelse. En Svært god oversettelse oversettelse er når det er et 1:1 forhold mellom NiN 2.3 og NiN 3.0. En God oversettelse oversettelse vil her anses til å være hvor vi bruker arter til å bestemme hvilken type det er. Siden arter kun representerer en 1 m2 rute vil aldri en sirkel ha God oversettelsee oversettelser. Akseptable oversettelser er hvor vi bruker andre former for vegetasjonsregistreringer for å sette type, slik som f.eks. tresjiktsdekning, og det er her de fleste sirkelregistreringene av KE vil havne. Dårlig oversettelsee oversettelser er hvor vi ikke kan bruke informasjon i ANO-datasettet til å sette typer, og vi derfor setter typen til å være den første kronologiske oversettelsen. Dette vil f.eks. gjelde hvor typene defineres etter variabelen dominerende kornstørrelse, hvor en slik oversettelse ikke kan vurderes automatisk fra datasettet, men man måtte enten hatt feltbefaring eller brukt bildene tatt i felt.

Sikkerhet vil kun anslås etter oversettelse til M005 og ikke M020 ettersom nesten alle M005-typer er nøstet i en M020-type (utenom typer i T32/TK01), og usikkerheten vil derfor være den samme for M005 og M020.

```
ano_type_list$type2024 <- ano_type_list$type2024 %>%
  mutate(sikkerhet_mo_1m2 = NA,
         sikkerhet_mo_250m2 = NA)

ano_type_list$type2025 <- ano_type_list$type2025 %>%
  mutate(sikkerhet_mo_1m2 = NA,
```

```
        sikkerhet_mo_250m2 = NA)
```

7.6 Manuelle oversettelser

I1 Snø- og isdekt fastmark

Her er det én NiN 2.3 type som skal oversettes til M005.

Tabell 1: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
I1-C-1	IA01-M005-01, IA01-M005-02, IA01-M005-03	3	IA01-M020-01	1

I1-C-1 snø- og isdekt fastmark

I NiN 2.3 er I1 delt inn i kun én kartleggingsenhet i 1:5000 målestokk (I1-C-1). I NiN 3.0 oversettes denne til fire M005 og to M020 hvor det er snø- og istype (SN) som definerer variasjonen i hovedtypen. Den ene typen som defineres av polar havis-overflate er ikke aktuell her, og derfor er det tre mulige M005 og én M020 som I1-C-1 kan overesettes til. Det er ingen informasjo i ANO-datasettet som kan hentes ut automatisk for å settes riktig KE, og vi oversetter derfor til den typen som er først i kronologisk rekkefølge (IA01-M005-01).

```
KE <- "I1-C-1"
M005_KE <- "IA01-M005-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE
)
```

T1 Nakent berg

Flere T1 Nakent berg er oversatt til å være flere av TA01Nakent berg. Hvor det i NiN 2.3 1:5000 var to trinn langs UF og fire trinn langs KA, er det i NiN 3.0 fire trinn langs UF og fem trinn langs KA. På grunn av lite til ingen informasjon som kan hentes automatisk ut fra datasettet,

setter vi her de ulike NiN 2.0 kartleggingsenhetene til å være de NiN 3.0 kartleggingsenhetene som har lavest UF og KA i sine trinn.

Tabell 2: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T1-C-1	TA01-M005-01, TA01-M005-02, TA01-M005-03, TA01-M005-04	4	TA01-M020-01, TA01-M020-02	2
T1-C-2	TA01-M005-03, TA01-M005-04	2	TA01-M020-02	1
T1-C-3	TA01-M005-05, TA01-M005-06, TA01-M005-07, TA01-M005-08	4	TA01-M020-01, TA01-M020-02	2
T1-C-4	TA01-M005-07, TA01-M005-08	2	TA01-M020-02	1
T1-C-5	TA01-M005-09, TA01-M005-10, TA01-M005-11, TA01-M005-12	4	TA01-M020-03, TA01-M020-04	2
T1-C-6	TA01-M005-11, TA01-M005-12	2	TA01-M020-04	1
T1-C-7	TA01-M005-13, TA01-M005-14, TA01-M005-15, TA01-M005-16, TA01-M005-17, TA01-M005-18, TA01-M005-19, TA01-M005-20	8	TA01-M020-05, TA01-M020-06	2
T1-C-8	TA01-M005-15, TA01-M005-16, TA01-M005-19, TA01-M005-20	4	TA01-M020-06	1

```
sett_T1_koder <- function(df) {

  # Kobler T1-C-x → M005 for 1 m2 og 250 m2
  map_M005 <- c(
    "T1-C-1" = "TA01-M005-01",
    "T1-C-2" = "TA01-M005-03",
    "T1-C-3" = "TA01-M005-05",
    "T1-C-4" = "TA01-M005-07",
    "T1-C-5" = "TA01-M005-09",
```

```
  "T1-C-6" = "TA01-M005-11",
  "T1-C-7" = "TA01-M005-13",
  "T1-C-8" = "TA01-M005-15"
)

# Kobler T1-C-x → M020 for 1 m2 og 250 m2
map_M020 <- c(
  "T1-C-1" = "TA01-M020-01",
  "T1-C-2" = "TA01-M020-02",
  "T1-C-3" = "TA01-M020-01",
  "T1-C-4" = "TA01-M020-02",
  "T1-C-5" = "TA01-M020-03",
  "T1-C-6" = "TA01-M020-04",
  "T1-C-7" = "TA01-M020-05",
  "T1-C-8" = "TA01-M020-06"
)

df %>%
  mutate(
    # 1m2 ----------------------------------------------------------------------
    M005_mo_1m2 = if_else(
      kartleggingsenhet_1m2_oppd %in% names(map_M005),
      unname(map_M005[kartleggingsenhet_1m2_oppd]),
      M005_mo_1m2
    ),

    M020_mo_1m2 = if_else(
      kartleggingsenhet_1m2_oppd %in% names(map_M020),
      unname(map_M020[kartleggingsenhet_1m2_oppd]),
      M020_mo_1m2
    ),
    # 250m2 --------------------------------------------------------------------
    M005_mo_250m2 = if_else(
      kartleggingsenhet_250m2_oppd %in% names(map_M005),
      unname(map_M005[kartleggingsenhet_250m2_oppd]),
      M005_mo_250m2
    ),

    M020_mo_250m2 = if_else(
      kartleggingsenhet_250m2_oppd %in% names(map_M020),
      unname(map_M020[kartleggingsenhet_250m2_oppd]),
      M020_mo_250m2
    ),
```

```
      # Legg til manuell oversettelse
      sikkerhet_mo_1m2 = case_when(
  !is.na(kartleggingsenhet_1m2_oppd) &
    kartleggingsenhet_1m2_oppd %in% names(map_M005) ~ "Dårlig oversettelse",
  TRUE ~ sikkerhet_mo_1m2
)
,
      sikkerhet_mo_250m2 = case_when(
  !is.na(kartleggingsenhet_250m2_oppd) &
    kartleggingsenhet_250m2_oppd %in% names(map_M005) ~ "Dårlig oversettelse",
  TRUE ~ sikkerhet_mo_250m2
)



    )
}

# Kjør funksjon
ano_type_list$type2024 <- sett_T1_koder(ano_type_list$type2024)
ano_type_list$type2025 <- sett_T1_koder(ano_type_list$type2025)
```

T2 Åpen grunnlendt mark

Her er det T2-C-1 og T2-C-5 som oversettes flere enn én gang.

Tabell 3: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T2-C-1	TA02-M005-01, TA02-M005-07	2	TA02-M020-01	1
T2-C-5	TA02-M005-02, TA02-M005-03	2	TA02-M020-02, TA02-M020-03	2

I NiN 2.3 defineres T2 av KA og uttørkningsfase (UF). I NiN 3.0 er KA og UF fortsatt definerende, men i tillegg er vannmetning (VM) lagt til. KA har også endret hvilke basistrinn som er i hvert trinn, hvor i NiN 2.3 var fordelingen abc, de, fg og hi, og i NiN 3.0 er fordelingen bc, def, ghi og j for VM 0a og bc, def og ghi for VM bc.

T2-C-1

Denne typen har i NiN 2.3 KA abc og UF def. I NiN 3.0 oversettes typen til TA02-M005-01 med KA bc, UF ef og VM 0a og TA02-M005-07 med KA bc, UF ef og VM bc. Typen er også oversatt til kun én M020-type. Vi bruker artstabell for VM arter på fastmark for å skille mellom disse typene, hvor tilstedeværelse av fuktkrevende arter gir TA02-M005-07 og ellers TA02-M005-01.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T2-C-1",
  M005_high = "TA02-M005-07",
  M005_low  = "TA02-M005-01",
  VM_arter_df = VM_arter)
```

T2-C-5

TA02-M005-02/TA02-M020-02 skilles fra TA02-M005-03/TA02-M020-03 ved mangel på sterkt kalkkrevende arter. Setter her kalkkrevende arter med de som har KA d og oppover større eller lik 0. Tilstedeværelse av en eller flere av disse gjør at observasjonen er TA02-M005-03, ellers TA02-M005-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T2-C-5",
  M005_high = "TA02-M005-03",
  M005_low  = "TA02-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)

# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T2-C-5",
  M020_high = "TA02-M020-03",
  M020_low  = "TA02-M020-02",
  KA_arter_df = KA_arter,
```

```
  KA_rules = KA_high_d_rules)
```

T3 Fjellhei, leside og tundra

Her er det seks NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 4: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T3-C-14	TA03-M005-12, TA03-M005-13	2	TA03-M020-02, TA03-M020-03	2
T3-C-2	TA03-M005-04, TA03-M005-14	2	TA03-M020-04	1
T3-C-5	TA03-M005-05, TA03-M005-15	2	TA03-M020-05	1
T3-C-7	TA03-M005-02, TA03-M005-03	2	TA03-M020-02, TA03-M020-03	2
T3-C-8	TA03-M005-05, TA03-M005-06	2	TA03-M020-05, TA03-M020-06	2
T3-C-9	TA03-M005-09, TA03-M005-10	2	TA03-M020-05, TA03-M020-06	2

I NiN 2.3 defineres T3 av KA og uttørkningsfase (UF). I NiN 3.0 er KA og UF fortsatt definerende, men i tillegg er vannmetning (VM) lagt til. KA har også endret hvilke basistrinn som er i hvert trinn, hvor i NiN 2.3 var fordelingen abc, de, fg og hi, og i NiN 3.0 er fordelingen bc, def, ghi og j for VM 0a og bc, def og ghi for VM bc.

T3-C-2

Denne typen har i NiN 2.3 KA abc og UF de. I NiN 3.0 oversettes typen til TA03-M005-04 med KA bc, UF de og VM 0a og TA03-M005-14 med KA bc, UF de og VM bc. Typen er også oversatt til kun én M020-type. Vi bruker artstabell for VM arter på fastmark for å skille mellom disse typene, hvor tilstedeværelse av fuktkrevende arter gir TA03-M005-14 og ellers TA03-M005-04.

```
# M005 ------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
```

```
  KE_code = "T3-C-2",
  M005_high = "TA03-M005-14",
  M005_low  = "TA03-M005-04",
  VM_arter_df = VM_arter)
```

T3-C-5

Denne typen har i NiN 2.3 KA de og UF de. I NiN 3.0 oversettes typen til TA03-M005-05 med KA def, UF de og VM 0a og TA03-M005-15 med KA def, UF de og VM bc. Typen er også oversatt til kun én M020-type.

Vi bruker artstabell for VM arter på fastmark for å skille mellom disse typene, hvor tilstedeværelse av fuktkrevende arter gir TA03-M005-15 og ellers TA03-M005-05.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-5",
  M005_high = "TA03-M005-15",
  M005_low  = "TA03-M005-05",
  VM_arter_df = VM_arter)
```

T3-C-7

Denne typen har i NiN 2.3 KA fg og UF bc. I NiN 3.0 oversettes typen til TA03-M005-02/TA03-M020-02 med KA def, UF bc og VM 0a og TA03-M005-03/TA03-M020-03 med KA ghi, UF bc og VM 0a. Vi bruker artstabell for KA fastmark til å skille typene fra hverandre, hvor tilstedeværelse av arter med KA g og oppover gir TA03-M005-03/TA03-M020-03 og ellers TA03-M005-02/TA03-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-7",
  M005_high = "TA03-M005-03",
  M005_low  = "TA03-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

```
# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-7",
  M020_high = "TA03-M020-03",
  M020_low  = "TA03-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T3-C-8

Denne typen har i NiN 2.3 KA fg og UF de. I NiN 3.0 oversettes typen til TA03-M005-05/TA03-M020-05 med KA def, UF de(fg) og VM 0a og TA03-M005-06/TA03-M020-06 med KA ghi, UF de(fg) og VM 0a. Vi bruker artstabell for KA fastmark til å skille typene fra hverandre, hvor tilstedeværelse av arter med KA g og oppover gir TA03-M005-06/TA03-M020-06 og ellers TA03-M005-05/TA03-M020-05.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-8",
  M005_high = "TA03-M005-06",
  M005_low  = "TA03-M005-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-8",
  M020_high = "TA03-M020-06",
  M020_low  = "TA03-M020-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T3-C-9

Denne typen har i NiN 2.3 KA fg og UF fg. I NiN 3.0 oversettes typen til TA03-M005-09/TA03-M020-05 med KA def, UF (de)fg og VM 0a og TA03-M005-10/TA03-M020-06 med KA ghi, UF (de)fg og VM 0a. Vi bruker artstabell for KA fastmark til å skille typene fra hverandre, hvor tilstedeværelse av arter med KA g og oppover gir TA03-M005-10/TA03-M020-06 og ellers TA03-M005-09/TA03-M020-05.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-9",
  M005_high = "TA03-M005-10",
  M005_low  = "TA03-M005-09",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-9",
  M020_high = "TA03-M020-06",
  M020_low  = "TA03-M020-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T3-C-14

Denne typen har i NiN 2.3 KA de og UF bc i tillegg til kildepåvirkning (KI) bc. I NiN 3.0 oversettes typen til TA03-M005-12/TA03-M020-02 med KA def, UF bc og VM bc og TA03-M005-13/TA03-M020-03 med KA ghi, UF bc og VM bc. Vi bruker artstabell for KA fastmark til å skille typene fra hverandre, hvor tilstedeværelse av arter med KA g og oppover gir TA03-M005-13/TA03-M020-03 og ellers TA03-M005-12/TA03-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-14",
  M005_high = "TA03-M005-13",
  M005_low  = "TA03-M005-12",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T3-C-14",
  M020_high = "TA03-M020-03",
  M020_low  = "TA03-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T4 Fastmarksskogsmark

Her er det ni NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 5: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T4-C-1	TB01-M005-01, TB01-M005-13	2	TB01-M020-01	1
T4-C-11	TB01-M005-08, TB01-M005-09	2	TB01-M020-05, TB01-M020-06	2
T4-C-15	TB01-M005-11, TB01-M005-12	2	TB01-M020-05, TB01-M020-06	2
T4-C-18	TB01-M005-14, TB01-M005-15	2	TB01-M020-02, TB01-M020-03	2
T4-C-19	TB01-M005-17, TB01-M005-18	2	TB01-M020-02, TB01-M020-03	2

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T4-C-20	TB01-M005-17, TB01-M005-18	2	TB01-M020-02, TB01-M020-03	2
T4-C-3	TB01-M005-02, TB01-M005-03	2	TB01-M020-02, TB01-M020-03	2
T4-C-5	TB01-M005-04, TB01-M005-16	2	TB01-M020-01	1
T4-C-7	TB01-M005-05, TB01-M005-06	2	TB01-M020-02, TB01-M020-03	2

Hvor det i NiN 2.3 var fire trinn langs KA, er det i NiN 3.0 kun tre trinn, og trinndelingen som skiller en kartleggingsenhet fra den neste, er endret fra NiN 2.3 til NiN 3.0. I tillegg er det i NiN 3.0 en del kartleggingsenheter hvor VM eller UF er den styrende LKM-en for oversettelsen. Som eksempel er det kommet inn fuktmarksutforminger av flere kjente typer skog; for eksempel for blåbærskog T4-C1 fra NiN 2.3 som kan oversettes til både blåbærskog og eventuelt blåbærfuktskog i NiN 3.0. I slike tilfeller har vi benyttet oss av hvorvidt det er en dominerende dekning av torvmoser i bunnsjiktet, og tilstedeværelse av særlig fuktkrevende arter.

T4-C-1

Kan oversettes til TB01-M005-13 og TB01-M005-01. TB01-M005-13 skilles fra TB01-M005-01 ved tilstedeværelse av særlig fuktkrevende arter, det vil si arter som ligger på UF ab, det vil si arter med verdi 0 for trinn over ab, tilstedeværelse av arter som myrfiol, skrubbær og tråsdiv og/eller et bunnsjikt i kvadratmeterruten som utgjøres av minst 50% torvmoser.

*(*tilstedeværelse av smyle er indikator for TB01-M005-01, men usikker på om vi bør bruke den)*

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-1",
  M005_high = "TB01-M005-13",
  M005_low  = "TB01-M005-01",
  VM_arter_df = VM_arter)
```

T4-C-3

Kan oversettes til både TB01-M005-02/TB01-M020-02 eller TB01-M005-03/TB01-M020-03. T4-C-3 dekker KA fg, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TB01-M005-03/TB01-M020-03 og ellers TB01-M005-02/TB01-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-3",
  M005_high = "TB01-M005-03",
  M005_low  = "TB01-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-3",
  M020_high = "TB01-M020-03",
  M020_low  = "TB01-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T4-C-5 IKKE FERDIG

Kan oversettes til TB01-M005-04 eller TB01-M005-16, det vil si at UF/VM er de definerende LKM. TB01-M005-16 er en fuktmarksutforming. Ved tilstedeværelse av arter som har trinn UF ef, oversettes det til TB01-M005-16. Dominans av blåtopp og/eller torvmoser vil også være kvalifiserende for oversettelse til den mest fuktige typen (hvor dominans her er satt til dekning på >50%). For M020 er det kun én type i NiN 3.0.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-5",
```

```
  M005_high = "TB01-M005-16",
  M005_low  = "TB01-M005-04",
  VM_arter_df = VM_arter)
```

T4-C-7

Kan oversettes til TB01-M005-05/TB01-M020-02 eller TB01-M005-06/TB01-M020-03. T4-C-7 dekker KA fg, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TB01-M005-06/TB01-M020-03 og ellers TB01-M005-05/TB01-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-7",
  M005_high = "TB01-M005-06",
  M005_low  = "TB01-M005-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-7",
  M020_high = "TB01-M020-03",
  M020_low  = "TB01-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T4-C-11

Kan oversettes til både TB01-M005-08/TB01-M020-05 eller TB01-M005-09/T01-M020-06 . T4-C-11 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TB01-M005-09/TB01-M020-06 og ellers TB01-M005-08/TB01-M020-05.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-11",
  M005_high = "TB01-M005-09",
  M005_low  = "TB01-M005-08",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-11",
  M020_high = "TB01-M020-06",
  M020_low  = "TB01-M020-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T4-C-15

Kan oversettes til TB01-M005-11/ TB01-M020-05 eller TB01-M005-12/TB01-M020-06. T4-C-15 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TB01-M005-12/TB01-M020-06 og ellers TB01-M005-11/ TB01-M020-05.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-15",
  M005_high = "TB01-M005-12",
  M005_low  = "TB01-M005-11",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

```
# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-15",
  M020_high = "TB01-M020-06",
  M020_low  = "TB01-M020-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T4-C-18

Kan oversettes til TB01-M005-14/TB01-M020-02 eler TB01-M005-15/TB01-M020-03. T4-C-18 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TB01-M005-15/TB01-M020-03 og ellers TB01-M005-14/TB01-M020-02.

Kalktrinnet gir en oversettelse, som muligens kan forbedres ved å legge til grunn følgende kriterier i artssammensetning: ved dominans av store bregner som skogburkne, fjellburkne, smørtelg og strutseving oversettes typen til TB01-M005-14, mens ved dominans av høgstauder som tyrihjelm, turt, hvitbladtistel og hvitsoleie oversettes typen til TB01-M005-15. Dette tas ikke hensyn til her.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-18",
  M005_high = "TB01-M005-15",
  M005_low  = "TB01-M005-14",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-18",
  M020_high = "TB01-M020-03",
```

```
  M020_low  = "TB01-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T4-C-19

Kan oversettes til TB01-M005-17/TB01-M020-02 eller TB01-M005-18/TB01-M020-03. T4-C-19 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TB01-M005-18/TB01-M020-03 og ellers TB01-M005-17/TB01-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-19",
  M005_high = "TB01-M005-18",
  M005_low  = "TB01-M005-17",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-19",
  M020_high = "TB01-M020-03",
  M020_low  = "TB01-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T4-C-20

Kan oversettes til TB01-M005-17/TB01-M020-02 eller TB01-M005-18/TB01-M020-03. T4-C-20 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover TB01-M005-18/TB01-M020-03 og ellers TB01-M005-17/TB01-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-20",
  M005_high = "TB01-M005-18",
  M005_low  = "TB01-M005-17",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T4-C-19",
  M020_high = "TB01-M020-03",
  M020_low  = "TB01-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T6 Strandberg

Her er det kun én type som skal oversettes til M005. M020 har kun én type.

Tabell 6: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T6-C-1	TC01-M005-01, TC01-M005-03	2	TC01-M020-01	1

Denne settes til den første M005-typen i listen ettersom det ikke finnes noe i datasettet som kan brukes for å bestemme hvilken M005-type det er.

```
KE <- "T6-C-1"
M005_KE <- "TC01-M005-01"
```

```
ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE
)
```

T7 Snøleie

Her er det fire NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 7: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T7-C-13	TC08-M005-02, TC08-M005-03	2	TC08-M020-02, TC08-M020-03	2
T7-C-4	TC08-M005-04, TC08-M005-05	2	TC08-M020-01, TC08-M020-02	2
T7-C-6	TC08-M005-02, TC08-M005-03	2	TC08-M020-02, TC08-M020-03	2
T7-C-7	TC08-M005-05, TC08-M005-06	2	TC08-M020-02, TC08-M020-03	2

T7-C-4

Kan være TC08-M005-04 eller TC08-M005-05 og TC08-M020-01 eller TC08-M020-02. T7-C-4 dekker både KA bc eller de, og NiN 3.0 typene dekker enten bc eller de. Bruker derfor KA-art-tabellen her til å skille begge, hvor KA d og oppover gir TC08-M005-05 og TC08-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T7-C-4",
  M005_high = "TC08-M005-05",
  M005_low  = "TC08-M005-04",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)
```

```
# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T7-C-4",
  M020_high = "TC08-M020-02",
  M020_low  = "TC08-M020-01",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)
```

T7-C-6 og T7-C-13

Både T7-C-6 og T7-C-13 oversettes til de samme NiN 3.0 typene, så gjør disse sammen her. Her skal KA være fg i NiN 2.3, men i NiN 3.0 er de enten def eller ghi. Skiller mellom def og ghi ved å si at typen er den med høyest KA (TC08-M005-03 og TC08-M020-03) hvis den har arter med KA g og oppover og den med lavest KA (TC08-M005-02 og TC08-M020-02) hvis den ikke har det.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = c("T7-C-6", "T7-C-13"),
  M005_high = "TC08-M005-03",
  M005_low  = "TC08-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = c("T7-C-6", "T7-C-13"),
  M020_high = "TC08-M020-03",
  M020_low  = "TC08-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T7-C-7

Her skal KA være fg i NiN 2.3, men i NiN 3.0 er de enten def eller ghi. Skiller mellom def og ghi ved å si at typen er den med høyest KA (TC08-M005-06 og TC08-M020-03) hvis den har hi-arter og den med lavest KA (TC08-M005-05 og TC08-M020-02) hvis den ikke har det.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T7-C-7",
  M005_high = "TC08-M005-06",
  M005_low  = "TC08-M005-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T7-C-7",
  M020_high = "TC08-M020-03",
  M020_low  = "TC08-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)
```

T13 Rasmark

Her er det seks NiN 2.3 typer som skal oversettes til både M005 og M020. På grunn av lite til ingen informasjon som kan hentes automatisk ut fra datasettet, setter vi her de ulike NiN 2.0 kartleggingsenhetene til å være de NiN 3.0 kartleggingsenhetene som kommer først alfabetisk.

Tabell 8: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T13-C-1	TD01-M005-04, TD01-M005-10	2	TD01-M020-04, TD01-M020-07	2
T13-C-10	TD01-M005-04, TD01-M005-07	2	TD01-M020-04	1
T13-C-12	TD01-M005-05, TD01-M005-08	2	TD01-M020-05	1
T13-C-14	TD01-M005-06, TD01-M005-09	2	TD01-M020-06	1
T13-C-4	TD01-M005-05, TD01-M005-11	2	TD01-M020-05, TD01-M020-08	2
T13-C-7	TD01-M005-06, TD01-M005-12	2	TD01-M020-06, TD01-M020-09	2

```
sett_T13_koder <- function(df) {

  # Mapping: T13-C-x → M005 for 1 m2 and 250 m2
  map_M005 <- c(
    "T13-C-1"  = "TD01-M005-04",
    "T13-C-4"  = "TD01-M005-05",
    "T13-C-7"  = "TD01-M005-06",
    "T13-C-10" = "TD01-M005-04",
    "T13-C-12" = "TD01-M005-05",
    "T13-C-14" = "TD01-M005-06"
  )

  # Mapping: T13-C-x → M020 for 1 m2 and 250 m2
  map_M020 <- c(
    "T13-C-1"  = "TD01-M020-04",
    "T13-C-4"  = "TD01-M020-05",
    "T13-C-7"  = "TD01-M020-06",
    "T13-C-10" = "TD01-M020-04",
    "T13-C-12" = "TD01-M020-05",
    "T13-C-14" = "TD01-M020-06"
  )

  df %>%
    mutate(
      # 1m2 ------------------------------------------------------------------------
```

```
      M005_mo_1m2 = if_else(
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% names(map_M005),
        unname(map_M005[kartleggingsenhet_1m2_oppd]),
        M005_mo_1m2
      ),
      M020_mo_1m2 = if_else(
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% names(map_M020),
        unname(map_M020[kartleggingsenhet_1m2_oppd]),
        M020_mo_1m2
      ),

      # 250 m2 ------------------------------------------------------------------
      M005_mo_250m2 = if_else(
        !is.na(kartleggingsenhet_250m2_oppd) & kartleggingsenhet_250m2_oppd %in% names(map_M005),
        unname(map_M005[kartleggingsenhet_250m2_oppd]),
        M005_mo_250m2
      ),
      M020_mo_250m2 = if_else(
        !is.na(kartleggingsenhet_250m2_oppd) & kartleggingsenhet_250m2_oppd %in% names(map_M020),
        unname(map_M020[kartleggingsenhet_250m2_oppd]),
        M020_mo_250m2
      ),


      # Legg til manuell oversettelse sikkerhet
      sikkerhet_mo_1m2 = case_when(
  !is.na(kartleggingsenhet_1m2_oppd) &
    kartleggingsenhet_1m2_oppd %in% names(map_M005) ~ "Dårlig oversettelse",
  TRUE ~ sikkerhet_mo_1m2
)
,
      sikkerhet_mo_250m2 = case_when(
  !is.na(kartleggingsenhet_250m2_oppd) &
    kartleggingsenhet_250m2_oppd %in% names(map_M005) ~ "Dårlig oversettelse",
  TRUE ~ sikkerhet_mo_250m2
)

        )
}

# Kjør funksjon
ano_type_list$type2024 <- sett_T13_koder(ano_type_list$type2024)
ano_type_list$type2025 <- sett_T13_koder(ano_type_list$type2025)
```

T14 Rabbe

Her er det to NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 9: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T14-C-1	TD06-M005-01, TD06-M005-02	2	TD06-M020-01, TD06-M020-02	2
T14-C-2	TD06-M005-02, TD06-M005-03	2	TD06-M020-02, TD06-M020-03	2

T14 er delt i to KE (begge langs KA). I NiN 3.0 er det fire KE-er både for M005 og M020, også disse langs KA. I NiN 2.3 skilles typene mellom abcde og fghi, men i NiN 3.0 er skillene bc, def, ghi og j, hvor j representerer saltanriket rabbe (vil ikke bli oversatt til dette her). Vi bruker KA-artstabellen til å skille typene fra hverandre.

T14-C-1

T14-C-1 er TD06-M005-02 med tilstedeværelse av KA d arter og oppover og TD06-M005-01 hvis ikke.

```
# M005
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T14-C-1",
  M005_high = "TD06-M005-02",
  M005_low  = "TD06-M005-01",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)

# M020
ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T14-C-1",
  M020_high = "TD06-M020-02",
  M020_low  = "TD06-M020-01",
```

```
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)
```

T14-C-2

T14-C-2 er TD06-M005-03 med tilstedeværelse av KA g arter og oppover og TD06-M005-02 hvis ikke.

```
# M005
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T14-C-2",
  M005_high = "TD06-M005-03",
  M005_low  = "TD06-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)

# M020
ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T14-C-2",
  M020_high = "TD06-M020-03",
  M020_low  = "TD06-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T15 Fosse-eng

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020.

Tabell 10: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T15-C-1	TD04-M005-01, TD04-M005-03	2	TD04-M020-01, TD04-M020-03	2

I NiN 2.3 T15 kun definert av KA, men er i NiN 3.0 også definert av vannsprutintensitet (VS). Denne settes til den første M005-typen i listen ettersom det ikke finnes noe i datasettet som kan brukes for å bestemme hvilken M005-type det er.

```
KE <- "T15-C-1"
M005_KE <- "TD04-M005-01"
M020_KE <- "TD04-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T16 Rasmarkhei og -eng

Her er det to NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 11: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T16-C-3	TD03-M005-02, TD03-M005-03	2	TD03-M020-02, TD03-M020-03	2
T16-C-6	TD03-M005-04, TD03-M005-05	2	TD03-M020-02, TD03-M020-03	2

Både T16-C-3 og T16-C-6 deles opp i to M005 og M020 KE-er basert på KA, hvor i NiN 2.3 omfattet de fg hvor i NiN 3.0 er f og g fordelt i def og ghi trinnene. Vi bruker KA-artstabellen til å skille typene fra hverandre. For M020 vil samme mønsteret gjelde, men her for begge NiN 2.3 typene: tilstedeværelse av KA g-arter og oppover gir TD03-M020-03 og hvis ikke er det TD03-M020-02.

T16-C-3

T16-C-3 er TD03-M005-03 med tilstedeværelse av hi arter og TD06-M005-02 hvis ikke

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T16-C-3",
  M005_high = "TD03-M005-03",
  M005_low  = "TD03-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)

# M020 ------------------------------------------------------------------------------
ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T16-C-3",
  M020_high = "TD03-M020-03",
  M020_low  = "TD03-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T16-C-6

T16-C-6 er TD03-M005-05 med tilstedeværelse av KA g arter og oppver og TD06-M005-04 hvis ikke

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T16-C-6",
  M005_high = "TD03-M005-05",
  M005_low  = "TD03-M005-04",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)

# M020 ------------------------------------------------------------------------------
ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T16-C-6",
  M020_high = "TD03-M020-03",
  M020_low  = "TD03-M020-02",
```

```
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T17 Aktiv skredmark

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020.

Tabell 12: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T17-C-2	TE02-M005-02, TE02-M005-03	2	TE02-M020-02, TE02-M020-03	2

T17-C-2

Denne typen er delt i to M005 og M020 basert på LKM-en Dominerende kornstørrelseklasse (S1) hvor grus og sand var slått sammen i NiN 2.3 og er egne typer i NiN 3.0. Ettersom vi ikke har noe som kan brukes i datasettet til å automatisk oversette typen, så oversettes alle til den typen som er først alfabetisk.

```
KE <- "T17-C-2"
M005_KE <- "TE02-M005-02"
M020_KE <- "TE02-M020-02"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T18 Åpen flomfastmark

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020. I tillegg er det to mulige hovedtyper (TE03 og TE08).

Tabell 13: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T18-C-1	TE03-M005-02, TE03-M005-03, TE08-M005-01	3	TE03-M020-01, TE03-M020-02, TE08-M020-01	3

I NiN 2.3 var T18 definert av Dominerende kornstørrelsesklasse (S1) hvor T18-C-1 inkluderte sand, grus og stein. I NiN 3.0 M005 og M020 er TE03 fordelt slik at sand og grus er én type (TE03-M005-02 og TE03-M020-01) og stein (TE03-M005-03 og TE03-M020-02).

Den andre NiN 3.0 hovedtypen inneholder TE08-M005-01, hvor "vegetasjonen her beskrives til å være mer sammenhengende og tettere enn i TE03". Vi skiller derfor TE03 og TE08 fra hverandre etter total karplantedekning, hvor høy dekning tyder på at typen er TE08 (TE08-M005-01 og TE08-M020-01). Vi setter her en terskel for høy dekning av total karplantedekning til å være 50 % i vegetasjonsruter. For sirkler er det ingen informasjon som kan brukes for å skille typene fra hverandre, og her setter vi typen til å være den som forekommer først kronologisk (TE03) utenom hvis både 1 m2 og 250 m2 er begge samme NiN 2.3 KE. Det er ingen informasjon i ANO-datasettet som gjør at man automatisk kan skille på dominerende kornstørrelse i TE03, og vi setter derfor typen til å være den første i alfabetisk rekkefølge (TE03-M005-02 og TE03-M020-01).

Vi står derfor igjen med to typer på M005-nivå (TE03-M005-02 og TE08-M005-01) og én type på M020-nivå (TE03-M020-01). For å skille mellom de to potensielle typene for M005 setter vi opp en funksjon som filtrerer på om vegetasjonsruta har over 50 % karplantedekning eller ikke. For M020 setter vi opp dette manuelt.

```
sett_T18C1 <-      function(ano_df, KE = "T18-C-1",
                            M005_high,
                            M005_low) {


  ano_df <- ano_df %>%
    # Standardiser tresjiktsdekning-kolonnenavn
    rename(karplanter_feltsjikt = all_of(intersect(c("karplanter_feltsjikt",
    ↪  "karplanter_feltsjikt2"), names(ano_df)))) %>%
    mutate(
      # 1 m2 ------------------------------------------------------------------
      M005_mo_1m2 = case_when(

        # Hvis kartleggingsenhet_1m2_oppd er KE og tresjiktsdekning i sirkel er over 10 %
```

```
  !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE &
   ↪  karplanter_feltsjikt >= karplanter_feltsjikt ~ M005_high,
  # Hvis tresjiktsdekningen ikke er over 10 % vil M005_mo_1m2 være NA
  !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE &
   ↪  is.na(M005_mo_1m2) ~ M005_low,

  TRUE ~ M005_mo_1m2
),

# 250 m2 ----------------------------------------------------------------
M005_mo_250m2 = case_when(
  # Hvis begge er KE → bruk 1m2-verdien
  !is.na(kartleggingsenhet_1m2_oppd) & !is.na(kartleggingsenhet_250m2_oppd) &
   ↪  kartleggingsenhet_250m2_oppd %in% KE &
    kartleggingsenhet_1m2_oppd %in% KE ~ M005_high,

  # Hvis bare 250m2 er KE → sett til ikke tresatt KE
  !is.na(kartleggingsenhet_1m2_oppd) & !is.na(kartleggingsenhet_250m2_oppd) &
   ↪  kartleggingsenhet_250m2_oppd %in%  KE &
    !(kartleggingsenhet_1m2_oppd %in%  KE) ~ M005_low,

  TRUE ~ M005_mo_250m2
),

# Legg til manuell oversettelse sikkerhet
sikkerhet_mo_1m2 =  case_when(
  !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE &
   ↪  karplanter_feltsjikt >= karplanter_feltsjikt ~ "Akseptabel oversettelse",
  !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE &
   ↪  is.na(M005_mo_1m2) ~ "Dårlig oversettelse",
  TRUE ~ sikkerhet_mo_1m2
),
sikkerhet_mo_250m2 =  case_when(
   !is.na(kartleggingsenhet_1m2_oppd) & !is.na(kartleggingsenhet_250m2_oppd) &
    ↪  kartleggingsenhet_250m2_oppd %in% KE &
    kartleggingsenhet_1m2_oppd %in% KE ~ "Akseptabel oversettelse",


  !is.na(kartleggingsenhet_1m2_oppd) & !is.na(kartleggingsenhet_250m2_oppd) &
   ↪  kartleggingsenhet_250m2_oppd %in%  KE &
    !(kartleggingsenhet_1m2_oppd %in%  KE) ~ "Dårlig oversettelse",
```

```
        TRUE ~ sikkerhet_mo_250m2
      )

    )

  return(ano_df)


}
```

Kjør funksjonen.

```
ano_type_list$type2024 <- sett_T18C1(ano_type_list$type2024,
                                     M005_high = "TE08-M005-01",
                                     M005_low = "TE03-M005-02"
                              )
ano_type_list$type2025 <- sett_T18C1(ano_type_list$type2025,
                                     M005_high = "TE08-M005-01",
                                     M005_low = "TE03-M005-02"
                              )

# M020 ------------------------------------------------------
KE <- "T18-C-1"
M020_KE <- "TI01-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M020_KE = M020_KE
)
```

T19 Oppfrysingsmark

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020.

Tabell 14: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T19-C-2	TE05-M005-02, TE05-M005-03	2	TE05-M020-02, TE05-M020-03	2

I NIN 2.3 er det to typer fordelt langs KA, hvor T19-C-2 den mest kalkrike delen av T19. I NiN 3.0 er kalkgradienten delt opp i tre typer, hvor KA fgh nå fordeler seg i KA def og ghi. Vi bruker her kalkkrevende arter for å skille ut den allermest kalkrike typen.

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T19-C-2",
  M005_high = "TE05-M005-03",
  M005_low  = "TE05-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------
ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T19-C-2",
  M020_high = "TE05-M020-03",
  M020_low  = "TE05-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T20 Isinnfrysingsmark

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020.

Tabell 15: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T20-C-1	TE09-M005-01, TE09-M005-02	2	TE09-M020-01, TE09-M020-02	2

I NIN 2.3 er det to typer fordelt langs KA, hvor T20-C-1 den mest kalkfattige delen av T20 (KA bcde). I NiN 3.0 er TE09 delt opp i tre typer langs KA, hvor KA nå fordeler seg i KA bc, def og ghi. Vi bruker her kalkkrevende arter (KA d og oppover) for å skille typene.

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T20-C-1",
  M005_high = "TE09-M005-02",
  M005_low  = "TE09-M005-01",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)


# M020 ------------------------------------------------------------------------------
ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T20-C-1",
  M020_high = "TE09-M020-02",
  M020_low  = "TE09-M020-01",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)
```

T21 Sanddynemark

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020.

Tabell 16: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T21-C-3	TE01-M005-03, TE01-M005-05	2	TE01-M020-02, TE01-M020-04	2

I NiN 2.3 defineres T21 av sandstabilisering (SS) og vannmetning (VM). I NiN 3.0 defineres den oversatte typen TE01 av SS, marin salinitet (SA) og VM.

T21-C-3

I NiN 2.3 defineres T21-C-3 av SS ghi og VM 0. I NiN 3.0 oversettes denne typen til TE01-M005-03/TE01-M020-02 (SS gh, VM 0 og SA a-g) og TE01-M005-05/TE01-M020-04 (SS efgh, VM 0 og SA 0). Etter våre undersøkelser finnes det ingen offiesielle lister over arter som er salttolerante, og ettersom dette er manglende er det ikke mulig å automatisk tillegne T21-C-3 til den ene eller den andre typen. Vi setter derfor her typen til å være den som forekommer først kronologisk (TE01-M005-03/TE01-M020-02)

```
KE <- "T21-C-3"
M005_KE <- "TE01-M005-03"
M020_KE <- "TE01-M020-02"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T22 Fjellgrashei og grastundra

Her er det fire NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 17: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T22-C-1	TA04-M005-01, TA04-M005-02	2	TA04-M020-01, TA04-M020-02	2

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T22-C-2	TA04-M005-01, TA04-M005-02	2	TA04-M020-01, TA04-M020-02	2
T22-C-3	TA04-M005-02, TA04-M005-03	2	TA04-M020-02, TA04-M020-03	2
T22-C-4	TA04-M005-02, TA04-M005-03	2	TA04-M020-02, TA04-M020-03	2

I NiN 2.3 er T22 definert av to LKM-er (KA og snødekkebetinget vekstreduksjon (SV)). I NiN 3.0 utgår SV og det er kun KA som er gjeldende. Derfor oversettes f.eks. både T22-C-1 og T22-C-2 til de samme M005 typene. I NiN 3.0 er det derimot tre typer langs KA (bc, def, ghi) istedenfor to (bcde, fgh). Bruker KA-arter for å definere hvor de ulike typene skal fordeles.

T22-C-1 og T22-C-2

Disse to typene fordeles til TA04-M005-01 eller TA04-M005-02 avhengig av tilstedeværelsen av KA d-arter og oppover.

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = c("T22-C-1", "T22-C-2"),
  M005_high = "TA04-M005-02",
  M005_low  = "TA04-M005-01",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)

# M020 ------------------------------------------------------------------------------
ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = c("T22-C-1", "T22-C-2"),
  M020_high = "TA04-M020-02",
  M020_low  = "TA04-M020-01",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_d_rules)
```

T22-C-3 og T22-C-4

Disse to typene fordeles til TA04-M005-02 eller TA04-M005-03 avhengig av tilstedeværelsen av KA g-arter og oppover.

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = c("T22-C-3", "T22-C-4"),
  M005_high = "TA04-M005-03",
  M005_low  = "TA04-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)

# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = c("T22-C-3", "T22-C-4"),
  M020_high = "TA04-M020-03",
  M020_low  = "TA04-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T25 Historisk skredmark

Her er det én NiN 2.3 type som skal oversettes til M005.

Tabell 18: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T25-C-2	TG01-M005-33, TG01-M005-34	2	TG01-M020-15	1

I NiN 2.3 er T25 kun definert av KA, hvor den aktuelle KE-en T25-C-2 inkluderer KA de og fg. I NiN 3.0 er TG01 definert av andre variabler i tillegg til KA, hvor T25-C-2 er oversatt til TG01-M005-33 og TG01-M005-34 hvor forskjellen mellom disse baserer seg på LKM-en Økologisk differensiering (ØD). Forskjellen på disse typene er at TG01-M005-34 har skogsmarkpreg og klart tredekke. Bruker derfor tresjiktsdekning i sirkel for å definere registreringer i både vegetasjonsrute og sirkel med antagelse om at vegetasjonsruta også dekkes av eventuell tresjiktsdekning i sirkel.

```
KE <- "T25-C-2"
M005_tresatt <- "TG01-M005-34"
M005_ikketresatt <- "TG01-M005-33"

# Kjør funksjonen
ano_type_list$type2024 <- sett_tresjikt(ano_type_list$type2024,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt)

ano_type_list$type2025 <- sett_tresjikt(ano_type_list$type2025,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt)
```

T26 Breforland og snø-avsmeltingsområde

Her er det tre NiN 2.3 typer som skal oversettes til M005.

Tabell 19: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T26-C-1	TG01-M005-26, TG01-M005-29	2	TG01-M020-12	1
T26-C-2	TG01-M005-26, TG01-M005-29	2	TG01-M020-12	1
T26-C-3	TG01-M005-25, TG01-M005-28	2	TG01-M020-12	1

T26-C-1 og T26-C-2

I NiN 2.3 var T26 delt inn i fire kartleggingsenheter hvor de tre første er tilstede i ANO-datasettet. De to første KE-ene (T26-C-1 og T26-C-2) er i NiN 3.0 begge oversatt til TG01-M005-26 og TG01-M005-29, hvor begge har de samme trinnene i ØD (A-C), men TG01-M005-26 har DK leire til grus og TG01-M005-29 har DK stein til stor blokk. Siden det er ingen måte å differensiere DK fra ANO-datasettet, settes begge typer til å være den første i kronologisk rekkefølge (TG01-M005-26).

```
KE <- c("T26-C-1", "T26-C-2")
M005_KE <- "TG01-M005-26"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE
)
```

T26-C-3

Denne typen oversettes i NiN 3.0 til to typer, hvor begge disse har ØD 0, men varierer langs DK: TG01-M005-25 har DK leire eller silt og TG01-M005-28 har DK stein til stor blokk. Siden det er ingen måte å differensiere DK fra ANO-datasettet, settes begge typer til å være den første i kronologisk rekkefølge (TG01-M005-25).

```
KE <- "T26-C-3"
M005_KE <- "TG01-M005-25"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE
)
```

T27 Blokkmark

Her er det fem NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 20: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T27-C-2	TG01-M005-12, TG01-M005-15	2	TG01-M020-06, TG01-M020-08	2
T27-C-4	TG01-M005-13, TG01-M005-15	2	TG01-M020-07, TG01-M020-08	2

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T27-C-5	TG01-M005-12, TG01-M005-13, TG01-M005-15	3	TG01-M020-06, TG01-M020-07, TG01-M020-08	3
T27-C-6	TG01-M005-10, TG01-M005-15	2	TG01-M020-06, TG01-M020-08	2
T27-C-7	TG01-M005-11, TG01-M005-15	2	TG01-M020-07, TG01-M020-08	2

T27-C-2

I NiN 2.3 var T27 delt inn i sju KE-er hvor fem av disse er tilstede i ANO-datasettet. T27-C-2 er i NiN 3.0 oversatt til to M005 og to M020, hvor en M005 og M020 hovesakelig defineres av ØD og KA (TG01-M005-12 og TG01-M020-06) og de andre av ØD og DK (TG01-M005-15 og TG01-M020-08). Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T27-C-2"
M005_KE <- "TG01-M005-12"
M020_KE <- "TG01-M020-06"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T27-C-4

Denne typen er i NiN 3.0 oversatt til to M005 og to M020, hvor en M005 og M020 hovesakelig defineres av ØD og KA (TG01-M005-13 og TG01-M020-07) og de andre av ØD og DK (TG01-M005-15 og TG01-M020-08). Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T27-C-4"
M005_KE <- "TG01-M005-13"
M020_KE <- "TG01-M020-07"
```

```
ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T27-C-5

Denne typen er i NiN 3.0 oversatt til tre M005 og tre M020, hvor to M005 og M020 hovesakelig defineres av ØD og KA (TG01-M005-12 og TG01-M020-06, og TG01-M005-13 og TG01-M020-07) og de andre av ØD og DK (TG01-M005-15 og TG01-M020-08). Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T27-C-5"
M005_KE <- "TG01-M005-12"
M020_KE <- "TG01-M020-06"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T27-C-6

Denne typen er i NiN 3.0 oversatt til to M005 og to M020, hvor en M005 og M020 hovesakelig defineres av ØD og KA (TG01-M005-10 og TG01-M020-06) og de andre av ØD og DK (TG01-M005-15 og TG01-M020-08). Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T27-C-6"
M005_KE <- "TG01-M005-10"
M020_KE <- "TG01-M020-06"

ano_type_list <- lapply(
```

```
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T27-C-7

Denne typen er i NiN 3.0 oversatt til to M005 og to M020, hvor en M005 og M020 hovesakelig defineres av ØD og KA (TG01-M005-11 og TG01-M020-07) og de andre av ØD og DK (TG01-M005-15 og TG01-M020-08). Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T27-C-7"
M005_KE <- "TG01-M005-11"
M020_KE <- "TG01-M020-07"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T29 Grus- og steindominert strand og strandlinje

Her er det tre NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 21: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T29-C-1	TG01-M005-17, TG01-M005-21	2	TG01-M020-09	1
T29-C-2	TG01-M005-18, TG01-M005-19, TG01-M005-20, TG01-M005-23, TG01-M005-24	5	TG01-M020-10, TG01-M020-11	2

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T29-C-5	TC03-M005-03, TC03-M005-05	2	TC03-M020-03, TC03-M020-05	2

T29-C-1

I NiN 2.3 var T29 delt inn i seks KE-er hvor tre av disse er tilstede i ANO-datasettet. T29-C-1 er i NiN 3.0 oversatt til to M005 og én M020, hvor disse hovedsakelig defineres av ØD og DK. Begge har ØD 0, men differensieres etter om dominerende kornstørrelse er grus (TG01-M005-17) eller stein-blokk (TG01-M005-21). Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T29-C-1"
M005_KE <- "TG01-M005-17"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE
)
```

T29-C-2

Denne typen er i NiN 3.0 oversatt til fem M005 og to M020, hvor disse hovedsakelig defineres av ØD og DK. Alle varierer i om de har ØD A, B eller D og DK grus eller stein-blokk. Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T29-C-2"
M005_KE <- "TG01-M005-18"
M020_KE <- "TG01-M020-10"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
```

```
)
```

T29-C-5

Denne typen er i NiN 3.0 oversatt til to M005 og to M020 i TC03. Alle typene varierer hovedsakelig langs DK, hvor TC03-M005-03 og TC03-M020-03 er dominert av sand og grus, og TC03-M005-05 og TC03-M020-05 av stor blokk. Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T29-C-5"
M005_KE <- "TC03-M005-03"
M020_KE <- "TC03-M020-03"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T30 Flomskogsmark

Her er det to NiN 2.3 typer som skal oversettes til M005.

Tabell 22: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T30-C-1	TF02-M005-04, TF02-M005-05	2	TF02-M020-01	1
T30-C-2	TF02-M005-01, TF02-M005-02	2	TF02-M020-01	1

T30-C-1

De to T30 typene i ANO-datasettet er hovedsakelig definert av dominerende kornstørrelse i NiN 2.3, hvor i NiN 3.0 er det DK sammen med vannforstyrrelsesintensitet (VF). De to NiN 3.0 typene som T30-C-1 oversettes til er begge dominert av DK sand-stein, men varierer i hvilken grad de har VF beskyttet eller eksponert. Siden det ikke finnes noe i datasettet som

kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T30-C-1"
M005_KE <- "TF02-M005-04"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE
)
```

T30-C-2

De to NiN 3.0 typene som T30-C-2 oversettes til er begge dominert av DK leire-silt, men varierer i hvilken grad de har VF beskyttet eller eksponert. Siden det ikke finnes noe i datasettet som kan brukes for å automatisk oversette fra NiN 2.3 til NiN 3.0, bruker vi den typen som kommer først kronologisk.

```
KE <- "T30-C-2"
M005_KE <- "TF02-M005-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE
)
```

T31 Boreal hei

Her er det fem NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 23: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T31-C-14	TH01-M005-10, TH01-M005-11	2	TH01-M020-02, TH01-M020-03	2

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T31-C-2	TH01-M005-04, TH01-M005-12	2	TH01-M020-04	1
T31-C-5	TH01-M005-05, TH01-M005-13	2	TH01-M020-05	1
T31-C-7	TH01-M005-02, TH01-M005-03	2	TH01-M020-02, TH01-M020-03	2
T31-C-8	TH01-M005-05, TH01-M005-06	2	TH01-M020-05, TH01-M020-06	2

T31-C-2

Kan oversettes til TH01-M005-04 og TH01-M005-12. De to typene i NiN 3.0 ligger på samme KA-trinn og UF-trinn, men har ulik vannmetning. Vi trenger altså å skille VM0a|bc. *Hvordan? Vi kan bruke UF, og si at dersom arter på trinn ef er til stede, så skal men oversette til fuktutformingen (TH01-M005-12). Eventuelt benytte myrflatepreg? Spørre Rune?*

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T31-C-2",
  M005_high = "TH01-M005-12",
  M005_low  = "TH01-M005-04",
  VM_arter_df = VM_arter)
```

T31-C-5 IKKE FERDIG

Kan oversettes til TH01-M005-05 og TH01-M005-13. De to typene i NiN 3.0 ligger på samme KA-trinn og UF-trinn, men har ulik vannmetning. Vi trenger altså å skille VM0a|bc.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T31-C-5",
  M005_high = "TH01-M005-13",
  M005_low  = "TH01-M005-05",
  VM_arter_df = VM_arter)
```

T31-C-7

Kan oversettes til TH01-M005-02/TH01-M020-02 eller TH01-M005-03/TH01-M020-03. T31-C-7 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TH01-M005-03/TH01-M020-03 og ellers TH01-M005-02/TH01-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T31-C-7",
  M005_high = "TH01-M005-03",
  M005_low  = "TH01-M005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T31-C-7",
  M020_high = "TH01-M020-03",
  M020_low  = "TH01-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T31-C-8

Kan oversettes til TH01-M005-05/TH01-M020-05 eller TH01-M005-06/TH01-M020-06. T31-C-8 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TH01-M005-06/TH01-M020-06 og ellers TH01-M005-05/TH01-M020-05.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T31-C-8",
```

```
  M005_high = "TH01-M005-06",
  M005_low  = "TH01-M005-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T31-C-8",
  M020_high = "TH01-M020-06",
  M020_low  = "TH01-M020-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T31-C-14

Kan oversettes til TH01-M005-10/TH01-M020-02 eller TH01-M005-11/TH01-M020-03. T31-C-14 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TH01-M005-11/TH01-M020-03 og ellers TH01-M005-10/TH01-M020-02.

I tillegg vil man kunne identifisere TH01-M005-10 ved tilstedeværelse av store bregner som skogburkne og fjellburkne, og på samme måte identifisere TH01-M005-11 ved tilstedeværelse av høgstaudearter som turt og tyrihjelm. Dette gjøres ikke her.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T31-C-14",
  M005_high = "TH01-M005-11",
  M005_low  = "TH01-M005-10",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
```

```
ano_df = ano_type_list,
ano_art_df = ano_art_list,
KE_code = "T31-C-14",
M020_high = "TH01-M020-03",
M020_low  = "TH01-M020-02",
KA_arter_df = KA_arter,
KA_rules = KA_high_g_rules)
```

T32 Semi-naturlig eng

Her er det ni NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 24: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T32-C-10	TK01-M005-25	1	TK01-M020-02, TK01-M020-05	2
T32-C-14	TK01-M005-11, TK01-M005-17, TK01-M005-23	3	TK01-M020-02, TK01-M020-05	2
T32-C-15	TK01-M005-05, TK01-M005-06	2	TK01-M020-02, TK01-M020-03	2
T32-C-2	TK01-M005-07, TK01-M005-13, TK01-M005-19	3	TK01-M020-01, TK01-M020-04	2
T32-C-20	TK01-M005-08, TK01-M005-09, TK01-M005-14	3	TK01-M020-02, TK01-M020-03, TK01-M020-05	3
T32-C-21	TK01-M005-20, TK01-M005-21	2	TK01-M020-05, TK01-M020-06	2
T32-C-4	TK01-M005-08, TK01-M005-14, TK01-M005-20	3	TK01-M020-02, TK01-M020-05	2
T32-C-5	TK01-M005-02, TK01-M005-03	2	TK01-M020-02, TK01-M020-03	2
T32-C-9	TK01-M005-25	1	TK01-M020-02, TK01-M020-05	2

T32-C-2

Kan oversettes til en av de tre typene TK01-M005-07 eller TK01-M005-13 eller TK01-M005-19 for målestokk 1: 5000. For målestokk 1:20 000 oversettes det til TK01-M020-01 eller TK01-M020-04.

TK01-M005-07 skilles fra TK01-M005-13 ogTK01-M005-19 ved at TK01-M005-07 skal ha tresjikt og de to andre skal ikke ha det. Vi skiller derfor på disse ved å si at gitt at sirkelen har tresjikt (altså, tresjiktsdekning > 0 eller større eller lik 1), så er typen TK01-M005-07.

Både TK01-M005-13 ogTK01-M005-19 skal ha ingen tresjikt, og for å skille disse to må man derimot se til lokal miljøvariabel LM_HM markbearbeidingsintensitet og skille trinn 0 (uryddet mark) fra trinn ab (markryddet for stein/mark med utjevnet overflate). Det er ingen informasjon i datasettet som kan brukes for å skille disse typene fra hverandre. Vi setter derfor typen til å være den første i kronologisk rekkefølge (TK01-M005-13) gitt at tresjikt er 0.

For målestokk 1:20 000 styrer miljøvariabel HA åpning av tresjikt. Gitt tresjiktsdekning = 0, oversettes det til TK01-M020-04. Gitt tresjiktsdekning >0 oversettes det til TK01-M020-01.

```
KE <- "T32-C-2"
M005_tresatt     <- "TK01-M005-07"
M005_ikketresatt <- "TK01-M005-13"
M020_tresatt     <- "TK01-M020-01"
M020_ikketresatt <- "TK01-M020-04"
tresjikt_dekning <- 1

# Kjør funksjonen
ano_type_list$type2024 <- sett_tresjikt(ano_type_list$type2024,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt,
                                        M020_tresatt = M020_tresatt,
                                        M020_ikketresatt = M020_ikketresatt,
                                        tresjikt_dekning = tresjikt_dekning)

ano_type_list$type2025 <- sett_tresjikt(ano_type_list$type2025,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt,
                                        M020_tresatt = M020_tresatt,
                                        M020_ikketresatt = M020_ikketresatt,
                                        tresjikt_dekning = tresjikt_dekning)
```

T32-C-4

Kan oversettes til TK01-M005-08, TK01-M005-14 eller TK01-M005-20 i målestokk 1: 5 000, og til TK01-M020-02 eller TK01-M020-05 i målestokk 1: 20 000.

Vi starter med målestokk 1:5000. Hvis det er tresjikt >0, skal det oversettes til TK01-M005-08. Videre for å skille -14 og -20 må man se benytte miljøvariabel HM markbearbeidingsintensitet og skille trinn 0 uryddet mark fra trinn ab markryddet for stein/mark med utjevnet overflate. Det er ingen informasjon i datasettet som kan brukes for å skille disse typene fra hverandre. Vi setter derfor typen til å være den første i kronologisk rekkefølge (TK01-M005-14) gitt at tresjikt er 0.

For målestokk 1:20 000 styrer miljøvariabel HA åpning av tresjikt. Gitt tresjiktsdekning = 0, oversettes det til TK01-M020-05. Gitt tresjiktsdekning >0 oversettes det til TK01-M020-02.

```
KE <- "T32-C-4"
M005_tresatt     <- "TK01-M005-08"
M005_ikketresatt <- "TK01-M005-143"
M020_tresatt     <- "TK01-M020-02"
M020_ikketresatt <- "TK01-M020-05"
tresjikt_dekning <- 1

# Kjør funksjonen
ano_type_list$type2024 <- sett_tresjikt(ano_type_list$type2024,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt,
                                        M020_tresatt = M020_tresatt,
                                        M020_ikketresatt = M020_ikketresatt,
                                        tresjikt_dekning = tresjikt_dekning)

ano_type_list$type2025 <- sett_tresjikt(ano_type_list$type2025,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt,
                                        M020_tresatt = M020_tresatt,
                                        M020_ikketresatt = M020_ikketresatt,
                                        tresjikt_dekning = tresjikt_dekning)
```

T32-C-5

Kan oversettes til TK01-M005-02/TK01-M020-02 eller TK01-M005-03/TK01-M020-03. T32-C-5 dekker KA f og g, mens oversettelse til en av de to typene i NiN 3.0 krever at man skiller mellom KA f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TK01-M005-03/TK01-M020-03 og ellers TK01-M005-02/TK01-M020-02.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T32-C-5",
  M005_high = "TK01-005-03",
  M005_low  = "TK01-005-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T32-C-5",
  M020_high = "TK01-M020-03",
  M020_low  = "TK01-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T32-C-9 og T32-C-10

Oversettes i målestokk 1:5000 til TK01-M005-25 eller TK01-M005-26. I målestokk 1: 20 000 oversettes det til enten TK01-M020-02, TK01-M020-03 eller TK01-M020-05 eller TK01-M020-06.

For målestokk 1:5000 er KA definerende, og man kan skille de to kartleggingsenhetene i NiN 3.0 på kalktrinn f og g. Vi bruker derfor artstabellen for KA fastmark til å skille begge, hvor KA g og oppover gir TK01-M005-26 og ellers TK01-M005-25.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = c("T32-C-9","T32-C-10"),
  M005_high = "TK01-M005-26",
  M005_low  = "TK01-M005-25",
  KA_arter_df = KA_arter,
```

```
  KA_rules = KA_high_g_rules)
```

For målestokk 1:20 000 er det fire mulig typer i NiN3.0, som må skilles på kombinasjoner av KA og lokal miljøvariabel HA åpning av tresjikt. Oversettelsen kan da skrives slik:

- gitt KA_def og tresjiktsdekning > 0, oversettes det til TK01-M020-02
- gitt KA_ghi og tresjiktsdekning >0, oversettes det til TK01-M020-03
- gitt KA_def og tresjiktsdekning = 0, oversettes det til TK01-M020-05
- gitt KA_ghi og tresjiktsdekning = 0, oversettes det til TK01-M020-06

For å gjøre dette lager vi en egen funksjon som håndterer KA samtidig og tresjiktsdekning.

```
# M020 ------------------------------------------------------
sett_KA_tresjikt_M005 <- function(ano_df, ano_art_df,
                              KE_code,
                              M005_high_KA_tresjikt,
                              M005_low_KA_tresjikt,
                              M005_high_KA_ingentresjikt,
                              M005_low_KA_ingentresjikt,
                              KA_arter_df,
                              KA_rules,
                              tresjikt_dekning) {

  # Build filter expressions from formula-based rules
  cond_expr <- make_filter_expr(KA_rules)

  # 1. Find relevant plots
  id <- ano_df %>%
    st_drop_geometry() %>%
    filter(kartleggingsenhet_1m2_oppd %in% KE_code) %>%
    select(globalid) %>%
    distinct()

  # 2. Select KA species dynamically
  KA_utvalgte <- KA_arter_df %>%
    filter(!!cond_expr)

  # 3. Standardise species name column
  navnkol <- intersect(c("art_norsk_navn", "norsk_navn"), names(ano_art_df))
  ano_art_df <- ano_art_df %>% rename(navn_brukt = all_of(navnkol))

  # 4. Find matching species observations
  utvalgte <- ano_art_df %>%
```

```
    filter(parentglobalid %in% id$globalid) %>%
    filter(navn_brukt %in% KA_utvalgte$NorskNavn)

  # 5. Update M020 codes
  ano_df %>%
    mutate(
      # 1 m2 ------------------------------------------------------------
      M005_mo_1m2 = case_when(
        globalid %in% utvalgte$parentglobalid & tresjikt_dekning >= tresjikt_dekning ~
↪  M005_high_KA_tresjikt,
        globalid %in% utvalgte$parentglobalid & tresjikt_dekning < tresjikt_dekning  ~
↪  M005_low_KA_tresjikt,
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE_code &
         ↪  is.na(M005_mo_1m2) & tresjikt_dekning >= tresjikt_dekning ~
         ↪  M005_high_KA_ingentresjikt,
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE_code &
         ↪  is.na(M005_mo_1m2) & tresjikt_dekning < tresjikt_dekning  ~
         ↪  M005_low_KA_ingentresjikt,

        TRUE ~ M005_mo_1m2
      ),

      # 250 m2 ----------------------------------------------------------
      M005_mo_250m2 = case_when(
        # Hvis 1 m2 og 250 m2 NiN 2.3 KE er likt så antar vi at det er samme type i sirkel
        !is.na(kartleggingsenhet_1m2_oppd) & !is.na(kartleggingsenhet_250m2_oppd) &
         ↪  kartleggingsenhet_250m2_oppd %in% KE_code &
          kartleggingsenhet_1m2_oppd %in% KE_code ~ M005_mo_1m2,

        # Hvis M005_mo_1m2 og M005_mo_250m2 ikke er de samme, sett lav KA-typer, men skill på
         ↪  tresjikt
        is.na(M005_mo_250m2) & tresjikt_dekning >= tresjikt_dekning ~ M005_low_KA_tresjikt,
        is.na(M005_mo_250m2) & tresjikt_dekning < tresjikt_dekning  ~ M005_low_KA_ingentresjikt,


        TRUE ~ M005_mo_250m2
      ),

      # sikkerhet -------------------------------------------------------
      sikkerhet_mo_1m2 = dplyr::case_when(
        globalid %in% utvalgte$parentglobalid & tresjikt_dekning >= tresjikt_dekning ~ "God
↪  oversettelse",
```

```
        globalid %in% utvalgte$parentglobalid & tresjikt_dekning < tresjikt_dekning ~ "God
↪  oversettelse",
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE_code &
         ↪  is.na(M005_mo_1m2) & tresjikt_dekning >= tresjikt_dekning ~ "God oversettelse",
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE_code &
         ↪  is.na(M005_mo_1m2) & tresjikt_dekning < tresjikt_dekning  ~ "God oversettelse",

        TRUE ~ sikkerhet_mo_1m2
      ),

      sikkerhet_mo_250m2 = dplyr::case_when(
        !is.na(kartleggingsenhet_1m2_oppd) & !is.na(kartleggingsenhet_250m2_oppd) &
         ↪  kartleggingsenhet_250m2_oppd %in% KE_code &
          kartleggingsenhet_1m2_oppd %in% KE_code ~ "Akseptabel oversettelse",

        # Hvis M005_mo_1m2 og M005_mo_250m2 ikke er de samme, sett lav KA-typer, men skill på
         ↪  tresjikt
        is.na(M005_mo_250m2) & tresjikt_dekning >= tresjikt_dekning ~ "Akseptabel oversettelse",
        is.na(M005_mo_250m2) & tresjikt_dekning < tresjikt_dekning  ~ "Akseptabel oversettelse",

        TRUE ~ sikkerhet_mo_250m2
    ))
}


sett_M005_KA_tresjikt_multi <- function(ano_df, ano_art_df,
                              KE_code,
                              M005_high_KA_tresjikt,
                              M005_low_KA_tresjikt,
                              M005_high_KA_ingentresjikt,
                              M005_low_KA_ingentresjikt,
                              KA_arter_df,
                              KA_rules,
                              tresjikt_dekning) {

  purrr::map2(
    ano_df,
    ano_art_df,
    ~ sett_KA_tresjikt_M005(
        ano_df = .x,
        ano_art_df = .y,
        KE_code = KE_code,
```

```
        M005_high_KA_tresjikt = M005_high_KA_tresjikt,
        M005_low_KA_tresjikt = M005_low_KA_tresjikt,
        M005_high_KA_ingentresjikt = M005_high_KA_ingentresjikt,
        M005_low_KA_ingentresjikt = M005_low_KA_ingentresjikt,
        KA_arter_df = KA_arter_df,
        KA_rules = KA_rules,
        tresjikt_dekning = tresjikt_dekning
      )
  )
}

# M020 ------------------------------------------------------------
sett_KA_tresjikt_M020 <- function(ano_df, ano_art_df,
                                  KE_code,
                                  M020_high_KA_tresjikt,
                                  M020_low_KA_tresjikt,
                                  M020_high_KA_ingentresjikt,
                                  M020_low_KA_ingentresjikt,
                                  KA_arter_df,
                                  KA_rules,
                                  tresjikt_dekning) {

  # Build filter expressions from formula-based rules
  cond_expr <- make_filter_expr(KA_rules)

  # 1. Find relevant plots
  id <- ano_df %>%
    st_drop_geometry() %>%
    filter(kartleggingsenhet_1m2_oppd %in% KE_code) %>%
    select(globalid) %>%
    distinct()

  # 2. Select KA species dynamically
  KA_utvalgte <- KA_arter_df %>%
    filter(!!cond_expr)

  # 3. Standardise species name column
  navnkol <- intersect(c("art_norsk_navn", "norsk_navn"), names(ano_art_df))
  ano_art_df <- ano_art_df %>% rename(navn_brukt = all_of(navnkol))

  # 4. Find matching species observations
  utvalgte <- ano_art_df %>%
    filter(parentglobalid %in% id$globalid) %>%
```

```
    filter(navn_brukt %in% KA_utvalgte$NorskNavn)

  # 5. Update M020 codes
  ano_df %>%
    mutate(
      # 1 m2 ------------------------------------------------------------
      M020_mo_1m2 = case_when(
        globalid %in% utvalgte$parentglobalid & tresjikt_dekning >= tresjikt_dekning ~
↪  M020_high_KA_tresjikt,
        globalid %in% utvalgte$parentglobalid & tresjikt_dekning < tresjikt_dekning  ~
↪  M020_low_KA_tresjikt,
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE_code &
         ↪  is.na(M020_mo_1m2) & tresjikt_dekning >= tresjikt_dekning ~
         ↪  M020_high_KA_ingentresjikt,
        !is.na(kartleggingsenhet_1m2_oppd) & kartleggingsenhet_1m2_oppd %in% KE_code &
         ↪  is.na(M020_mo_1m2) & tresjikt_dekning < tresjikt_dekning  ~
         ↪  M020_low_KA_ingentresjikt,

        TRUE ~ M020_mo_1m2
      ),

      # 250 m2 ----------------------------------------------------------
      M020_mo_250m2 = case_when(
        # Hvis 1 m2 og 250 m2 NiN 2.3 KE er likt så antar vi at det er samme type i sirkel
        !is.na(kartleggingsenhet_1m2_oppd) &  !is.na(kartleggingsenhet_250m2_oppd) &
         ↪  kartleggingsenhet_250m2_oppd %in% KE_code &
          kartleggingsenhet_1m2_oppd %in% KE_code ~ M020_mo_1m2,

        # Hvis M020_mo_1m2 og M020_mo_250m2 ikke er de samme, sett lav KA-typer, men skill på
         ↪  tresjikt
        is.na(M020_mo_250m2) & tresjikt_dekning >= tresjikt_dekning ~ M020_low_KA_tresjikt,
        is.na(M020_mo_250m2) & tresjikt_dekning < tresjikt_dekning  ~ M020_low_KA_ingentresjikt,


        TRUE ~ M020_mo_250m2
      )
      )

}


sett_M020_KA_tresjikt_multi <- function(ano_df, ano_art_df,
```

```
                            KE_code,
                            M020_high_KA_tresjikt,
                            M020_low_KA_tresjikt,
                            M020_high_KA_ingentresjikt,
                            M020_low_KA_ingentresjikt,
                            KA_arter_df,
                            KA_rules,
                            tresjikt_dekning) {

  purrr::map2(
    ano_df,
    ano_art_df,
    ~ sett_KA_tresjikt_M020(
        ano_df = .x,
        ano_art_df = .y,
        KE_code = KE_code,
        M020_high_KA_tresjikt = M020_high_KA_tresjikt,
        M020_low_KA_tresjikt = M020_low_KA_tresjikt,
        M020_high_KA_ingentresjikt = M020_high_KA_ingentresjikt,
        M020_low_KA_ingentresjikt = M020_low_KA_ingentresjikt,
        KA_arter_df = KA_arter_df,
        KA_rules = KA_rules,
        tresjikt_dekning = tresjikt_dekning
      )
  )
}
```

Kjør funksjonen.

```
# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_tresjikt_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = c("T32-C-9","T32-C-10"),
  M020_high_KA_tresjikt = "TK01-M020-03",
  M020_low_KA_tresjikt = "TK01-M020-02",
  M020_high_KA_ingentresjikt = "TK01-M020-06",
  M020_low_KA_ingentresjikt = "TK01-M020-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules,
  tresjikt_dekning = 1
```

```
  )
```

T32-C-14

Kan oversettes til TK01-M005-11, TK01-M005-17 eller TK01-M005-23, og til TK01-M020-02 eller TK01-M020-05 i NiN 3.0.

Vi starter med målestokk 1:5000. Hvis det er tresjikt >0, skal det oversettes til TK01-M005-11.

Så gjenstår det å skille TK01-M005-17 mot TK01-M005-23, og de skilles ved lokal miljøvariabel HM markbearbeidingsintensitet, trinn ab (mark ryddet for stein, ryddet mark med utjevnet overflate) mot trinn 0 (uryddet mark). Det er ingen informasjon i datasettet som kan brukes for å skille disse typene fra hverandre. Vi setter derfor typen til å være den første i kronologisk rekkefølge (TK01-M005-17) gitt at tresjikt er 0.

For målestokk 1:20 000 skal det hvis tresjiktsdekning >0 oversettes til TK01-M020-02.
Hvis tresjiktsdekning = 0 skal det oversettes til TK01-M020-05.

```
KE <- "T32-C-14"
M005_tresatt     <- "TK01-M005-11"
M005_ikketresatt <- "TK01-M005-17"
M020_tresatt     <- "TK01-M020-02"
M020_ikketresatt <- "TK01-M020-05"
tresjikt_dekning <- 1

# Kjør funksjonen
ano_type_list$type2024 <- sett_tresjikt(ano_type_list$type2024,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt,
                                        M020_tresatt = M020_tresatt,
                                        M020_ikketresatt = M020_ikketresatt,
                                        tresjikt_dekning = tresjikt_dekning)

ano_type_list$type2025 <- sett_tresjikt(ano_type_list$type2025,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt,
                                        M020_tresatt = M020_tresatt,
                                        M020_ikketresatt = M020_ikketresatt,
                                        tresjikt_dekning = tresjikt_dekning)
```

T32-C-15

Kan oversettes til TK01-M005-05 eller TK01-M005-06, og til TK01-M020-02 eller TK01-M020-03. C15 dekker KA f og g, mens oversettelse til en av de to typene i hver målestokk i NiN 3.0 krever at man skiller mellom KA f og g. Bruker derfor KA-art-tabellen her til å skille mellom KA opp til og med trinn f, som gir TK01-005-05 eller TK01-M020-02. KA fra trinn g og oppover gir TK01-005-06 eller TK01-M020-03.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T32-C-15",
  M005_high = "TK01-005-06",
  M005_low  = "TK01-005-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T32-C-15",
  M020_high = "TK01-M020-03",
  M020_low  = "TK01-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T32-C-20

Denne typen kan oversettes til fire typer i NiN 3.0 for målestokk 1: 5000; TK01-M005-08, TK01-M005-09, TK01-M005-14 eller TK01-M005-15, og fire typer i målestokk 1: 20 000: TK01-M020-02, TK01-M020-03, TK01-M020-05 eller TK01-M020-06.

Oversettelsen her defineres av både kalkgradienten (KAf|g) og hvorvidt arealet har et åpent tresjikt, eller er uten tresjikt (HA bc|y). Denne variabelen finnes ikke i NiN 2.3, men vi kan bruke data fra 250 m^2 sirkel på tresjiktsdekning.

Gitt KAdef og tresjiktsdekning > 0, oversettes det til TK01-M005-08 / TK01-M020-02
Gitt KAghi og tresjiktsdekning >0, oversettes det til TK01-M005-09 / TK01-M020-03

Gitt KAdef og tresjiktsdekning $= 0$, oversettes det til TK01-M005_14 / TK01-M020-05
Gitt KAghi og tresjiktsdekning $= 0$, oversettes det til Tk01-M005-15 / TK01-M020-06.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_tresjikt_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T32-C-20",
  M005_high_KA_tresjikt = "TK01-M005-09",
  M005_low_KA_tresjikt = "TK01-M005-08",
  M005_high_KA_ingentresjikt = "TK01-M005-15",
  M005_low_KA_ingentresjikt = "TK01-M005-14",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules,
  tresjikt_dekning = 1
  )

# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_tresjikt_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T32-C-20",
  M020_high_KA_tresjikt = "TK01-M020-03",
  M020_low_KA_tresjikt = "TK01-M020-02",
  M020_high_KA_ingentresjikt = "TK01-M020-06",
  M020_low_KA_ingentresjikt = "TK01-M020-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules,
  tresjikt_dekning = 1
  )
```

T32-C-21

Kan oversettes til TK01-M005-20 eller TK01-M005-21, og til TK01-M020-05 eller TK01-M020-06. Typen i NiN 2.3 dekker KA f og g, mens oversettelse til typene i NiN 3.0 krever at man skiller mellom KA f og g. Bruker derfor KA-art-tabellen her til å skille begge, hvor KA opp til og med trinn f gir TK01-005-20 eller TK01-M020-05, mens KA fra trinn g og oppover gir TK01-M005-21 eller TK01-M020-06.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
```

```
  ano_art_df = ano_art_list,
  KE_code = "T32-C-21",
  M005_high = "TK01-005-21",
  M005_low  = "TK01-005-20",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T32-C-21",
  M020_high = "TK01-M020-06",
  M020_low  = "TK01-M020-05",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T34 Kystlynghei

Her er det tre NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 25: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T34-C-2	TK03-M005-03, TK03-M005-06	2	TK03-M020-01	1
T34-C-4	TK03-M005-04, TK03-M005-07	2	TK03-M020-02	1
T34-C-5	TK03-M005-04, TK03-M005-05	2	TK03-M020-02, TK03-M020-03	2

Her er oversettelsesforholdet mellom NiN 2.3 og NiN 3.0 på grunntypenivå mer detaljert enn mellom KE-er, hvor det dukker opp to grunntyper for fukt (VM) for T34 i NiN 2.3 hvor disse oversettes til egne KE-er i NiN 3.0 (TK03-M005-06 og TK03-M005-07).

T34-C-2

Denne typen har i NiN 2.3 KA abc og UF d-h. I NiN 3.0 oversettes typen til TK03-M005-03 med KA bc, VM 0a og UF d-g og TK03-M005-06 med KA bc, VM bc og UF d-g. Typen er

også oversatt til kun én M020-type. Vi bruker artstabell for VM arter på fastmark for å skille mellom disse typene, hvor tilstedeværelse av fuktkrevende arter gir TK03-M005-06 og ellers TK03-M005-03.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T34-C-2",
  M005_high = "TK03-M005-06",
  M005_low  = "TK03-M005-03",
  VM_arter_df = VM_arter)
```

T34-C-4

Denne typen har i NiN 2.3 KA de og UF d-h. I NiN 3.0 oversettes typen til TK03-M005-04 med KA def, VM 0a og UF d-g og TK03-M005-07 med KA def, VM bc og UF d-g. Typen er også oversatt til kun én M020-type. Vi bruker artstabell for VM arter på fastmark for å skille mellom disse typene, hvor tilstedeværelse av fuktkrevende arter gir TK03-M005-07 og ellers TK03-M005-04.

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_VM_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T34-C-4",
  M005_high = " TK03-M005-07",
  M005_low  = "TK03-M005-04",
  VM_arter_df = VM_arter)
```

T34-C-5

Denne typen har i NiN 2.3 KA fg og UF d-h. I NiN 3.0 oversettes typen til TK03-M005-04/TK03-M020-02 med KA def, VM 0a og UF d-g og TK03-M005-05/TK03-M020-03 med KA ghi, VM bc og UF d-g. Typen er også oversatt til kun én M020-type. Vi bruker artstabell for VM arter på fastmark for å skille mellom disse typene, hvor tilstedeværelse av fuktkrevende arter gir TK03-M005-05/TK03-M020-03 og ellers TK03-M005-04/TK03-M020-02 .

```
# M005 ------------------------------------------------------------------------------

ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T34-C-5",
  M005_high = "TK03-M005-05",
  M005_low  = "TK03-M005-04",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)


# M020 ------------------------------------------------------------------------------

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T34-C-5",
  M020_high = "TK03-M020-03",
  M020_low  = "TK03-M020-02",
  KA_arter_df = KA_arter,
  KA_rules = KA_high_g_rules)
```

T36 Ny fastmark på tidligere våtmark og ferskvannsbunn

Her er det én NiN 2.3 type som skal oversettes til M005.

Tabell 26: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T36-C-1	TM05-M005-01, TM05-M005-02	2	TM05-M020-01	1

T36-C-1

Denne typen beskriver i NiN 2.3 den sterkt endrede tidligere våtmarka. I NiN 3.0 er denne typen blitt en egen hovedtype med to M005 KE-er, hvor det som skiller disse er om den drenerte myra er en jordvannsmyr (TM05-M005-01) eller en nedbørsmyr (TM05-M005-02). Jordvannsmyr vil som regel ha mer kalkkrevende arter enn nedbørsmyr, og vi bruker derfor KA-artstabell for våtmark (KA_V_arter) til å skille mellom disse typene. Hvis det er tilstedeværelse av arter

med KA a og oppover, så er typen TM05-M005-01 (jordvannsmyr-varianten). Hvis ikke er typen TM05-M005-02.

```
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "T36-C-1",
  M005_high = "TM05-M005-01",
  M005_low  = "TM05-M005-02",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_a_rules)
```

T37 Ny fastmark på sterkt modifiserte og syntetiske substrater, rask suksesjon

Her er det to NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 27: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T37-C-2	TM03-M005-07, TM03-M005-08	2	TM03-M020-06, TM03-M020-07	2
T37-C-3	TM03-M005-05, TM03-M005-06	2	TM03-M020-04, TM03-M020-05	2

Det er ingen informasjon i datasettet som kan brukes for å skille noen av disse typene fra hverandre. Vi setter derfor alle til å være den første typen i kronologisk rekkefølge.

T37-C-2

```
KE <- "T37-C-2"
M005_KE <- "TM03-M005-07"
M020_KE <- "VB01-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T37-C-3

```
KE <- "T37-C-3"
M005_KE <- "TM03-M005-05"
M020_KE <- "VB01-M020-02"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T39 Sterkt endret og ny fastmark i langsom suksesjon

Her er det én NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 28: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T39-C-3	TM02-M005-01, TM02-M005-02	2	TM02-M020-01, TM02-M020-02	2

Det er ingen informasjon i datasettet som kan brukes for å skille noen av disse typene fra hverandre. Vi setter derfor alle til å være den første typen i kronologisk rekkefølge.

T39-C-3

```
KE <- "T39-C-3"
M005_KE <- "TM02-M005-01"
M020_KE <- "TM02-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T45 Oppdyrket varig eng

Her er det to NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 29: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
T45-C-1	TO03-M005-01, TO04-M005-01, TO04-M005-02	3	TO03-M020-01, TO04-M020-01	2
T45-C-2	TO03-M005-02, TO03-M005-03	2	TO03-M020-01	1

Det er ingen informasjon i datasettet som kan brukes for å skille noen av disse typene fra hverandre. Vi setter derfor alle til å være den første typen i kronologisk rekkefølge.

T45-C-1

```
KE <- "T45-C-1"
M005_KE <- "TO03-M005-01"
M020_KE <- "TO03-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T45-C-2

```
KE <- "T45-C-2"
M005_KE <- "TO03-M005-02"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE
)
```

V1 Åpen jordvannsmyr

Her er det åtte NiN 2.3 typer som skal oversettes til M005.

Tabell 30: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
V1-C-1	VA01-M005-01, VA01-M005-02	2	VA01-M020-01	1
V1-C-2	VA01-M005-03, VA01-M005-04	2	VA01-M020-01	1
V1-C-3	VA01-M005-05, VA01-M005-06	2	VA01-M020-02	1
V1-C-4	VA01-M005-07, VA01-M005-08, VA01-M005-09	3	VA01-M020-03	1
V1-C-5	VA01-M005-10, VA01-M005-11	2	VA01-M020-01	1
V1-C-6	VA01-M005-12, VA01-M005-13	2	VA01-M020-01	1
V1-C-7	VA01-M005-14, VA01-M005-15	2	VA01-M020-02	1
V1-C-8	VA01-M005-16, VA01-M005-17, VA01-M005-18	3	VA01-M020-03	1

I V1 i NiN 2.3 er det totalt ni KE-er og i NiN 3.0 tilsvarende type VA01 er det 19 potensielle typer, hvor 18 av disse er listet opp her. Det som skiller V1 fra VA01 er at i VA01 er også LKM-en tørrleggingsvarighet (TV) definerende. Vi bruker arter langs TV til å skille alle de nye typene, hvor de har lav tørrleggingsvarighet har trinnverdiene cdefgh og de med høy har ijk. Vi vil her bruke et toveis-filter for å fange opp retningen på artene.

I tillegg er KA i blitt en egen KE i M005, hvor denne i NiN 2.3 ikke var en egen type i 1:5000. Vi bruker her KA artsliste for våtmark.

V1-C-1

```
# M005 ------------------------------------------------------------------------
ano_type_list <- sett_M005_TV_for_KE_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-1",
  M005_high = "VA01-M005-02",
```

```
  M005_low  = "VA01-M005-01",
  TV_rules_high = TV_rules_high,
  TV_rules_low  = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V1-C-2

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_TV_for_KE_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-2",
  M005_high = "VA01-M005-04",
  M005_low  = "VA01-M005-03",
  TV_rules_high = TV_rules_high,
  TV_rules_low  = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V1-C-3

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_TV_for_KE_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-3",
  M005_high = "VA01-M005-06",
  M005_low  = "VA01-M005-05",
  TV_rules_high = TV_rules_high,
  TV_rules_low  = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V1-C-4

Denne typen er her delt i tre M005 avhengig av TV, men også KA. V1-C-4 har i NiN 2.3 KA ghi og ingen skille på TV, men i NiN 3.0 er denne typen delt i VA01-M005-07 (KA gh og TV cdefgh), VA01-M005-08 (KA i og TV cdefgh) og VA01-M005-09 (KA ghi og TV ij). Vi kjører først funksjonen for KA hvor V1-C-4 fordeles etter kalk, og deretter en modifisert TV-funksjonen hvor de V1-C-4-observasjonene som har arter med høy TV-verdi istedenfor blir tillegnet VA01-M005-09.

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-4",
  M005_high = "VA01-M005-08",
  M005_low  = "VA01-M005-07",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_g_rules
)
```

Lager en modifisert funksjon av TV-funksjonen.

```
sett_M005_TV_for_KE_2 <- function(ano_df, ano_art_df,
                           KE_code,
                           M005_high,
                           TV_rules_high, TV_rules_low,
                           TV_V_arter_df) {


  # Build OR-expressions for high and low rule sets
  high_expr <- make_filter_expr(TV_rules_high)
  low_expr  <- make_filter_expr(TV_rules_low)

  # 1. Identify matching plots (KE type)
  id <- ano_df %>%
    st_drop_geometry() %>%
    filter(kartleggingsenhet_1m2_oppd %in% KE_code) %>%
    select(globalid) %>%
    distinct()

  # 2. Standardize species name column in the observation table
  navnkol <- intersect(c("art_norsk_navn", "norsk_navn"), names(ano_art_df))
  ano_art_df <- ano_art_df %>% rename(navn_brukt = all_of(navnkol))

  # 3. Apply high/low species rules on the TV lookup table
  TV_high_species <- TV_V_arter_df %>% filter(!!high_expr)
  TV_low_species  <- TV_V_arter_df %>% filter(!!low_expr)

  # 4. Match these species with the actual observations
  art_high <- ano_art_df %>%
    filter(parentglobalid %in% id$globalid) %>%
```

```
    filter(navn_brukt %in% TV_high_species$NorskNavn)

  art_low <- ano_art_df %>%
    filter(parentglobalid %in% id$globalid) %>%
    filter(navn_brukt %in% TV_low_species$NorskNavn)

  # 5. Update M005 values (TV codes)
  ano_df %>%
    mutate(
      # ---- 1 m² ------------------------------------------------------
      M005_mo_1m2 = case_when(
        globalid %in% art_high$parentglobalid ~ M005_high,
        globalid %in% art_low$parentglobalid  ~ M005_mo_1m2, # Lave typer skal være M005_mo_1m2
        # Hvis noen ikke overlapper med filteret i det hele tatt, sett til TV_low
        #!(globalid %in% c(art_high$parentglobalid, art_low$parentglobalid)) ~ M005_low,
        TRUE ~ M005_mo_1m2
      ),

      # ---- 250 m² ----------------------------------------------------
      M005_mo_250m2 = case_when(
        # Both KE codes match → inherit from 1m2
        !is.na(kartleggingsenhet_1m2_oppd) & !is.na(kartleggingsenhet_250m2_oppd) &
         ↪  kartleggingsenhet_250m2_oppd %in% KE_code &
        kartleggingsenhet_1m2_oppd %in% KE_code &
        M005_mo_1m2 %in% M005_high ~ M005_mo_1m2,

        # Only 250m2 has KE type og M005_mo_250m2 er NA → set low value
        #kartleggingsenhet_250m2_oppd %in% KE_code &
        #!(kartleggingsenhet_1m2_oppd %in% KE_code) &
        # is(M005_mo_250m2) ~ M005_low_1m2,

        TRUE ~ M005_mo_250m2
      )
    )
}

sett_M005_TV_for_KE_multi_2 <- function(ano_df_list,
                                ano_art_df_list,
                                KE_code,
                                M005_high, #M005_low,
                                TV_rules_high, TV_rules_low,
                                TV_V_arter_df) {
```

```
  purrr::map2(
    ano_df_list,
    ano_art_df_list,
    ~ sett_M005_TV_for_KE_2(
        ano_df  = .x,
        ano_art_df = .y,
        KE_code = KE_code,
        M005_high = M005_high,
        #M005_low  = M005_low,
        TV_rules_high = TV_rules_high,
        TV_rules_low  = TV_rules_low,
        TV_V_arter_df = TV_V_arter_df
      )
  )
}
```

Kjør den tilpassede TV-funksjonen.

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_TV_for_KE_multi_2(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-4",
  M005_high = "VA01-M005-09",
  TV_rules_high = TV_rules_high,
  TV_rules_low  = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V1-C-5

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_TV_for_KE_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-5",
  M005_high = "VA01-M005-11",
  M005_low  = "VA01-M005-10",
  TV_rules_high = TV_rules_high,
  TV_rules_low  = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V1-C-6

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_TV_for_KE_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-6",
  M005_high = "VA01-M005-13",
  M005_low  = "VA01-M005-12",
  TV_rules_high = TV_rules_high,
  TV_rules_low  = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V1-C-7

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_TV_for_KE_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-7",
  M005_high = "VA01-M005-15",
  M005_low  = "VA01-M005-14",
  TV_rules_high = TV_rules_high,
  TV_rules_low  = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V1-C-8

Denne typen er her delt i tre M005 avhengig av TV, men også KA. V1-C-8 har i NiN 2.3 KA ghi og ingen skille på TV, men i NiN 3.0 er denne typen delt i VA01-M005-16 (KA gh og TV cdefgh), VA01-M005-17 (KA i og TV cdefgh) og VA01-M005-18 (KA ghi og TV ij). Vi kjører først funksjonen for KA hvor V1-C-8 fordeles etter kalk, og deretter en modifisert versjon av TV-funksjonen (se V1-C-4) hvor de V1-C-8-observasjonene som har arter med høy TV-verdi istedenfor blir tillegnet VA01-M005-18.

```
# M005 ------------------------------------------------------------------------------
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-8",
```

```
  M005_high = "VA01-M005-17",
  M005_low  = "VA01-M005-16",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_g_rules
)
```

Kjør den tilpassede TV-funksjonen.

```
# M005 ------------------------------------------------------------------------
ano_type_list <- sett_M005_TV_for_KE_multi_2(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V1-C-8",
  M005_high = "VA01-M005-18",
  TV_rules_high = TV_rules_high,
  TV_rules_low = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V2 Myr- og sumpskogsmark

Her er det to NiN 2.3 typer som skal oversettes til M005.

Tabell 31: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
V2-C-1	VB01-M005-01, VB01-M005-02	2	VB01-M020-01	1
V2-C-2	VB01-M005-02, VB01-M005-03	2	VB01-M020-02	1

Her er oversettelsesforholdet mellom NiN 2.3 og NiN 3.0 på grunntypenivå mer detaljert enn mellom KE-er, hvor det dukker opp to grunntyper for kildesumpskogsmark i NiN 3.0 hvor den ene har KA def, noe som gjør at denne oversettes til både VB01-M005-02 og 03.

V2-C-1

V2-C-1 har i NiN 2.3 KA abcd. I NiN 3.0 er KA abcd delt i to typer (VB01-M005-01 og VB01-M005-02). Vi bruker tilstedeværelsen av arter med KA c og oppover som skille mellom disse typene her.

```
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V2-C-1",
  M005_high = "VB01-M005-02",
  M005_low  = "VB01-M005-01",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_c_rules)
```

V2-C-2

V2-C-2 inneholder i NiN 2.3 også én av grunntypene med kildevannspåvirkning (V2-7). I NiN 3.0 er denne oversatt til VB1-09 som har KA def og derfor overlapper med både VB01-M005-02 (KA cd) og VB01-M005-03 (KA ef). Vi bruker tilstedeværelsen av arter med KA e og oppover som skille mellom disse typene her.

```
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V2-C-2",
  M005_high = "VB01-M005-03",
  M005_low  = "VB01-M005-02",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_e_rules)
```

V3 Nedbørsmyr

Her er det to NiN 2.3 typer som skal oversettes til både M005 og M020 i to hovedtyper (VC01 og VF01).

Tabell 32: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
V3-C-1	VC01-M005-01, VC01-M005-02	2	VC01-M020-01	1
V3-C-2	VC01-M005-02, VF01-M005-01	2	VC01-M020-01, VF01-M020-01	2

I NiN 2.3 er V3 hovedsakelig definert av myrflatepreg (MF) og tørrleggingsvarighet (TV).

V3-C-1

I NiN 2.3 har V3-C-1 MF ef og TV c-k. I VC01 er det hovedsakelig kun TV som er definerende, og de to M005 KE-ene har enten TV c-h (VC01-M005-01) eller TV ijk (VC01-M005-02). Det er kun én KE i M020. Til å skille mellom de to M005 KE-ene bruker vi artstabell for TV.

```
ano_type_list <- sett_M005_TV_for_KE_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V3-C-1",
  M005_high = "VC01-M005-02",
  M005_low  = "VC01-M005-01",
  TV_rules_high = TV_rules_high,
  TV_rules_low  = TV_rules_low,
  TV_V_arter_df = TV_V_arter
)
```

V3-C-2

I NiN 2.3 har V3-C-2 MF cd og TV k. I NiN 3.0 oversettes denne typen til to hovedtyper avhengig om typen er tresatt (VF01) eller ikke (VC01). Bruker derfor tresjiktsdekning i sirkel for å definere registreringer i både vegetasjonsrute og sirkel med antagelse om at vegetasjonsruta også dekkes av eventuell tresjiktsdekning i sirkel.

```
KE <- "V3-C-2"
M005_tresatt     <- "VF01-M005-01"
M005_ikketresatt <- "VC01-M005-02"
M020_tresatt     <- "VF01-M020-01"
M020_ikketresatt <- "VC01-M020-01"
tresjikt_dekning <- 10

# Kjør funksjonen
ano_type_list$type2024 <- sett_tresjikt(ano_type_list$type2024,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt,
                                        M020_tresatt = M020_tresatt,
                                        M020_ikketresatt = M020_ikketresatt,
                                        tresjikt_dekning = tresjikt_dekning)

ano_type_list$type2025 <- sett_tresjikt(ano_type_list$type2025,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
```

```
                                    M005_ikketresatt = M005_ikketresatt,
                                    M020_tresatt = M020_tresatt,
                                    M020_ikketresatt = M020_ikketresatt,
                                    tresjikt_dekning = tresjikt_dekning)
```

V6 Våtsnøleie og snøleiekilde

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020.

Tabell 33: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
V6-C-9	VC04-M005-09, VC04-M005-10	2	VC04-M020-03, VC04-M020-04	2

V6-C-9

I NiN 2.3 er V6-C-9 definert av snødekkebetinget vekstsesongreduksjon (SV), kildevannspåvirkning (KI) og kalkinnhold (KA), med SV ef, KI de og KA c-i. I VC04 oversettes dette til to M005-typer og to M020-typer avhengig av om KA er c-f eller g-i. Vi bruker artstabell for KA til å skille typene, hvor KA g og oppover gir VC04-M005-10/VC04-M020-04 og ellers VC04-M005-09/VC04-M020-03.

```
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V6-C-9",
  M005_high = "VC04-M005-10",
  M005_low  = "VC04-M005-09",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_g_rules
)

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V6-C-9",
  M020_high = "VC04-M020-04",
  M020_low  = "VC04-M020-03",
  KA_arter_df = KA_V_arter,
```

```
  KA_rules = KA_V_high_g_rules
)
```

V10 Semi-naturlig våteng

Her er det tre NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 34: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
V10-C-1	VK02-M005-01, VK02-M005-02	2	VK02-M020-01	1
V10-C-2	VK02-M005-02, VK02-M005-03	2	VK02-M020-01, VK02-M020-02	2
V10-C-3	VK02-M005-01, VK02-M005-02, VK02-M005-03	3	VK02-M020-01, VK02-M020-02	2

I NiN 2.3 er V10 definert av kalkinnhold (KA) og kildevannspåvirkning (KI). I NiN 3.0 defineres VK02 kun av KA, hvor de tre M005-typene har KA bcd (VK02-M005-01), KA ef (VK02-M005-02) eller KA ghi (VK02-M005-03). For M020 er de to typene KA b-f (VK02-M020-01) eller KA ghi (VK02-M020-02). Vi bruker artstabell for KA i våtmark til å skille typene fra hverandre.

V10-C-1

I NiN 2.3 har V1-C-1 KA cde og KI 0a. I NiN 3.0 oversettes dette til én M020-type (VK02-M020-01) og to M005-typer: VK02-M005-01(KA bcd) og VK02-M005-02 (KA ef). Vi bruker artstabell for våtmark, hvor arter med KA e og oppover gir VK02-M005-02 og ellers VK02-M005-01.

```
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V10-C-1",
  M005_high = "VK02-M005-02",
  M005_low  = "VK02-M005-01",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_e_rules
)
```

V10-C-2

I NiN 2.3 har V1-C-2 KA fgh og KI 0a. I NiN 3.0 oversettes dette til to M005-typer VK02-M005-02 (KA ef) og VK02-M005-03 (KA ghi), og to M020-typer VK02-M020-01 (KA b-f) og VK02-M020-02 (KA ghi). Vi bruker artstabell for våtmark, hvor arter med KA g og oppover gir VK02-M005-03/VK02-M020-02 og ellers VK02-M005-02/VK02-M020-01.

```
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V10-C-2",
  M005_high = "VK02-M005-03",
  M005_low  = "VK04-M005-02",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_g_rules
)

ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V10-C-2",
  M020_high = "VK04-M020-02",
  M020_low  = "VK04-M020-01",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_g_rules
)
```

V10-C-3

I NiN 2.3 har V10-C-3 KA c-h og KI bc. I NiN 3.9 oversettes dette til alle VK02 typer (både M005 og M020). Til å skille typene fra hverandre her bruker vi artstabell for våtmark, hvor for M020 vil tilstedeværelse av arter med KA g og oppover gi VK02-M020-02 og ellers VK02-M020-01. For M005 må vi kjøre kalkfilteret to ganger, hvor vi først kjører det til å skille ut VK02-M005-03 (tilstedeværelse av arter med KA g og oppover) og deretter kjører et modifisert filter for VK02-M005-01 (ingen tilstedeværelse av arter med KA e og oppover).

Kjører først for M020.

```
ano_type_list <- sett_M020_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V10-C-3",
  M020_high = "VK04-M020-02",
```

```
  M020_low  = "VK04-M020-01",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_g_rules
)
```

Kjører deretter for M005.

```
ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V10-C-3",
  M005_high = "VK02-M005-03",
  M005_low  = "VK04-M005-02",
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_high_g_rules
)


ano_type_list <- sett_M005_KA_multi(
  ano_df = ano_type_list,
  ano_art_df = ano_art_list,
  KE_code = "V10-C-3",
  M005_high = "VK02-M005-03",
  M005_low  = ~M005_mo_1m2,
  KA_arter_df = KA_V_arter,
  KA_rules = KA_V_low_d_rules
)
```

V11 Torvtak

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020.

Tabell 35: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
V11-C-1	VM01-M005-05, VM01-M005-06, VM04-M005-01	3	VM01-M020-04, VM01-M020-05, VM04-M020-01	3

I NiN 2.3 er V11 definert som en sterkt endret våtmarkstype. Siden typen er sterkt endret vil det ikke være noe informasjon i datasettet som kan brukes for å skille de ulike oversettelsene fra hverandre. Vi velger derfor å her oversette disse typene til den som forekommer først kronologisk i oversettelsen.

V11-C-1

Denne typen er oversatt til enten VM01-M005-05/VM01-M020-04, VM01-M005-06/VM01-M020-05 eller VM04-M005-01/VM04-M020-01. Som sagt over om mangel på informasjon i datasettet, vil denne oversettes til VM01-M005-05/VM01-M020-04.

```
KE <-"V11-C-1"
M005_KE <- "VM01-M005-05"
M020_KE <- "VM01-M020-04"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

V12 Grøftet åpen torvmark

Her er det tre NiN 2.3 typer som skal oversettes til både M005 og M020.

Tabell 36: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
V12-C-1	VM01-M005-01, VM04-M005-01	2	VM01-M020-01, VM04-M020-01	2
V12-C-2	VM01-M005-02, VM01-M005-03, VM04-M005-01	3	VM01-M020-01, VM01-M020-02, VM04-M020-01	3
V12-C-3	VM01-M005-04, VM04-M005-01	2	VM01-M020-03, VM04-M020-01	2

I NiN 2.3 er V13 definert som en sterkt endret våtmarkstype. Siden typen er sterkt endret vil det ikke være noe informasjon i datasettet som kan brukes for å skille de ulike oversettelsene

fra hverandre. Vi velger derfor å her oversette disse typene til den som forekommer først kronologisk i oversettelsen.

V12-C-1

Denne typen er oversatt til enten VM01-M005-01/VM01-M020-01 eller VM04-M005-01/VM04-M020-01. Som sagt over om mangel på informasjon i datasettet, vil denne oversettes til VM01-M005-01/VM01-M020-01.

```
KE <- "V12-C-1"
M005_KE <- "VM01-M005-01"
M020_KE <- "VM01-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

V12-C-2

I NiN 2.3 har V12-C-2 KA efgh. I NiN 3.0 oversettes dette derfor til to M005 og M020 typer avhengig av om trinnet er KA ef (VM01-M005-02/VM01-M020-01) eller KA ghi (VM01-M005-03/VM01-M020-02). Vi bruker tilstedeværelsen av arter med KA g og oppover til å skille typene.

Denne typen er oversatt til enten VM01-M005-01/VM01-M020-01 eller VM04-M005-01/VM04-M020-01. Som sagt over om mangel på informasjon i datasettet, vil denne oversettes til VM01-M005-01/VM01-M020-01.

```
KE <-"V12-C-2"
M005_KE <- "VM01-M005-02"
M020_KE <- "VM01-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

V12-C-3

Denne typen er oversatt til enten VM01-M005-04/VM01-M020-03 eller VM04-M005-01/VM04-M020-01. Som sagt over om mangel på informasjon i datasettet, vil denne oversettes til VM01-M005-04/VM01-M020-01.

```
KE <- "V12-C-3"
M005_KE <- "VM01-M005-04"
M020_KE <- "VM01-M020-03"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

V13 Ny våtmark

Her er det én NiN 2.3 type som skal oversettes til både M005 og M020.

Tabell 37: NiN 2.3 KE-er til NiN 3.0 KE-er (M005 og M020)

NiN 2.3 KE	NiN 3.0 M005	Antall NiN 3.0 M005	NiN 3.0 M020	Antall NiN 3.0 M020
V13-C-1	VM03-M005-01, VM06-M005-01	2	VM03-M020-01, VM06-M020-01	2

I NiN 2.3 er V13 definert som en sterkt endret våtmarkstype. Siden typen er sterkt endret vil det ikke være noe informasjon i datasettet som kan brukes for å skille de ulike oversettelsene fra hverandre. Vi velger derfor å her oversette denne typen til den som forekommer først kronologisk i oversettelsen.

```
KE <- "V13-C-1"
M005_KE <- "VM03-M005-01"
M020_KE <- "VM03-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
```

```
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

7.7 Manuelle oversettelser for de med ingen automatiske oversettelser

Ved å bruke Halvorsen (2025) var det flere typer som ikke ble oversatt. Dette kunne være f.eks. at typen er registrert på hovedtypenivå og derfor ikke ble oversatt da NiN 2.3 KE 1:5000 ble oversatt til grunntyper. Eller så kan det være at typen ikke er oversatt i Halvorsen (2025) og at det derfor kreves en manuell og kontekstavhengig vurdering for å oversette.

```
uoversatte_typer <- ano_type_list$type2024 %>%
  # 2024 ------------------------------------------------------------------------------------------
   ↪  --------------------------------
  # 1 m2
  filter((!is.na(kartleggingsenhet_1m2_oppd)) & is.na(ano_GT_NiN3_0_1m2)) %>%
  select(kartleggingsenhet_1m2_oppd) %>%
  rename(KE_NiN2_3 = kartleggingsenhet_1m2_oppd) %>%
  # 250 m2
  bind_rows(
    ano_type_list$type2024 %>%
    filter(!is.na(kartleggingsenhet_250m2_oppd) & is.na(ano_GT_NiN3_0_250m2)) %>%
    select(kartleggingsenhet_250m2_oppd) %>%
    rename(KE_NiN2_3 = kartleggingsenhet_250m2_oppd)
    ) %>%
  # 2025 ------------------------------------------------------------------------------------------
   ↪  --------------------------------
  # 1 m2
    bind_rows(
      ano_type_list$type2025 %>%
        filter((!is.na(kartleggingsenhet_1m2_oppd)) & is.na(ano_GT_NiN3_0_1m2)) %>%
        select(kartleggingsenhet_1m2_oppd) %>%
        rename(KE_NiN2_3 = kartleggingsenhet_1m2_oppd)
    ) %>%
  # 250 m2
  bind_rows(
    ano_type_list$type2025 %>%
    filter((!is.na(kartleggingsenhet_250m2_oppd)) & is.na(ano_GT_NiN3_0_250m2)) %>%
```

```
    select(kartleggingsenhet_250m2_oppd) %>%
    rename(KE_NiN2_3 = kartleggingsenhet_250m2_oppd)
    ) %>%
  st_drop_geometry() %>%
  select(KE_NiN2_3) %>%
  distinct() %>%
  arrange(KE_NiN2_3)
```

Her er det registrert typer alle nivåer mellom hovedtypegruppeer (f.eks. T), hovedtyper (f.eks. T1) og på kartleggingsenhetsnivå (f.eks. T17-C-1). hovedtypegruppe-kodene er de samme i NiN 2.3 og i NiN 3.0, så de kopieres bare. For å oversette hovedtyper bruker vi Vedlegg 6 i Halvorsen (2025). For å oversette kartleggingsenheter gjør vi manuelle vurderinger per type.

Hovedtypegruppe

Alle kartlegginger som kun er hovedtypegrupper overføres fra NiN 2.3 til NiN 3.0.

```
# Først, filtrer ut de med kun hovedtypegruppe
ano_hts <- uoversatte_typer %>%
  filter(str_length(KE_NiN2_3) == 1)

# Kopier den registrerte typen hvis den er et hovedtypegruppe
ano_type_list$type2024 <- ano_type_list$type2024 %>%
  mutate(M005_mo_1m2=ifelse(kartleggingsenhet_1m2_oppd %in% ano_hts$KE_NiN2_3,
                            kartleggingsenhet_1m2_oppd,
                            M005_mo_1m2),
         M005_mo_250m2=ifelse(kartleggingsenhet_250m2_oppd %in% ano_hts$KE_NiN2_3,
                              kartleggingsenhet_250m2_oppd,
                              M005_mo_250m2),
         M020_mo_1m2=ifelse(kartleggingsenhet_1m2_oppd %in% ano_hts$KE_NiN2_3,
                            kartleggingsenhet_1m2_oppd,
                            M020_mo_1m2),
         M020_mo_250m2=ifelse(kartleggingsenhet_250m2_oppd %in% ano_hts$KE_NiN2_3,
                              kartleggingsenhet_250m2_oppd,
                              M020_mo_250m2),
      # sikkerhet ----------------------------------------------------------------
      sikkerhet_mo_1m2 = ifelse(
        kartleggingsenhet_1m2_oppd %in% ano_hts$KE_NiN2_3,
        "Svært god oversettelse",
        sikkerhet_mo_1m2
      ),
```

```
      sikkerhet_mo_250m2 = ifelse(
        kartleggingsenhet_250m2_oppd %in% ano_hts$KE_NiN2_3,
        "Svært god oversettelse",
        sikkerhet_mo_250m2
      )

         )

ano_type_list$type2025 <- ano_type_list$type2025 %>%
  mutate(M005_mo_1m2=ifelse(kartleggingsenhet_1m2_oppd %in% ano_hts$KE_NiN2_3,
                            kartleggingsenhet_1m2_oppd,
                            M005_mo_1m2),
         M005_mo_250m2=ifelse(kartleggingsenhet_250m2_oppd %in% ano_hts$KE_NiN2_3,
                              kartleggingsenhet_250m2_oppd,
                              M005_mo_250m2),
         M020_mo_1m2=ifelse(kartleggingsenhet_1m2_oppd %in% ano_hts$KE_NiN2_3,
                            kartleggingsenhet_1m2_oppd,
                            M020_mo_1m2),
         M020_mo_250m2=ifelse(kartleggingsenhet_250m2_oppd %in% ano_hts$KE_NiN2_3,
                              kartleggingsenhet_250m2_oppd,
                              M020_mo_250m2),
         # sikkerhet ------------------------------------------------------------
      sikkerhet_mo_1m2 = ifelse(
        kartleggingsenhet_1m2_oppd %in% ano_hts$KE_NiN2_3,
        "Svært god oversettelse",
        sikkerhet_mo_1m2
      ),
      sikkerhet_mo_250m2 = ifelse(
        kartleggingsenhet_250m2_oppd %in% ano_hts$KE_NiN2_3,
        "Svært god oversettelse",
        sikkerhet_mo_250m2
      ))
```

Hovedtyper

For å hente ut oversettelse mellom hovedtyper filtrerer vi først ut de radene med hovedtype-oversettelser (de radene hvor Kode-kolonnen ikke inneholder "-") i Vedlegg 6 i Halvorsen (2025). Til å oversette til NiN 3.0 bruker vi HT-kolonnen, hvor vi fjerner "NA-".

```
nin3_oversettelsnokkel_HT <- nin3_oversettelsnokkel %>%
  # Velg de to relevante kolonnene
```

```
  select(Kode, HT) %>%
  # Finn rader med hovedtypeoversettelser
  filter(!str_detect(Kode, "-")) %>%
  # Finpuss HT kolonnen
  mutate(
         HT = HT |>
           # Fjern "NA-"
           str_remove_all("NA-") |>
           # Fjern \r\n
           str_replace_all("\r\n",",") |>
           # Noen hadde komma fra før av og andre ikke. Fjerner dobbel komma
           str_replace_all(",,",",")

         ) %>%
  distinct()
```

Legg til alle oversettelser som ikke inneholder flere typer eller mangler oversettelser (markert med "×").

```
# Først, filtrer ut de med kun hovedtyper
ano_ht <- uoversatte_typer %>%
  filter(str_length(KE_NiN2_3) > 1 & !str_detect(KE_NiN2_3,"-")) %>%
  arrange(KE_NiN2_3)

# Finn oversettelsen i oversettelsesnøkkelen
ht_oversettelse_auto <- nin3_oversettelsnokkel_HT %>%
  filter(Kode %in% ano_ht$KE_NiN2_3) %>%
  # Hent ut de som har kun én hovedtype i NiN 3.0
  filter(!str_detect(HT,",")) %>%
  # Lag en kopi av Kode-kolonnen
  mutate(Kode2=Kode)


ano_type_list$type2024 <- ano_type_list$type2024 %>%

  # Join for 1 m²
  left_join(ht_oversettelse_auto %>%
              rename(HT_1m2 = HT, Kode_1m2 = Kode, Kode2_1m2 = Kode2),
            by = c("kartleggingsenhet_1m2_oppd" = "Kode2_1m2")) %>%

  # Join for 250 m²
  left_join(ht_oversettelse_auto %>%
```

```
              rename(HT_250m2 = HT, Kode_250m2 = Kode, Kode2_250m2 = Kode2),
            by = c("kartleggingsenhet_250m2_oppd" = "Kode2_250m2")) %>%

  mutate(
    # apply translations
    M005_mo_1m2     = coalesce(HT_1m2, M005_mo_1m2),
    M005_mo_250m2   = coalesce(HT_250m2, M005_mo_250m2),

    M020_mo_1m2     = coalesce(HT_1m2, M020_mo_1m2),
    M020_mo_250m2   = coalesce(HT_250m2, M020_mo_250m2),

    sikkerhet_mo_1m2   = if_else(!is.na(HT_1m2), "Svært god oversettelse", sikkerhet_mo_1m2),
    sikkerhet_mo_250m2 = if_else(!is.na(HT_250m2), "Svært god oversettelse", sikkerhet_mo_250m2)
  ) %>%

  select(-starts_with("HT_"), -starts_with("Kode"))


ano_type_list$type2025 <- ano_type_list$type2025 %>%

  # Join for 1 m²
  left_join(ht_oversettelse_auto %>%
              rename(HT_1m2 = HT, Kode_1m2 = Kode, Kode2_1m2 = Kode2),
            by = c("kartleggingsenhet_1m2_oppd" = "Kode2_1m2")) %>%

  # Join for 250 m²
  left_join(ht_oversettelse_auto %>%
              rename(HT_250m2 = HT, Kode_250m2 = Kode, Kode2_250m2 = Kode2),
            by = c("kartleggingsenhet_250m2_oppd" = "Kode2_250m2")) %>%

  mutate(
    # apply translations
    M005_mo_1m2     = coalesce(HT_1m2, M005_mo_1m2),
    M005_mo_250m2   = coalesce(HT_250m2, M005_mo_250m2),

    M020_mo_1m2     = coalesce(HT_1m2, M020_mo_1m2),
    M020_mo_250m2   = coalesce(HT_250m2, M020_mo_250m2),

    sikkerhet_mo_1m2   = if_else(!is.na(HT_1m2), "Svært god oversettelse", sikkerhet_mo_1m2),
    sikkerhet_mo_250m2 = if_else(!is.na(HT_250m2), "Svært god oversettelse", sikkerhet_mo_250m2)
  ) %>%

  select(-starts_with("HT_"), -starts_with("Kode"))
```

For de resterende typene må det gjøres manuelle oversettelser.

Tabell 38: NiN 2.3 HT til NiN 3.0 HT

NiN 2.3 HT	NiN 3.0 HT
T18	TE03,TE04,TE08
T29	TC03,TG01
T38	TI01,TM06
T39	TM01,TM02
T45	TO03,TO04
V3	VC01,VF01
V4	VC02,VC03
V12	VM01,VM04

T18 Åpen flomfastmark

I NiN 3.0 oversettes denne hovedtypen til TE03 Åpen flomfastmark, TE04 Langvarig oversvømt flommark og TE08 Flommarkseng. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den første som forekommer kronologisk.

```
KE <- "T18"
M005_KE <- M020_KE <- "TE03"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T29 Grus- og steindominert strand og strandlinje

I NiN 3.0 oversettes denne hovedtypen til TC03 Løsmasse-strand og TG01 Nakne løsmasser. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den første som forekommer kronologisk.

```
KE <- "T29"
M005_KE <- M020_KE <- "TC03"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T38 Treplantasje

I NiN 3.0 oversettes denne hovedtypen til TI01 Klart endret skogsmark og TM06 Sterkt endret skogsmark. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den første som forekommer kronologisk.

```
KE <- "T38"
M005_KE <- M020_KE <- "TI01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T39 Hard sterkt endret og ny fastmark i langsom suksesjon [hard sterkt endret fastmark]

I NiN 3.0 oversettes denne hovedtypen til TM01 Hard sterkt endret fastmark og TM02 Ny hard fastmark på tørrlagt ferskvannsbunn. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den første som forekommer kronologisk.

```
KE <- "T39"
M005_KE <- M020_KE <- "TM01"

ano_type_list <- lapply(
  ano_type_list,
```

```
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T45 Oppdyrket varig eng

I NiN 3.0 oversettes denne hovedtypen til TO03 Oppdyrket varig eng og TO04 Upløyd jordbruksmark med intensivt hevdpreg. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den første som forekommer kronologisk.

```
KE <- "T45"
M005_KE <- M020_KE <- "TO03"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

V3 Nedbørsmyr

I NiN 3.0 oversettes denne hovedtypen til VC01 Åpen nedbørsmyr og VF01 Nedbørsmyr-skogsmark. Vi bruker tresjiktsdekning i sirkel for å si om en type er VC01 eller VF01, hvor en tresjiktdekning > 10 % gir VF01. Vi antar at vegetasjonsruta også dekkes av eventuell tresjiktsdekning i sirkel.

```
KE <- "V3"
M005_tresatt <- M020_tresatt <- "VF01"
M005_ikketresatt <- M020_ikketresatt <- "VC01"

# Kjør funksjonen
ano_type_list$type2024 <- sett_tresjikt(ano_type_list$type2024,
                                        KE = KE,
                                        M005_tresatt = M005_tresatt,
                                        M005_ikketresatt = M005_ikketresatt,
```

```
                                      M020_tresatt = M020_tresatt,
                                      M020_ikketresatt = M020_ikketresatt)

ano_type_list$type2025 <- sett_tresjikt(ano_type_list$type2025,
                                      KE = KE,
                                      M005_tresatt = M005_tresatt,
                                      M005_ikketresatt = M005_ikketresatt,
                                      M020_tresatt = M020_tresatt,
                                      M020_ikketresatt = M020_ikketresatt)
```

V4 Kaldkilde

I NiN 3.0 oversettes denne hovedtypen til VC02 Torvmarkskilde og VC03 Grunnkilde. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den første som forekommer kronologisk.

```
KE <- "V4"
M005_KE <- M020_KE <- "VC01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

V12 Grøftet torvmark

I NiN 3.0 oversettes denne hovedtypen til VM01 Sterkt endret torvmark og VM04 Sterkt endret, ikke torvproduserende våtmark. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den første som forekommer kronologisk.

```
KE <- "V12"
M005_KE <- M020_KE <- "VM01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
```

```
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

Kartleggingsenheter

Her må det gjøres manuelt arbeid for de resterende KE-ene som mangler oversettelser.

```
uoversatte_KEer <- uoversatte_typer %>%
  filter(str_detect(KE_NiN2_3, "C"))
```

T17-C-1 Jordskred

Denne typen oversettes ikke til NiN 3.0. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den som er nærmest i systemet (TE02-M005-01/TE02-M020-01 Leir- og siltskred).

```
KE <- "T17-C-1"
M005_KE <- "TE02-M005-01"
M020_KE <- "TE02-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T27-C-1 Kalkfattig og intermediær blokkmark

Denne typen oversettes ikke til NiN 3.0. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den som er nærmest i systemet, altså T27-C-2 som har samme kalktrinn, men er ikke 0 i SV. Denne typen oversettes til TG01-M005-12/TG01-M020-06.

```
KE <- "T27-C-1"
M005_KE <- "TG01-M005-12"
M020_KE <- "TG01-M020-06"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T27-C-3 Kalkrik blokkmark

Denne typen oversettes ikke til NiN 3.0. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor her typen til å være den som er nærmest i systemet, altså T27-C-4 som har samme kalktrinn, men er ikke 0 i SV. Denne typen oversettes til TG01-M005-13/TG01-M020-07.

```
KE <- "T27-C-3"
M005_KE <- "TG01-M005-13"
M020_KE <- "TG01-M020-07"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

T38-C-1 Plantasjeskog

Denne typen oversettes ikke til NiN 3.0, men siden T38 oversettes til TI01 og TM06 i Halvorsen (2025), antar vi at T38-C-1 kan oversettes til KE-nivået også. Det er ingen informasjon i ANO-datasettet som kan brukes som hjelp i oversettelsen, og vi setter derfor typen til å være den første i kronologisk rekkefølge, altså TI01-M005-01/TI01-M020-01 Grøftet eller markforstyrret klart endret friskere ikke-kalkrik skog.

```
KE <- "T38-C-1"
M005_KE <- "TI01-M005-01"
M020_KE <- "TI01-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

Andre registreringer

Det er også registrert én grunntype (L4-1) og "ikke_kartlagt".

Siden det kun er én grunntype som må oversettes manuelt, gjøres det her totalt manuelt. L4-1 Kalkfattig helofyttsump i NiN 2.3 oversettes til LB01-01 i NiN 3.0. Denne oversettes deretter til LB01-M005-01 og LB01-M020-01.

```
KE <- "L4-1"
M005_KE <- "LB01-M005-01"
M020_KE <- "LB01-M020-01"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
  M020_KE = M020_KE
)
```

"ikke_kartlagt" blir bare flyttet over.

```
KE <- M005_KE <- M020_KE <- "ikke_kartlagt"

ano_type_list <- lapply(
  ano_type_list,
  sett_manuell_koder,
  KE = KE,
  M005_KE = M005_KE,
```

```
  M020_KE = M020_KE
)
```

7.8 Rydd i ferdig datasett

Hvis de manuelle oversettelsene har en bedre sikkerhet enn de automatiske (f.eks. god oversettelse i de manuelle, men dårlig i de automatiske), så erstatter vi de manuelle oversettelsene med de automatiske.

```
ano_type_ferdig <- ano_type_list

oppdater_ano_dfs <- function(df) {

  sikkerhet_nivå <- c(
  "Svært god oversettelse",
  "God oversettelse",
  "Akseptabel oversettelse",
  "Dårlig oversettelse"
)
  df %>% mutate(
    # Lag en ny kolonne hvor både automatiske oversettelser og manuelle legges til
    oversettelse_type_1m2 = case_when(
      is.na(kartleggingsenhet_1m2_oppd) ~ NA_character_,
      is.na(M005_mo_1m2) ~ "auto",
      TRUE ~ "manuell"
    ),

    oversettelse_type_250m2 = case_when(
      is.na(kartleggingsenhet_250m2_oppd) ~ NA_character_,
      is.na(M005_mo_250m2) ~ "auto",
      TRUE ~ "manuell"
    ),

    # Kopier verdier: manuell hvis finnes, ellers auto
    M005_1m2   = if_else(is.na(M005_mo_1m2),  M005_kode_1m2,  M005_mo_1m2),
    M005_250m2 = if_else(is.na(M005_mo_250m2), M005_kode_250m2, M005_mo_250m2),
    M020_1m2   = if_else(is.na(M020_mo_1m2),  M020_kode_1m2,  M020_mo_1m2),
    M020_250m2 = if_else(is.na(M020_mo_250m2), M020_kode_250m2, M020_mo_250m2),

    # Sikkerhet ------------------------------
    oversettelse_sikkerhet_1m2 = case_when(
```

```
      # Hvis sikkerhet_mo_1m2 mangler, bruk sikkerhet_1m2
      is.na(sikkerhet_mo_1m2) ~ sikkerhet_1m2,
      # Erstatt manuelle oversettelser med automatiske hvis de automatiske har en dårligere score
      match(sikkerhet_mo_1m2, sikkerhet_nivå) < match(sikkerhet_1m2, sikkerhet_nivå) ~
       ↪  sikkerhet_1m2,
      TRUE ~ sikkerhet_mo_1m2
    ),
    oversettelse_sikkerhet_250m2 = case_when(
      # Hvis sikkerhet_mo_1m2 mangler, bruk sikkerhet_1m2
      is.na(sikkerhet_mo_250m2) ~ sikkerhet_250m2,
      # Erstatt manuelle oversettelser med automatiske hvis de automatiske har en dårligere score
      match(sikkerhet_mo_250m2, sikkerhet_nivå) < match(sikkerhet_250m2, sikkerhet_nivå) ~
       ↪  sikkerhet_250m2,
      TRUE ~ sikkerhet_mo_250m2
    )
  ) %>%
  # Endre navn på kolonner som viser alle mulige M005 og M020 oversettelser
  rename(oversettelse_M005_1m2 = M005_kode_1m2,
         oversettelse_M005_250m2 = M005_kode_250m2)
}

# Kjør funksjon
ano_type_ferdig$type2024 <- oppdater_ano_dfs(ano_type_ferdig$type2024)
ano_type_ferdig$type2025 <- oppdater_ano_dfs(ano_type_ferdig$type2025)
```

En del kolonner er blitt laget underveis og som ikke trengs i det ferdige datasettet. De nye kolonnene som skal være med er de som inneholder:

- NiN 3.0 KE-er i målestokk 1:5000 (M005) og i 1:20 000 (M020) for vegetasjonsruter (1 m2) og i sirkel (250 m2). Disse heter M005_mo_1m2, M005_mo_250m2, M020_mo_1m2 og M020_mo_250m2.
- Informasjon om oversettelsen, både hvilken type oversettelse som er gjort på vegetasjonsrute- og sirkelnivå (oversettelse_type_1m2 og oversettelse_type_250m2) og den antatte usikkerheten oversettelsen har (oversettelse_sikkerhet_1m2 og oversettelse_sikkerhet_250m2).
- Mulige NiN 3.0 typer som NiN 2.3 typene be oversatt til i prosessen (oversettelse_M005_1m2 og oversettelse_M005_250m2).

```
col_keep <- function(df,
                     start_col = "globalid", end_col   = "medobservatoer") {
```

```
  cols_1m2 <- c(
    "M005_mo_1m2",
    "M020_mo_1m2",
    "oversettelse_type_1m2",
    "oversettelse_sikkerhet_1m2",
    "oversettelse_M005_1m2"
  )

    cols_250m2 <- c(
    "M005_mo_250m2",
    "M020_mo_250m2",
    "oversettelse_type_250m2",
    "oversettelse_sikkerhet_250m2",
    "oversettelse_M005_250m2"
  )

  # Keep only columns that actually exist in df
  cols <- intersect(c(cols_1m2, cols_250m2), names(df))

  df %>%
    # 1. Keep the main block
    select(all_of(start_col):all_of(end_col),
           all_of(cols)) %>%

    # 2. Reorder extra columns: 1m2 first, then 250m2
    relocate(
      all_of(cols_1m2),
      .after = all_of(end_col)
    ) %>%
    relocate(
      all_of(cols_250m2),
      .after = last_col()
    )

}

# Kjør funksjon
ano_type_ferdig$type2024 <- col_keep(ano_type_ferdig$type2024, start_col = "globalid", end_col
↪  = "medobservatoer")
ano_type_ferdig$type2025 <- col_keep(ano_type_ferdig$type2025, start_col = "globalid", end_col
↪  = "long_unedit")
```

7.9 Legg til beskrivelsesvariabler

Legger her til de oversatte beskrivelsesvariablene.

```
ano2024_bv <- read.csv("data/ano2024_bv.csv")
ano2025_bv <- read.csv("data/ano2025_bv.csv")

ano_type_ferdig$type2024 <- ano_type_ferdig$type2024 %>%
  left_join(ano2024_bv, by="globalid")
ano_type_ferdig$type2025 <- ano_type_ferdig$type2025 %>%
  left_join(ano2025_bv, by="globalid")
```

7.10 Eksporter

Eksporter de ferdige datasettene.

```
ano_type_ferdig$type2024 %>% st_write("data/ano_2018_2024_surveyPoint_oversatt.geojson",
↪  append=F)
ano_type_ferdig$type2025 %>% st_write("data/ano_2025_surveyPoint_oversatt.geojson", append=F)
```

Referanser

Halvorsen, R. 2025. Oversettelse mellom natursystem-typesystemene i NiN (Natur i Norge) versjonene 2.0 (2015), 2.3 (2021) og 3.0 (2023). - Natur i Norge Systemdokumentasjon 2: 1-232.