Taxonomic backbone for sPlot 2.1 and TRY 3.0

Oliver Purschke

12 August, 2017

Abstract

This document describes the workflow (with contributions from Jürgen Dengler and Florian Jansen) that was used to generate the taxonomic backbone that standardizes taxon names across the (i) global vegetation plot database sPlot version 2.1 and (ii) the global plant trait data base TRY version 3.

Contents

1 Combine taxon names from sPlot 2.0 and TRY 3.0

1.1 Load required packages

```
library(stringr)
library(knitr)
library(vegdata)
library(doParallel)
library(dplyr)
library(plyr)
library(Taxonstand)
library(rgbif)
```

1.2 Read in taxon names from sPlot and TRY

1.2.1 Reading sPlot Species

```
path.sPlot <- "/))
home/oliver/Dokumente/PhD/PostPhD/IDiv/sDiv/sPlot/Analyses/Data/Species/
sPlot/sPlot_14_04_2015/"
sPlot.version <- "sPlot_14_4_2015_species"

splot.species <- read.csv(paste0(path.sPlot, sPlot.version, ".csv"), sep = "\t")
gc()
```

```
head(splot.species)
```

```
##   PlotObservationID Taxonomy Taxon.group Taxon.group.ID
## 1                 1 Pasl2012     Unknown              0
## 2                 1 Pasl2012     Unknown              0
## 3                 1 Pasl2012     Unknown              0
## 4                 1 Pasl2012     Unknown              0
## 5                 1 Pasl2012     Unknown              0
## 6                 1 Pasl2012     Unknown              0
##                  Turboveg2.concept                Matched.concept Match
## 1     Allantoparmelia almquistii     Allantoparmelia almquistii     0
## 2             Andreaea rupestris             Andreaea rupestris     0
## 3       Arctoparmelia centrifuga       Arctoparmelia centrifuga     0
## 4             Cetraria nigricans             Cetraria nigricans     0
## 5           Cladonia amaurocraea           Cladonia amaurocraea     0
## 6 Cladonia arbuscula subsp. lat. Cladonia arbuscula subsp. lat.     0
##   Original.taxon.concept Layer Cover.. Cover.code x_
## 1                            0       1          1 NA
## 2                            0       2          2 NA
## 3                            0       5          5 NA
## 4                            0       2          2 NA
## 5                            0       1          1 NA
## 6                            0       2          2 NA
```

Use the Matched.concept column (Federhen 2010), as it already contains some standardization by Stephan Hennekkens according to synbiosys[1].

```
splot.species$Matched.concept<- as.character(splot.species$Matched.concept)
```

1.2.2 Read in TRY 3.0 species list

```
try3.species <- read.csv("/home/oliver/Dokumente/PhD/PostPhD/IDiv/sDiv/sPlot/Analyses/Data/
Species/TRY/Species/TRY30_Gapfilling2015_Species.csv")
```

```
head(try3.species)
```

```
##   AccSpeciesID                Species
## 1            1                  Aa sp
## 2            2     Abarema adenophora
## 3            3    Abarema barbouriana
## 4            4 Abarema brachystachya
## 5            6        Abarema jupunba
## 6            7          Abarema laeta
```

1.3 Combine species lists

```
spec.list.TRY.sPlot <- sort(unique(c(as.character(splot.species$Matched.concept),
                                     as.character(try3.species$Species))))
length(spec.list.TRY.sPlot)
```

Create dataframe that will later hold the original (uncleaned) names in sPlot and TRY:

```
spec.list.TRY.sPlot.2 <- cbind(spec.list.TRY.sPlot, spec.list.TRY.sPlot,
                               spec.list.TRY.sPlot)
str(spec.list.TRY.sPlot.2)
dim(spec.list.TRY.sPlot.2)
spec.list.TRY.sPlot.2 <- as.data.frame(spec.list.TRY.sPlot.2)
```

Convert factors into characters:

[1]`sPlot_14_4_2015_species` no longer exists, but will be used for documentation purposes here. This is the version of splot to generate the backbone for to match splot 2.0 to try 3.0. The following workflow describes the steps needed for generating a backbone for `sPlot_14_4_2015` and `TRY 3.0`. The backbone used in the third sPlot workshop contains the 7,701 additional (and/or different) species names in sPlot 2.1 (`sPlot_2015_07_19_species`) that had to be standardized in a ad-hoc fashion prior to the workshop.

```
spec.list.TRY.sPlot.2[,1] <- as.character(spec.list.TRY.sPlot.2[,1])
spec.list.TRY.sPlot.2[,2] <- as.character(spec.list.TRY.sPlot.2[,2])
spec.list.TRY.sPlot.2[,3] <- as.character(spec.list.TRY.sPlot.2[,3])
```

Give column names[2]:

```
colnames(spec.list.TRY.sPlot.2) <- c("names.sPlot.TRY", "names.corr.string", "sPlot.TRY")
write.csv(spec.list.TRY.sPlot.2, file = "spec.list.TRY3.sPlot2.csv")
```

1.4 Add species' database affiliation

Species in sPlot 2.0:

```
spec.list.TRY.sPlot.2$sPlot.TRY[which(spec.list.TRY.sPlot.2$names.sPlot.TRY %in%
                                      unique(splot.species$Matched.concept))] <- "S"
```

Species in TRY 3.0:

```
spec.list.TRY.sPlot.2$sPlot.TRY[which(spec.list.TRY.sPlot.2$names.sPlot.TRY %in%
                                      unique(try3.species$Species))] <- "T"
```

Species that are both in sPlot 2.0 and TRY 3.0

```
spec.list.TRY.sPlot.2$sPlot.TRY[which(spec.list.TRY.sPlot.2$names.sPlot.TRY %in%
                                      unique(try3.species$Species) &
                                      spec.list.TRY.sPlot.2$names.sPlot.TRY %in%
                                      unique(splot.species$Matched.concept))] <- "ST"
```

2 A-priori cleaning of names

2.1 Manual cleaning

Based one the column `Matched.concept` in `sPlot`, a list of 4,093 "weird" species names (consisting mainly of trivial names) was generated, and corrected manually by Jürgen Dengler (thereafter JD). Some examples:

- Gras silbrig Bl haarig 133106 → Poaceae sp. [silbrig Bl haarig 133106]
- Ãfâ€"Achnella species → Achnella sp.
- [KHH Composite (breite Bl.)] → Asteraceae sp. [breite Bl.]
- LICH Xanthomaculina hottentotta → Xanthomaculina hottentotta
- cf. Silberknospe 134249 → Spermatophyta sp. [Silberknospe 134249]

2.2 String manipulation routines

Stripping unwanted characters as well as abbreviation (such as hybrid markers) which would prevent name matching:

```
OriginalNames <- gsub('*', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('cf. ', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('Cf. ', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('[', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(']', '', OriginalNames, fixed=TRUE)
```

[2]This species list only contains 122,901 species. The final backbone contains 7,701 additional (and/or different) species from from the ad-hoc version `sPlot_2015_07_19`.

```
OriginalNames <- gsub(' x ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('Ã-', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('aff ', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('(', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(')', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(' cf ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(' aff. ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('c,e', 'ceae', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('    ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('   ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('  ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('x-', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('X-', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('like ', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(',', '', OriginalNames, fixed=TRUE)
```

For all names, that have a number in their first word, and consist of > 1 words, remove that word:

```
library(stringr)
firstWordWithNumbers <- grepl('[0-9]', word(OriginalNames, 1))
numberOfWords <- sapply(gregexpr("\\W+", OriginalNames), length) + 1
OriginalNames[firstWordWithNumbers & numberOfWords > 1] <-
    sapply(OriginalNames[firstWordWithNumbers & numberOfWords > 1],
           function(x) substr(x, start=regexpr(pattern =' ', text=x)+1, stop=nchar(x)))
```

Correct some name abbreviations using `taxname.abbr` in `vegdata`:

```
CleanedNames <- taxname.abbr(OriginalNames)
save(CleanedNames, file = "CleanedNames.Rdata")
write.csv(CleanedNames, file = "CleanedNames.csv")
length(CleanedNames)
```

Create a data frame from the string cleaned names:

```
CleanedNames.df <- as.data.frame(CleanedNames, stringsAsFactors=FALSE)
spec.list.TRY.sPlot.3 <- cbind(spec.list.TRY.sPlot.2[,c(1,2,4)], CleanedNames.df,
                               CleanedNames.df)
```

Skip column `names.corr.string`, as the `cleanednames.df` column as well as another column that will include the names corrected by JD.

```
colnames(spec.list.TRY.sPlot.3) <- c("index","original.names.sPlot.TRY","sPlot.TRY",
                                     "CleanedNames","CleanedNames.Juergen")
Tax_Back_sPlot2_TRY3 <- spec.list.TRY.sPlot.3
save(Tax_Back_sPlot2_TRY3, file = "Tax_Back_sPlot2_TRY3.Rdata")
load("Tax_Back_sPlot2_TRY3.Rdata")
write.csv(Tax_Back_sPlot2_TRY3, file = "Tax_Back_sPlot2_TRY3.csv")
head(Tax_Back_sPlot2_TRY3)
```

2.2.1 Substitute a-priorily cleaned names with corrected names

```
names.correct.JD <- read.csv("weird.names_JD_string_correct.csv")
str(names.correct.JD)
head(names.correct.JD)
```

```r
index <- match(names.correct.JD$weird.names, Tax_Back_sPlot2_TRY3$original.names.sPlot.TRY)
Tax_Back_sPlot2_TRY3$CleanedNames.JD[index] <- names.correct.JD$Name.corrected
```

2.2.2 Do some more cleaning on `CleanedNames.JD`

This was done manually in a csv-file.

```r
CleanedNames.JD <- Tax_Back_sPlot2_TRY3$CleanedNames.JD
write.csv(CleanedNames.JD, file = "CleanedNames.JD.csv")
```

2.2.3 Substitute old `CleanedNames.JD` with new string-manipulated one:

```r
Tax_Back_sPlot2_TRY3$CleanedNames.JD <- CleanedNames.JD
save(Tax_Back_sPlot2_TRY3, file = "Tax_Back_sPlot2_TRY3.Rdata")
```

3 Match names against Taxonomic Name Resolution Service (TNRS)

3.1 Slice `CleanedNames` into chunks

... of 5000 species:

```r
seq1 <- seq(from =1, to = 120001, 5000)
seq2 <- seq(from =5000, to = 125000, 5000)

for(i in 1:length(seq1)) {
    write.csv(Tax_Back_sPlot2_TRY3$CleanedNames.JD[seq1[i]:seq2[i]],
              file = paste(paste("tnrs_submit", seq1[i], sep = "_"), "csv", sep = "."))
}
```

The single csv-files, containing 5,000 names each, were submitted to Taxonomic Name Resolution Service web application (Boyle *et al.* 2013, iPlant Collaborative (2015)). TNRS version 4.0 was used, which became available in August 2015 (this version also included The Plant List version 1.1).

3.2 TNRS settings

The following settings were used for resolving names on TNRS.

3.2.1 Sources for name resolution

The initial TNRS name resolution run was based on the **five standard sources** that were **ranked according to preference** in the following order (default of TNRS):

1. The Plant List (TPL)(The Plant List 2013)
2. The Global Compositae Checklist (GCC)(Flann 2009)
3. The International Legume Database and Information Service (ILDIS)(International legume database and information service 2006)
4. Tropicos (Missouri Botanical Garden 2013)
5. PLANTS Database (USDA)(USDA, NRCS 2012)

Because it is possible that the best match is found in lower ranked sources, see section TNRS settings, two additional name resolution runs were realized in which the highest ranking was given to **(1) Tropicos**, or **(2)** the sixth source available in TNRS, **NCBI** (The National Center for Biotechnology Information's Taxonomy database; (Federhen 2010)), respectively, see section TNRS settings.

3.2.2 Family Classification

Resolved names were assigned to families based on the APGIII classification (Chase & Reveal 2009), the same classification system used by Tropicos.

3.3 Retrieve results

Once the matching process was finished, results were retrieved from TNRS using the `Detailed Download` option that included the full name information (parsed components, warnings, links to sources, etc.). We retrived the single best match for each species, constrained by source (TNRS default), where the name in the first source was selected as best match, unless there was `no suitable match found` in that source, the match from the next lower-ranked source was selected, until all resources where exhausted.

4 Manually inspecting name matching results

Manually inspect the TNRS-results table in a spreadsheat application (i.e. LibreOffice or Excel). Starting with the highest taxonomic rank considered (i.e. Family). For instance, if manual checking of the TRNS output reveals that all accepted names or synonyms that have accuracy scores >0.9 are correct taxon names, use the following selection procedure:

- Name_matched_rank (==Family)
- Taxonomic_status (==Accepted, Synomyn)
- Family_score (>0.9)

Continue this selection procedure for entries that were matched at lower taxonomic ranks, i.e. genus, species, etc..

4.1 Read and combine TNRS result files

4.1.1 Set up `doParallel` cluster to speed up the reading process

Find out how many cores are available:

```
detectCores()
```

```
## [1] 4
```

Create cluster with desired number of cores:

```
cl <- makeCluster(3)
```

Register cluster:

```
registerDoParallel(cl)
```

Find out how many cores are being used:

```
getDoParWorkers()
```

4.1.2 Read and combine the single files

Read the downloaded TNRS files, including 5000 names each[3] into R.

```
setwd("/home/oliver/Downloads/")
seq1 <- seq(from =1, to = 120001, 5000)
system.time(
    x <- foreach(i = 1:length(seq1), .combine = rbind) %dopar% {
        read.csv(paste(paste("/home/oliver/Downloads/tnrs.tpl", seq1[i], sep = "."),
                       "csv", sep = "."), sep = ",", stringsAsFactors = FALSE, skip = 0,
                 head = T)[-1, ]
    }
)
tnrs.tpl <- x
```

Assign index to first column:

```
tnrs.tpl$Name_number <- 1:length(tnrs.tpl$Name_number)
```

4.2 Select correctly resolved names

4.2.1 General procedure

1. Open `tnrs.tpl` in a spread sheat program and sort according to `Name_matched_rank`, `Taxonomic_status` and `Family_score`[4].
2. Repeat selection for entries matched at lower taxonomic ranks, such `Name_matched_rank` ==:
   - forma
   - genus
   - infraspecies
   - ...
3. Adjust accuracy score threshold values, e.g. use higher or lower values for infraspec., variety, ...

4.2.2 Family level

Manually inspect sorted table and select all entries at the highest hierarchical level (family). Manually identify the family accuracy score threshold value above which a name can be considered a correct name. In the following case, this corresponds to a score $>$0.88.

4.2.2.1 Create index for correct names

```
index.family <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "family" &
                               (tnrs.tpl$Taxonomic_status == "Accepted" |
                                tnrs.tpl$Taxonomic_status == "Synonym") &
                                tnrs.tpl$Family_score > 0.88), 1]
length(index.family)
```

[3]These were temporary files that were deleted after the name matching procedure.

[4]The full original `tnrs.tpl` table, including all 123,000+ species, has been overwritten, and now instead only contains the 121,000+ species from sPlot 2.1 and TRY 3.0.

4.2.3 Forma level

```r
index.forma <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "forma"), 1]
length(index.forma)
```

4.2.4 Genus level

```r
index.genus <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "genus" &
                              tnrs.tpl$Taxonomic_status == "Accepted" &
                              tnrs.tpl$Genus_score > 0.83), 1]
length(index.genus)
```

4.2.5 Infraspecies level

```r
index.infraspec <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "infraspecies"), 1]
length(index.infraspec)
```

4.2.6 Species level

```r
index.species <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "species" &
                                (tnrs.tpl$Taxonomic_status == "Accepted" |
                                 tnrs.tpl$Taxonomic_status == "Synonym") &
                                 tnrs.tpl$Genus_score > 0.78 &
                                 tnrs.tpl$Name_score > 0.93), 1]
length(index.species)
```

4.2.7 Subspecies level

```r
index.subspec <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "subspecies" &
                               (tnrs.tpl$Taxonomic_status == "Accepted" |
                                tnrs.tpl$Taxonomic_status == "Synonym")), 1]
length(index.subspec)
```

4.2.8 Variety level

```r
index.variety <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "variety" &
                               (tnrs.tpl$Taxonomic_status == "Accepted" |
                                tnrs.tpl$Taxonomic_status == "Synonym")), 1]
length(index.variety)
```

4.2.9 Identifying “non-matched” species that are spermatophyta

```r
index.spermatophyt <- tnrs.tpl[which(tnrs.tpl$Name_matched == "No suitable matches found."
                                     & word(tnrs.tpl$Name_submitted, 1) == "Spermatophyta")
```

```
                              , 1]
length(index.spermatophyt)
```

4.3 Select certain or uncertain names

Select names that do not fulfill the search criteria, i.e. that were not selected as certain species, for further name matching.

```
index.tpl <- c(index.family, index.forma, index.genus, index.species, index.subspec,
               index.variety, index.spermatophyt)
length((index.tpl))

tnrs.tpl.certain <- tnrs.tpl[index.tpl,]
dim(tnrs.tpl.certain)
save(tnrs.tpl.certain, file = "tnrs.tpl.certain.Rdata")
write.csv(tnrs.tpl.certain, file = "tnrs.tpl.certain.csv")

tnrs.tpl.uncertain <- tnrs.tpl[tnrs.tpl$Name_number %in% index.tpl == F, ]
dim(tnrs.tpl.uncertain)
save(tnrs.tpl.uncertain, file = "tnrs.tpl.uncertain.Rdata")
write.csv(tnrs.tpl.uncertain, file = "tnrs.tpl.uncertain.csv")
```

Generate list of uncertain species that are still to be resolved on TNRS:

```
write.csv(tnrs.tpl.uncertain[,2], file = "tnrs.tpl.uncertain.upload.csv")
```

4.4 Resolve 'uncertain' names using Tropicos

Because the matching procedure above, giving the highest ranking to the sources TPL, GCC and ILDIS (which seem to be the most reliable and up-to-date), will not always result in the match, in cases where better matching scores are achieved in lower ranked sources. Therefore the TNRS matching procedure was continued for the `uncertain` species from the previous step, but assigning the highest rank to the source `Tropicos`, while repeating the steps that were used to generate `tnrs.tpl`.[5]

4.4.1 Read in 'TNRS-Tropicos' results

```
setwd("/home/oliver/Downloads/")

system.time(
    x <- foreach(i = 1:length(seq1), .combine = rbind) %dopar% {
        read.csv(paste(paste("/home/oliver/Downloads/tnrs.trop", seq1[i], sep = "."),
                           "csv", sep = "."), sep = ",", stringsAsFactors = FALSE, skip = 0,
                 head = T)[-1, ]
    }
)

tnrs.trop <- x

str(tnrs.trop)
tnrs.trop$Name_number <- 1:length(tnrs.trop$Name_number)
```

[5]Note, the single temporary result files from TNRS are no longer available.

4.4.2 Reduce to set of uncertain species tnrs.tpl

```r
tnrs.trop.small <- tnrs.trop[tnrs.all.small.uncertain$Name_number, ]
str(tnrs.trop.small)

save(tnrs.trop.small, file = "tnrs.trop.small.Rdata")
write.csv(tnrs.trop.small, file = "tnrs.trop.small.csv")
```

4.5 Selecting correctly resolved names

Using the same procedure as above.

4.5.1 Family level

```r
index.family <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "family" &
                                      (tnrs.trop.small$Taxonomic_status == "Accepted"|
                                       tnrs.trop.small$Taxonomic_status == "Synonym")), 1]
length(index.family)
```

4.5.2 Forma level

```r
index.forma <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "forma" &
                                     (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                      tnrs.trop.small$Taxonomic_status == "Synonym")), 1]
length(index.forma)
```

4.5.3 Genus level

```r
index.genus <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "genus" &
                                     (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                      tnrs.trop.small$Taxonomic_status == "Synonym") &
                                      tnrs.trop.small$Genus_score > 0.83 &
                                      tnrs.trop.small$Name_score > 0.5), 1]
length(index.genus)
```

4.5.4 Species level

```r
index.species1 <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "species" &
                                        (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                         tnrs.trop.small$Taxonomic_status == "Synonym") &
                                         tnrs.trop.small$Genus_score > 0.88 &
                                         tnrs.trop.small$Name_score > 0.9), 1]
length(index.species1)

index.species2 <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "species" &
                                        (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                         tnrs.trop.small$Taxonomic_status == "Synonym") &
```

```
                                          tnrs.trop.small$Genus_score > 0.78 &
                                          tnrs.trop.small$Name_score > 0.94), 1]
length(index.species1)

index.species3 <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "species" &
                                        (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                         tnrs.trop.small$Taxonomic_status == "Synonym") &
                                        tnrs.trop.small$Genus_score > 0.88 &
                                        tnrs.trop.small$Name_score > 0.49), 1]
length(index.species3)

index.species <- unique(c(index.species1, index.species2, index.species3))
length(index.species)
```

4.5.5 Subspecies level

```
index.subspec <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "subspecies" &
                                       (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                        tnrs.trop.small$Taxonomic_status == "Synonym")), 1]
length(index.subspec)
```

4.5.6 Variety level

```
index.variety <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "variety" &
                                       (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                        tnrs.trop.small$Taxonomic_status == "Synonym")), 1]
length(index.variety)
```

4.5.7 Select certain and uncertain names

```
index.trop <- c(index.family, index.forma, index.genus, index.species, index.subspec,
                index.variety)
length((index.trop))
tnrs.trop.small.certain <- tnrs.trop.small[tnrs.trop.small$Name_number %in% index.trop
                                           == T,]
dim(tnrs.trop.small.certain)
save(tnrs.trop.small.certain, file = "tnrs.trop.small.certain.Rdata")
write.csv(tnrs.trop.small.certain, file = "tnrs.trop.small.certain.csv")

tnrs.trop.small.uncertain <- tnrs.trop.small[tnrs.trop.small$Name_number %in% index.trop
                                             == F, ]
dim(tnrs.trop.small.uncertain)
save(tnrs.trop.small.uncertain, file = "tnrs.trop.small.uncertain.Rdata")
write.csv(tnrs.trop.small.uncertain, file = "tnrs.trop.small.uncertain.csv")
```

4.6 Resolve ‘uncertain’ names using NCBI

Cut the list of the remaining 9,641 unresolved species into chunks of $\sim$ 5,000 species.

```
write.csv(tnrs.trop.small.uncertain$Name_submitted[1:5000], file = "trop.uncert.1.csv")
write.csv(tnrs.trop.small.uncertain$Name_submitted[5001:9641], file = "trop.uncert.5001.csv")
```

Match these lists against TNRS using the settings decribed above, but rank the additional (sixth) sources in TNRS, The National Center for Biotechnology Information's Taxonomy database NCBI, first.

4.6.1 Read in 'TNRS-NCBI' results

```
tnrs.ncbi.1 <- read.csv("/home/oliver/Downloads/tnrs.trop.uncert.1.csv",
                        stringsAsFactors = FALSE)[-1,]
tnrs.ncbi.2 <- read.csv("/home/oliver/Downloads/tnrs.trop.uncert.5001.csv",
                        stringsAsFactors = FALSE)[-1,]

tnrs.ncbi <- rbind(tnrs.ncbi.1, tnrs.ncbi.2)
str(tnrs.ncbi)
tnrs.ncbi$Name_number <- tnrs.trop.small.uncertain$Name_number
range(tnrs.ncbi$Name_number)

save(tnrs.ncbi, file = "tnrs.ncbi.Rdata")
write.csv(tnrs.ncbi, file = "tnrs.ncbi.csv")
```

4.7 Selecting correctly resolved names in `TNRS_NCBI`

4.7.1 Family level

```
index.family <- tnrs.ncbi[which(tnrs.ncbi$Name_matched_rank == "family" &
                                (tnrs.ncbi$Taxonomic_status == "Accepted"|
                                 tnrs.ncbi$Taxonomic_status == "Synonym") &
                                tnrs.ncbi$Family_score > 0.85), 1]
length(index.family)
```

4.7.2 Genus level

```
index.genus.1 <- tnrs.ncbi[which(tnrs.ncbi$Name_matched_rank == "genus" &
                                 (tnrs.ncbi$Taxonomic_status == "Accepted" |
                                  tnrs.ncbi$Taxonomic_status == "Synonym") &
                                 tnrs.ncbi$Genus_score > 0.89 &
                                 tnrs.ncbi$Name_score > 0.49), 1]
length(index.genus.1)

index.genus.2 <- tnrs.ncbi[which(tnrs.ncbi$Name_matched_rank == "genus" &
                                 (tnrs.ncbi$Taxonomic_status == "Accepted" |
                                  tnrs.ncbi$Taxonomic_status == "Synonym") &
                                 tnrs.ncbi$Genus_score > 0.99 &
                                 tnrs.ncbi$Name_score > 0.2), 1]
length(index.genus.2)

index.genus.3 <- tnrs.ncbi[which(tnrs.ncbi$Name_matched_rank == "genus" &
                                 tnrs.ncbi$Taxonomic_status == "No opinion" &
```

```r
                                  tnrs.ncbi$Genus_score > 0.88 &
                                  tnrs.ncbi$Name_score > 0.49), 1]
length(index.genus.3)

index.genus <- unique(c(index.genus.1, index.genus.2, index.genus.3))
length(index.genus)
```

4.7.3 Species level

```r
index.species.1 <- tnrs.ncbi[which(tnrs.ncbi$Name_matched_rank == "species" &
                                   (tnrs.ncbi$Taxonomic_status == "Accepted" |
                                    tnrs.ncbi$Taxonomic_status == "Synonym") &
                                   tnrs.ncbi$Name_score > 0.94), 1]
length(index.species.1)

index.species.2 <- tnrs.ncbi[which(tnrs.ncbi$Name_matched_rank == "species" &
                                   (tnrs.ncbi$Taxonomic_status == "Accepted" |
                                    tnrs.ncbi$Taxonomic_status == "Synonym") &
                                   tnrs.ncbi$Genus_score > 0.81 &
                                   tnrs.ncbi$Name_score > 0.51), 1]
length(index.species.2)

index.species.3 <- tnrs.ncbi[which(tnrs.ncbi$Name_matched_rank == "species" &
                                   tnrs.ncbi$Taxonomic_status == "No opinion"  &
                                   tnrs.ncbi$Genus_score > 0.7 &
                                   tnrs.ncbi$Specific_epithet_score > 0.75), 1]
length(index.species.3)

index.species <- unique(c(index.species.1, index.species.2, index.species.3))
length(index.species)
```

4.7.4 Variety level

```r
index.var <- tnrs.ncbi[which((tnrs.ncbi$Name_matched_rank == "subspecies" |
                              tnrs.ncbi$Name_matched_rank == "unknown" |
                              tnrs.ncbi$Name_matched_rank == "variety") &
                             (tnrs.ncbi$Taxonomic_status == "Accepted" |
                              tnrs.ncbi$Taxonomic_status == "No opinion" |
                              tnrs.ncbi$Taxonomic_status == "Synonym")), 1]
length(index.var)

index.ncbi <- c(index.family, index.genus, index.species, index.var)
length((index.ncbi))
```

4.7.5 Select certain or uncertain names

```r
tnrs.ncbi.certain <- tnrs.ncbi[tnrs.ncbi$Name_number %in% index.ncbi == T,]
dim(tnrs.ncbi.certain)
save(tnrs.ncbi.certain, file = "tnrs.ncbi.certain.Rdata")
```

```
write.csv(tnrs.ncbi.certain, file = "tnrs.ncbi.certain.csv")

tnrs.ncbi.uncertain <- tnrs.ncbi[tnrs.ncbi$Name_number %in% index.ncbi == F, ]
dim(tnrs.ncbi.uncertain)
save(tnrs.ncbi.uncertain, file = "tnrs.ncbi.uncertain.Rdata")
write.csv(tnrs.ncbi.uncertain, file = "tnrs.ncbi.uncertain.csv")
```

4.7.6 Resolve remaining uncertain names

`tnrs.ncbi` still contained 1,464 `uncertain` names that were resolved in the following way:

- 679 names (mainly trivial names) were selected manually and corrected by JD.
- of those 679 names, 62 were corrected using the matching tools on the TPL webpage (e.g. `Dicra vagin var. clathrata` → `Dicranella vaginata`).
- the remaining 785 names were checked manually.

```
JD.679 <- read.csv("JD.679.csv")
head(JD.679)
```

Manually correct, in a spread sheat program, some further mispellings (e.g. `ABIESNORD.`, `xboris`). Select `non-JD.corrected` and correct species in `tnrs.ncbi.uncertain.corrected.csv`[6].

```
ncbi.uncertain.corr <- read.csv("/home/oliver/Downloads/tnrs.ncbi.uncertain.corrected.csv")

str(ncbi.uncertain.corr)
```

Select the 679 JD-species, incl. the 62 species that were corrected manually.

```
index.JD.corrected <- ncbi.uncertain.corr[which((ncbi.uncertain.corr$for_JD ==
                                               "x" |
                                               ncbi.uncertain.corr$corrected !=
                                               "")), 2]
length(index.JD.corrected)
```

Further select some correct species within genus

Some species were cut down to the genus level by TNRS, although they might be resolvable using the name matching tools on the TPL webpage. Select those names for further name matching.

```
index.correct.genus <- ncbi.uncertain.corr[which((ncbi.uncertain.corr$Name_matched_rank ==
                                                "genus" &
                                                ncbi.uncertain.corr$Taxonomic_status ==
                                                "Accepted" &
                                                ncbi.uncertain.corr$Overall_score > 0.6))
                                         , 2]
length(index.correct.genus)
```

Further select some correct names that were matched at the rank of species

```
index.correct.species <- ncbi.uncertain.corr[which((ncbi.uncertain.corr$Name_matched_rank ==
                                                  "species" &
                                                  ncbi.uncertain.corr$Taxonomic_status ==
                                                  "Accepted" &
                                                  ncbi.uncertain.corr$Overall_score > 0.89))
```

[6] Temporary file that no longer exists.

```
                                               , 2]
length(index.correct.species)
```

Create an index for those names that could be further corrected:

```
index.ncbi <- unique(c(index.JD.corrected, index.correct.genus, index.correct.species))
length(index.ncbi)
```

4.7.7 Identify certain and uncertain species

```
ncbi.uncertain.corr.certain <- ncbi.uncertain.corr[ncbi.uncertain.corr$Name_number %in%
                                                    index.ncbi == T,]
dim(ncbi.uncertain.corr.certain)
save(ncbi.uncertain.corr.certain, file = "ncbi.uncertain.corr.certain.Rdata")
write.csv(ncbi.uncertain.corr.certain, file = "ncbi.uncertain.corr.certain.csv")

ncbi.uncertain.corr.uncertain <- ncbi.uncertain.corr[ncbi.uncertain.corr$Name_number %in%
                                                      index.ncbi == F, ]
dim(ncbi.uncertain.corr.uncertain)
save(ncbi.uncertain.corr.uncertain, file = "ncbi.uncertain.corr.uncertain.Rdata")
write.csv(ncbi.uncertain.corr.uncertain, file = "ncbi.uncertain.corr.uncertain.csv")
```

5 Using The Plant List matching tools for unresolved names

Generate names list from `ncbi.uncertain.corr.uncertain` to be matched against `The Plant List`, using `Taxonstand::TPL`.

```
ncbi.uncertain <- as.character(ncbi.uncertain.corr.uncertain$Name_submitted)
```

5.1 Lists of names for TPL

Run "raw" list against TPL:

Set large edit distance to allow for fuzzy matching, and strip some characters that would prevend matching:

```
tpl.ncbi.1 <- TPL(ncbi.uncertain, corr=T, diffchar = 9, max.distance = 9)
tpl.ncbi.1 <- gsub("[", "", tpl.ncbi.1)
tpl.ncbi.1 <- gsub("]", "", tpl.ncbi.1)
tpl.ncbi.1 <- gsub("|", "", tpl.ncbi.1)
tpl.ncbi.1 <- gsub("?", "", tpl.ncbi.1)
write.csv(tpl.ncbi.1, file = "tpl.ncbi.1.csv")
```

Extent each word by *, to allow for even 'fuzzier' matching:

```
ncbi.uncertain.2 <- paste(gsub(" ", "* ", ncbi.uncertain), "*", sep = "")
tpl.ncbi.2 <- TPL(ncbi.uncertain.2, corr=T, diffchar = 9, max.distance = 9)
write.csv(tpl.ncbi.2, file = "tpl.ncbi.2.csv")
```

Truncate each word to its first 3 (5 or 7) letters and extent by *, to do more fuzzy matching

```
simpleCap3 <- function(x) {
  s <- strsplit(x, " ")[[1]]
  paste(paste(substring(s, 1,1), substring(s, 2,3), sep="", collapse="* "), "*", sep = "")
}

simpleCap5 <- function(x) {
  s <- strsplit(x, " ")[[1]]
  paste(paste(substring(s, 1,1), substring(s, 2,5), sep="", collapse="* "), "*", sep = "")
}

simpleCap7 <- function(x) {
  s <- strsplit(x, " ")[[1]]
  paste(paste(substring(s, 1,1), substring(s, 2,7), sep="", collapse="* "), "*", sep = "")
}
```

Apply the string truncation functions above:

```
ncbi.uncertain.3 <- sapply(ncbi.uncertain, simpleCap3)
```

Repeat, but instead using `simpleCap5` or `simpleCap7`. Further, remove some strings to improve name matching:

```
ncbi.uncertain.3 <- gsub("[", "", ncbi.uncertain.3, fixed = T)
ncbi.uncertain.3 <- gsub("]", "", ncbi.uncertain.3, fixed = T)
ncbi.uncertain.3 <- gsub("|", "", ncbi.uncertain.3, fixed = T)
ncbi.uncertain.3 <- gsub("?", "", ncbi.uncertain.3, fixed = T)
ncbi.uncertain.3 <- gsub("+", "", ncbi.uncertain.3, fixed = T)
ncbi.uncertain.3 <- gsub(".", "", ncbi.uncertain.3, fixed = T)
ncbi.uncertain.3 <- gsub("<", "", ncbi.uncertain.3, fixed = T)
ncbi.uncertain.3 <- gsub("/", "", ncbi.uncertain.3, fixed = T)
str(ncbi.uncertain.3)

tpl.ncbi.3 <- TPL(ncbi.uncertain.3, corr=T, diffchar = 9, max.distance = 9)
write.csv(tpl.ncbi.3, file = "tpl.ncbi.3.csv")

tpl.ncbi.5 <- TPL(ncbi.uncertain.5, corr=T, diffchar = 9, max.distance = 9)
write.csv(tpl.ncbi.5, file = "tpl.ncbi.5.csv")

tpl.ncbi.7 <- TPL(ncbi.uncertain.7, corr=T, diffchar = 9, max.distance = 9)
write.csv(tpl.ncbi.7, file = "tpl.ncbi.7.csv")
```

Combine `tpl.ncbi` tables:

```
tpl.ncbi <- cbind(tpl.ncbi.1[,c(1,2,6,8,10,12)], tpl.ncbi.2[,c(6,8,10,12)],
                  tpl.ncbi.3[,c(6,8,10,12)], tpl.ncbi.5[,c(6,8,10,12)],
                  tpl.ncbi.7[,c(6,8,10,12)])

rownames(tpl.ncbi) <- rownames(tpl.ncbi.7)
tpl.ncbi <- cbind(ncbi.uncertain.corr.uncertain[,1:2], tpl.ncbi)
str(tpl.ncbi)
tpl.ncbi <- tpl.ncbi[,-1]
write.csv(tpl.ncbi, file = "tpl.ncbi.csv")
```

Manually select correct genera and species in `tpl.ncbi` csv-file and add columns `Genus.correct` and `Species.correct`.

Read the manually corrected `tpl.ncbi` table:

```
tpl.ncbi.2 <- read.csv("tpl.ncbi.csv")
str(tpl.ncbi.2)
```

```
names(tpl.ncbi.2)
```

Select corrected species and concatenate genus and epithet columns:

```
tpl.ncbi.2$name.correct <- paste(tpl.ncbi.2$Genus.correct, tpl.ncbi.2$Species.correct)
index.corr <- tpl.ncbi.2[which(tpl.ncbi.2$name.correct != " "), 2]
```

Merge `ncbi.uncertain` correspond and `tpl.ncbi.2`

```
ncbi.uncertain.corr.uncertain.2 <- join(ncbi.uncertain.corr.uncertain,
                                        tpl.ncbi.2[,c(1,2,6,26:29)], by = "Name_number")

str(ncbi.uncertain.corr.uncertain.2)
names(ncbi.uncertain.corr.uncertain.2)

write.csv(ncbi.uncertain.corr.uncertain.2, file = "ncbi.uncertain.corr.uncertain.2.csv")
ncbi.uncertain.corr.uncertain.2 <- read.csv("ncbi.uncertain.corr.uncertain.2.csv")
```

5.1.1 Tag names that could not be resolved

If names were not corrected, set `Taxonomic.status == ""`

```
ncbi.uncertain.corr.uncertain.2$Status.correct[
                                    ncbi.uncertain.corr.uncertain.2$Status.correct==""] <-
    ncbi.uncertain.corr.uncertain.2$Taxonomic.status[
                                        ncbi.uncertain.corr.uncertain.2$Status.correct ==""]

summary(ncbi.uncertain.corr.uncertain.2$Status.correct)
str(ncbi.uncertain.corr.uncertain.2$Status.correct)
```

... and assign `No suitable matches found.` to the remaining species:

```
ncbi.uncertain.corr.uncertain.2$Status.correct <-
    as.character(ncbi.uncertain.corr.uncertain.2$Status.correct)
ncbi.uncertain.corr.uncertain.2$Status.correct
[is.na(ncbi.uncertain.corr.uncertain.2$Status.correct)] <- "No suitable matches found."
```

Add uncorrected names in column `X` to `name.correct`:

```
ncbi.uncertain.corr.uncertain.2$name.correct[
                                    ncbi.uncertain.corr.uncertain.2$Genus.correct==""] <-
    as.character(ncbi.uncertain.corr.uncertain.2[,41])[
        ncbi.uncertain.corr.uncertain.2$Genus.correct==""]
```

Assign `No suitable matches found.` to remaining species in `name.correct` according to `Status.correct`.

```
ncbi.uncertain.corr.uncertain.2$name.correct[ncbi.uncertain.corr.uncertain.2$Status.correct==
                                             "No suitable matches found."] <-
    "No suitable matches found."

write.csv(ncbi.uncertain.corr.uncertain.2, file = "ncbi.uncertain.corr.uncertain.2.csv")
```

Done! Use `ncbi.uncertain.corr.uncertain.2` for later merging with the other data sets.

5.1.2 Check `ncbi.certain`

Create extra column `species.correct` and add the 679 species from `status=accepted` in `JD.corrected`. Assign status `No suitable matches found.` to the remaining non-resolved species.

```
JD.correct <- read.csv("/home/oliver/Dokumente/PhD/PostPhD/IDiv/sDiv/sPlot/Analyses/
Code/Unresolved_sPlot2.0_TRY3.0_2_JD.csv", stringsAsFactors=FALSE)
```

5.2 Assign `No suitable matches found.` to non-correctable species

```
JD.correct$Taxon[JD.correct$Taxon==""] <- "No suitable matches found."
str(JD.correct)
str(ncbi.uncertain.corr.certain)
```

Join to `ncbi.uncertain.corr.certain` based on `Name_number`:

```
ncbi.certain.JD.corr <- join(ncbi.uncertain.corr.certain, JD.correct[,c(2,3,8)],
                                   by = "Name_number")
str(ncbi.certain.JD.corr)
write.csv(ncbi.certain.JD.corr, file = "ncbi.certain.JD.corr.csv")
ncbi.certain.JD.corr.2 <- read.csv("ncbi.certain.JD.corr.csv",
                                         stringsAsFactors=FALSE)
str(ncbi.certain.JD.corr.2)
```

Add the corrected names from `ncbi.certain.JD.corr.2$corrected` to `name.correct`:

```
ncbi.certain.JD.corr.2$name.correct[is.na(ncbi.certain.JD.corr.2$name.correct)] <-
    ncbi.certain.JD.corr.2$corrected[is.na(ncbi.certain.JD.corr.2$name.correct)]
write.csv(ncbi.certain.JD.corr.2, file = "ncbi.certain.JD.corr.2.csv")
```

Fill the missing names in `name.correct`, because they were correctly resolved.

```
ncbi.certain.JD.corr.2$name.correct[ncbi.certain.JD.corr.2$name.correct==""] <-
    ncbi.certain.JD.corr.2$Name_matched[ncbi.certain.JD.corr.2$name.correct==""]
```

All entries in `ncbi.certain.JD.corr.2` are assigned to a name. Now merge it with `ncbi.uncertain.corr.uncertain.2` (~1,400 names). Add tag `manual matching`. Means the scores from TNRS matching are useless here, but nevertheless keep them.

```
ncbi.certain.JD.corr.3 <- read.csv("ncbi.certain.JD.corr.3.csv",
                                    stringsAsFactors=FALSE)
str(ncbi.certain.JD.corr.3)
ncbi.uncertain.corr.uncertain.3 <- read.csv("ncbi.uncertain.corr.uncertain.3.csv",
                                            stringsAsFactors=FALSE)
str(ncbi.uncertain.corr.uncertain.3)
```

Add status `No suitable matches found.` or `Accepted` in `ncbi.certain.JD.corr.3`

```
ncbi.certain.JD.corr.3$Status.correct[ncbi.certain.JD.corr.3$name.correct ==
                                           "No suitable matches found."] <-
    "No suitable matches found."
ncbi.certain.JD.corr.3$Status.correct[
                                  is.na(ncbi.certain.JD.corr.3$Status.correct)] <-
    "Accepted"
write.csv(ncbi.certain.JD.corr.3, file = "ncbi.certain.JD.corr.3.csv")
```

In `ncbi.certain.JD.corr.3`, assign all entries an `x` for `Manual matching`:

```r
ncbi.certain.JD.corr.3$Manual.matching <- "x"
ncbi.uncertain.corr.uncertain.3$Manual.matching <- "x"
write.csv(ncbi.certain.JD.corr.3, file = "ncbi.certain.JD.corr.3.csv")
write.csv(ncbi.uncertain.corr.uncertain.3, file = "ncbi.uncertain.corr.uncertain.3.csv")
```

5.2.1 Combine the uncertain and certain species lists

```r
names(ncbi.certain.JD.corr.3)
names(ncbi.uncertain.corr.uncertain.3)

ncbi.certain.JD.corr.3 <- ncbi.certain.JD.corr.3[,-c(1,2,3)]
ncbi.uncertain.corr.uncertain.3 <- ncbi.uncertain.corr.uncertain.3[,-c(1,2)]

match(names(ncbi.uncertain.corr.uncertain.3), names(ncbi.certain.JD.corr.3))

ncbi.uncertain.comb <- rbind(ncbi.uncertain.corr.uncertain.3, ncbi.certain.JD.corr.3)
dim(ncbi.uncertain.comb)
write.csv(ncbi.uncertain.comb, file = "ncbi.uncertain.comb.csv")
```

Read in `tnrs.ncbi.certain`:

```r
tnrs.ncbi.certain <- read.csv("/home/oliver/Downloads/tnrs.ncbi.certain.csv",
                              stringsAsFactors=FALSE)
str(tnrs.ncbi.certain)
```

5.3 Resolve names that were reduced to genus-level by TNRS

Read in `tnrs.ncbi.certain`":

```r
tnrs.ncbi.certain <- read.csv("/home/oliver/Downloads/tnrs.ncbi.certain.csv",
                              stringsAsFactors=FALSE)
str(tnrs.ncbi.certain)
```

To resolve names that were reduced to genus-level, identify `Name_submitted` where `Overall_score` == 0.5:

```r
index0.5 <- tnrs.ncbi.certain[which(tnrs.ncbi.certain$Overall_score == 0.5), 2]
tpl0.5 <- tnrs.ncbi.certain$Name_submitted[tnrs.ncbi.certain$Overall_score == 0.5]
names(tpl0.5) <- index0.5
str(tpl0.5)
length(tpl0.5)
tpl0.5[1:100]
save(tpl0.5, file = "tpl0.5.Rdata")
```

Do some string cleaning to improve matching:

```r
tpl0.5 <- gsub("[", "", tpl0.5, fixed = T)
tpl0.5 <- gsub("]", "", tpl0.5, fixed = T)
tpl0.5 <- gsub("|", "", tpl0.5, fixed = T)
tpl0.5 <- gsub("?", "", tpl0.5, fixed = T)
tpl0.5 <- gsub("+", "", tpl0.5, fixed = T)
tpl0.5 <- gsub(".", "", tpl0.5, fixed = T)
tpl0.5 <- gsub("<", "", tpl0.5, fixed = T)
tpl0.5 <- gsub("/", "", tpl0.5, fixed = T)
```

```r
system.time(
tpl0.5.res <- TPL(tpl0.5, corr=T, diffchar = 2, max.distance = 1)
)
```

Extent each word by *, to allow for even 'fuzzier' matching:

```r
write.csv(tpl0.5.res, file = "tpl0.5.res.csv")
tpl0.5.2 <- paste(gsub(" ", "* ", tpl0.5), "*", sep = "")
tpl0.5.2.res <- TPL(tpl0.5.2, corr=T, diffchar = 2, max.distance = 1)
write.csv(tpl0.5.2.res, file = "tpl0.5.2.res.csv")
```

Truncate each word to its first 5 (or 7) letters and extent words by *, to do more fuzzy matching:

```r
tpl0.5.5 <- sapply(tpl0.5, simpleCap5)
tpl0.5.5.res <- TPL(tpl0.5.5, corr=T, diffchar = 2, max.distance = 1)
write.csv(tpl0.5.5.res, file = "tpl0.5.5.res.csv")

tpl0.5.7 <- sapply(tpl0.5, simpleCap7)
tpl0.5.7.res <- TPL(tpl0.5.7, corr=T, diffchar = 2, max.distance = 1)
write.csv(tpl0.5.7.res, file = "tpl0.5.7.res.csv")

tpl0.5.res <- read.csv("tpl0.5.res.csv", stringsAsFactors=FALSE)
tpl0.5.2.res <- read.csv("tpl0.5.2.res.csv", stringsAsFactors=FALSE)
tpl0.5.5.res <- read.csv("tpl0.5.5.res.csv", stringsAsFactors=FALSE)
tpl0.5.7.res <- read.csv("tpl0.5.7.res.csv", stringsAsFactors=FALSE)
```

Combine `tpl.ncbi` tables:

```r
tpl.res.comb <- cbind(tpl0.5.res[,c(1,2,3,7,9,11,13)],
tpl0.5.2.res[,c(7,9,11,13)], tpl0.5.5.res[,c(7,9,11,13)],
tpl0.5.7.res[,c(7,9,11,13)])

head(tpl.res.comb)
write.csv(tpl.res.comb, file = "tpl.res.comb.csv")
tpl.res.comb <- read.csv("tpl.res.comb.csv", stringsAsFactors=FALSE)
```

5.3.1 Manually add and fill in new columns to `tpl.res.comb.csv`:

- `Status.correct`
- `Genus.correct`
- `Species.correct`

5.3.2 Combine `species.correct` and `genus.correct`

```r
tpl.res.comb <- read.csv("tpl.res.comb.csv", stringsAsFactors=FALSE)
str(tpl.res.comb)
names(tpl.res.comb)
tpl.res.comb$name.correct <- paste(tpl.res.comb$Genus.correct,
tpl.res.comb$Species.correct)
tpl.res.comb.2 <- tpl.res.comb
```

5.3.3 Join tnrs.ncbi.certain and tpl.res.comb.2

```
names(tpl.res.comb.2)
names(tpl.res.comb.2)[2] <- "Name_number"
names(tnrs.ncbi.certain)
```

5.3.4 Get match correct

```
tnrs.ncbi.certain.0.5 <- join(tnrs.ncbi.certain[match(index0.5,
                                                       tnrs.ncbi.certain$Name_number),],
                              tpl.res.comb.2[,c(3,23:26)], by =
"Name_number")
write.csv(tnrs.ncbi.certain.0.5, file = "tnrs.ncbi.certain.0.5.csv")
```

Fill in extra columns in `tnrs.ncbi.certain.0.5`:

```
tnrs.ncbi.certain.0.5 <- read.csv("tnrs.ncbi.certain.0.5.csv", stringsAsFactors=FALSE)
str(tnrs.ncbi.certain.0.5)
names(tnrs.ncbi.certain.0.5)
str(tnrs.ncbi.certain.0.5$Genus.correct)
```

Fill `Manual.matching`:

```
tnrs.ncbi.certain.0.5$Manual.matching[tnrs.ncbi.certain.0.5$Genus.correct != ""] <- "x"
```

Fill `Status.correct`:

```
tnrs.ncbi.certain.0.5$Status.correct[tnrs.ncbi.certain.0.5$Status.correct == ""] <-
    tnrs.ncbi.certain.0.5$Taxonomic_status[tnrs.ncbi.certain.0.5$Status.correct == ""]
```

Fill `name.correct`:

```
tnrs.ncbi.certain.0.5$name.correct[tnrs.ncbi.certain.0.5$name.correct == " "] <-
    tnrs.ncbi.certain.0.5$Name_matched[tnrs.ncbi.certain.0.5$name.correct == " "]

write.csv(tnrs.ncbi.certain.0.5, file = "tnrs.ncbi.certain.0.5.csv")
```

Combine `tnrs.ncbi.certain.0.5` with the remaining `tnrs.ncbi.certain`:

```
str(tnrs.ncbi.certain.0.5)
names(tnrs.ncbi.certain.0.5)

cert.0.5 <- tnrs.ncbi.certain[tnrs.ncbi.certain$Overall_score == 0.5,]
dim(cert.0.5)
str(cert.0.5)
cert.non.0.5 <- tnrs.ncbi.certain[tnrs.ncbi.certain$Overall_score != 0.5,]
dim(cert.non.0.5)
str(cert.non.0.5)
names(cert.non.0.5)
```

Add three more columns to `cert.non.0.5` so that the columns to those in `tnrs.ncbi.certain.0.5`:

```
cert.non.0.5$Manual.matching <- NA
cert.non.0.5$Status.correct <- NA
cert.non.0.5$name.correct <- NA
```

Combine the two tables:

```r
tnrs.ncbi.certain.comb <- rbind(tnrs.ncbi.certain.0.5[,c(3:41,44)], cert.non.0.5[,c(2:41)])
dim(tnrs.ncbi.certain.comb)
write.csv(tnrs.ncbi.certain.comb, file = "tnrs.ncbi.certain.comb.csv")
```

6 Merge the resolved species lists

6.1 Read files

6.1.1 `TPL.small.certain`

Contains TNRS results based on the five sources: TPL, GCC, ILDIS, Tropicos and USDA.

```r
load("tnrs.tpl.certain.Rdata")
dim(tnrs.tpl.certain)
```

6.1.2 `Trop.small.certain`

Contains TNRS results based on the five sources Tropicos (ranked first) TPL, GCC, ILDIS & USDA

```r
load("tnrs.trop.small.certain.Rdata")
dim(tnrs.trop.small.certain)
```

Combine the `certain` data sets:

```r
tnrs.tpl.all.trop.certain <- rbind(tnrs.tpl.certain, tnrs.trop.small.certain)
dim(tnrs.tpl.all.trop.certain)
```

… and add the four additional columns:

```r
names(tnrs.tpl.all.trop.certain)

tnrs.tpl.all.trop.certain$Manual.matching <- NA
tnrs.tpl.all.trop.certain$Status.correct <- NA
tnrs.tpl.all.trop.certain$name.correct <- NA
tnrs.tpl.all.trop.certain$rank.correct <- NA
```

6.1.3 Pick the respective `NCBI` data sets

… for the 8,177 certain species:

```r
names(tnrs.ncbi.certain.comb)
tnrs.ncbi.certain.comb$rank.correct <- NA
```

Combine the with the big list above:

```r
tnrs.tpl.all.trop.certain.2 <- rbind(tnrs.tpl.all.trop.certain, tnrs.ncbi.certain.comb)
dim(tnrs.tpl.all.trop.certain.2)
names(tnrs.tpl.all.trop.certain.2)
```

Pick the list containing the `uncertain` species:

```r
names(ncbi.uncertain.comb)
```

Exclude columns `JD` and `corrected`

```r
ncbi.uncertain.comb.2 <- ncbi.uncertain.comb[,-c(5,6)]
names(ncbi.uncertain.comb.2)
ncbi.uncertain.comb.2$rank.correct <- NA
```

Combine them big list containing the `certain` species with the `uncertain` species

```r
tnrs.tpl.all.trop.tnrs.certain <- rbind(tnrs.tpl.all.trop.certain.2, ncbi.uncertain.comb.2)
write.csv(tnrs.tpl.all.trop.tnrs.certain, file = "tnrs.tpl.all.trop.tnrs.certain.csv")
save(tnrs.tpl.all.trop.tnrs.certain, file = "tnrs.tpl.all.trop.tnrs.certain.Rdata")
```

Correct one more species name:

```r
tnrs.tpl.all.trop.tnrs.certain$name.correct[which(tnrs.tpl.all.trop.tnrs.certain$name.correct
                                                   == "ABIES NORDMANNIANA")] <-
    "Abies nordmanniana"
```

6.2 Complement the main columns in the backbone

```r
dim(tnrs.tpl.all.trop.tnrs.certain)
names(tnrs.tpl.all.trop.tnrs.certain)
```

6.2.1 Fill in `rank.correct`

```r
tnrs.tpl.all.trop.tnrs.certain.filled <- tnrs.tpl.all.trop.tnrs.certain

tnrs.tpl.all.trop.tnrs.certain.filled$rank.correct <- as.character(
    tnrs.tpl.all.trop.tnrs.certain.filled$rank.correct)

tnrs.tpl.all.trop.tnrs.certain.filled$rank.correct[
                                          is.na(tnrs.tpl.all.trop.tnrs.certain.filled$
                                                Manual.matching)] <-
    tnrs.tpl.all.trop.tnrs.certain.filled$Name_matched_rank[
                                              is.na(tnrs.tpl.all.trop.tnrs.certain.filled$
                                                    Manual.matching)]
```

6.2.2 Fill `status.correct`

```r
tnrs.tpl.all.trop.tnrs.certain.filled$Status.correct[
                                          is.na(tnrs.tpl.all.trop.tnrs.certain.filled$
                                                Manual.matching)] <-
    tnrs.tpl.all.trop.tnrs.certain.filled$Taxonomic_status[
                                              is.na(tnrs.tpl.all.trop.tnrs.certain.filled$
                                                    Manual.matching)]
```

6.2.3 Fill `name.correct`

```r
tnrs.tpl.all.trop.tnrs.certain.filled$name.correct[is.na(
                                          tnrs.tpl.all.trop.tnrs.certain.filled$
                                          Manual.matching)] <-
```

```r
    tnrs.tpl.all.trop.tnrs.certain.filled$Accepted_name[is.na(
                                                     tnrs.tpl.all.trop.tnrs.certain.filled$
                                                     Manual.matching)]
```

Fill `name.correct` where `status.correct` == `No opinion`:

```r
tnrs.tpl.all.trop.tnrs.certain.filled$name.correct[which(
                                            tnrs.tpl.all.trop.tnrs.certain.filled$
                                            Status.correct == "No opinion")] <-
    tnrs.tpl.all.trop.tnrs.certain.filled$Name_matched[which(
                                                tnrs.tpl.all.trop.tnrs.certain.filled$
                                                Status.correct == "No opinion")]

write.csv(tnrs.tpl.all.trop.tnrs.certain.filled,
          file = "tnrs.tpl.all.trop.tnrs.certain.filled.csv")
```

In `tnrs.tpl.all.trop.tnrs.certain.filled` resolve some species manually on the TPL webpage.

6.2.4 Fill in `names.short correct`

```r
tnrs.tpl.all.trop.tnrs.certain.filled <- read.csv("tnrs.tpl.all.trop.tnrs.certain.filled.csv",
                                                  stringsAsFactors=FALSE)
names(tnrs.tpl.all.trop.tnrs.certain.filled)
str(tnrs.tpl.all.trop.tnrs.certain.filled)
tnrs.tpl.all.trop.tnrs.certain.filled$name.short.correct <- as.character(
    tnrs.tpl.all.trop.tnrs.certain.filled$name.short.correct)
```

6.2.5 Shorten names that have more than two words and where the second word is a `x`

Because some names consisted of $>$ 2 words, the number of words in each name was counted:

```r
wordcount <- sapply(gregexpr("\\S+", tnrs.tpl.all.trop.tnrs.certain.filled$name.correct),
                    length)
```

If a name has $>$ 1 word, just keep the first two words:

```r
tnrs.tpl.all.trop.tnrs.certain.filled$name.short.correct[wordcount>1] <-
    word(tnrs.tpl.all.trop.tnrs.certain.filled$name.correct[wordcount>1], 1, 2)
```

Fill in one-word names `name.short.correct`:

```r
tnrs.tpl.all.trop.tnrs.certain.filled$name.short.correct[wordcount==1] <-
    tnrs.tpl.all.trop.tnrs.certain.filled$name.correct[wordcount==1]
```

Select names where the second word is a `x` (hybrids):

```r
length(word(tnrs.tpl.all.trop.tnrs.certain.filled$name.short.correct[wordcount>1], 2)=="x")

write.csv(tnrs.tpl.all.trop.tnrs.certain.filled,
          file = "tnrs.tpl.all.trop.tnrs.certain.filled.csv")
```

There are 7,900 species with more than 2 words and where 2nd word is “x”. For those names in `name.short.correct` (i) where the second word is a `x` and that (ii) had more than two words, pick the first and third word:

```
index <- word(tnrs.tpl.all.trop.tnrs.certain.filled$name.correct[wordcount>2], 2) == "x"
wo.x <- paste(word(tnrs.tpl.all.trop.tnrs.certain.filled$name.correct[wordcount>2][index],
1), word(tnrs.tpl.all.trop.tnrs.certain.filled$name.correct[wordcount>2][index], 3))

tnrs.tpl.all.trop.tnrs.certain.filled$name.short.correct[wordcount>2][index] <- wo.x
```

6.2.6 In `name.short correct` insert `NA` where `"No suitable matches found."`

```
tnrs.tpl.all.trop.tnrs.certain.filled <- read.csv("tnrs.tpl.all.trop.tnrs.certain.filled.csv",
                                                  stringsAsFactors=FALSE)
tnrs.tpl.all.trop.tnrs.certain.filled$name.short.correct[tnrs.tpl.all.trop.tnrs.certain.filled$
                                                         name.correct ==
                                                         "No suitable matches found."] <- NA
```

6.2.7 Fill in `rank.short.correct`

```
tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct <-
    tnrs.tpl.all.trop.tnrs.certain.filled$rank.correct
table(tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct)
```

Assign `rank` = `species` to the shortened names of `subspecies`, `subvariety`, `variety` and `forma`.

```
tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct[tnrs.tpl.all.trop.tnrs.certain.filled$
                                                         rank.correct == "infraspecies"] <-
    "species"

tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct[
                                          tnrs.tpl.all.trop.tnrs.certain.filled$
                                          rank.correct == "subspecies"] <- "species"

tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct[
                                          tnrs.tpl.all.trop.tnrs.certain.filled$
                                          rank.correct == "subvariety"] <- "species"

tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct[
                                          tnrs.tpl.all.trop.tnrs.certain.filled$
                                          rank.correct == "variety"] <- "species"

tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct[
                                          tnrs.tpl.all.trop.tnrs.certain.filled$
                                          rank.correct == "forma"] <- "species"
```

Assign `rank.short.correct` = `family` to `rank.correct` == `unknown` since all of those names were at the family level.

```
tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct[
                                          tnrs.tpl.all.trop.tnrs.certain.filled$
                                          rank.correct == "unknown"] <- "family"

write.csv(tnrs.tpl.all.trop.tnrs.certain.filled,
          file = "tnrs.tpl.all.trop.tnrs.certain.filled.csv")
```

6.3 Fix families

```
tnrs.tpl.all.trop.tnrs.certain.filled <- read.csv("../tnrs.tpl.all.trop.tnrs.certain.filled.csv",
                                                  stringsAsFactors=FALSE)
```

```
names(tnrs.tpl.all.trop.tnrs.certain.filled)
```

```
##  [1] "X.3"                              "X.2"
##  [3] "X.1"                              "X"
##  [5] "Name_number"                      "Name_submitted"
##  [7] "Overall_score"                    "Name_matched"
##  [9] "Name_matched_rank"                "Name_score"
## [11] "Name_matched_author"              "Name_matched_url"
## [13] "Author_matched"                   "Author_score"
## [15] "Family_matched"                   "Family_score"
## [17] "Name_matched_accepted_family"     "Genus_matched"
## [19] "Genus_score"                      "Specific_epithet_matched"
## [21] "Specific_epithet_score"           "Infraspecific_rank"
## [23] "Infraspecific_epithet_matched"    "Infraspecific_epithet_score"
## [25] "Infraspecific_rank_2"             "Infraspecific_epithet_2_matched"
## [27] "Infraspecific_epithet_2_score"    "Annotations"
## [29] "Unmatched_terms"                  "Taxonomic_status"
## [31] "Accepted_name"                    "Accepted_name_author"
## [33] "Accepted_name_rank"               "Accepted_name_url"
## [35] "Accepted_name_species"            "Accepted_name_family"
## [37] "Selected"                         "Source"
## [39] "Warnings"                         "Accepted_name_lsid"
## [41] "user_id"                          "Manual.matching"
## [43] "Status.correct"                   "name.correct"
## [45] "rank.correct"                     "family.correct"
## [47] "name.short.correct"               "rank.short.correct"
```

Fill in family names in `family.correct` for entries where the matched rank was `family` and that were not matched manually:

```
tnrs.tpl.all.trop.tnrs.certain.filled$family.correct <- as.character(
    tnrs.tpl.all.trop.tnrs.certain.filled$family.correct)

tnrs.tpl.all.trop.tnrs.certain.filled$family.correct[(
    is.na(tnrs.tpl.all.trop.tnrs.certain.filled$Manual.matching) &
    tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct == "family")] <-
    tnrs.tpl.all.trop.tnrs.certain.filled$name.correct[(
        is.na(tnrs.tpl.all.trop.tnrs.certain.filled$Manual.matching) &
        tnrs.tpl.all.trop.tnrs.certain.filled$rank.short.correct == "family")]

tnrs.tpl.all.trop.tnrs.certain.filled$family.correct[(
    is.na(tnrs.tpl.all.trop.tnrs.certain.filled$Manual.matching))] <-
    tnrs.tpl.all.trop.tnrs.certain.filled$Name_matched_accepted_family[(
    is.na(tnrs.tpl.all.trop.tnrs.certain.filled$Manual.matching))]

write.csv(tnrs.tpl.all.trop.tnrs.certain.filled,
          file = "tnrs.tpl.all.trop.tnrs.certain.filled.csv")
```

6.4 Link backbone to original taxa names and sPlot/TRY code

```
splot.try3.code <- read.csv("spec.list.TRY3.sPlot2.csv", stringsAsFactors=FALSE)
str(splot.try3.code)

backbone.splot.try3 <- read.csv("../tnrs.tpl.all.trop.tnrs.certain.filled.small.csv",
                                stringsAsFactors=FALSE)
```

```
str(backbone.splot.try3)
```

```
## 'data.frame':    122901 obs. of  30 variables:
##  $ Name_number                 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Name_submitted              : chr  "Spermatophyta sp." "Spermatophyta sp." "Chlorophytum sp. [126
##  $ Overall_score               : num  0 0 0.9 0.9 0.9 0.9 0.9 0 0.9 0 ...
##  $ Name_matched                : chr  "No suitable matches found." "No suitable matches found." "Chl
##  $ Name_matched_rank           : chr  "" "" "genus" "genus" ...
##  $ Name_score                  : num  0 0 1 1 1 1 1 0 1 0 ...
##  $ Family_score                : num  0 0 NA NA NA NA 1 0 1 0 ...
##  $ Name_matched_accepted_family: chr  "" "" "Asparagaceae" "Poaceae" ...
##  $ Genus_matched               : chr  "" "" "Chlorophytum" "Echinochloa" ...
##  $ Genus_score                 : num  0 0 1 1 1 1 NA 0 NA 0 ...
##  $ Specific_epithet_matched    : chr  "" "" "" "" ...
##  $ Specific_epithet_score      : num  0 0 NA NA NA NA NA 0 NA 0 ...
##  $ Unmatched_terms             : chr  "" "" "\"\"sp. [1269]" "\"\"sp." ...
##  $ Taxonomic_status            : chr  "" "" "Accepted" "Accepted" ...
##  $ Accepted_name               : chr  "" "" "Chlorophytum" "Echinochloa" ...
##  $ Accepted_name_author        : chr  "" "" "" "" ...
##  $ Accepted_name_rank          : chr  "" "" "genus" "genus" ...
##  $ Accepted_name_url           : chr  "" "" "http://www.theplantlist.org/tpl1.1/search?q=Chlorophytu
##  $ Accepted_name_species       : chr  "" "" "" "" ...
##  $ Accepted_name_family        : chr  "" "" "Asparagaceae" "Poaceae" ...
##  $ Selected                    : chr  "true" "true" "true" "true" ...
##  $ Source                      : chr  "" "" "tpl" "tpl" ...
##  $ Warnings                    : chr  " " " " " " " " ...
##  $ Manual.matching             : chr  NA NA NA NA ...
##  $ Status.correct              : chr  "No suitable matches found." "No suitable matches found." "Acc
##  $ name.correct                : chr  "No suitable matches found." "No suitable matches found." "Chl
##  $ rank.correct                : chr  "higher" "higher" "genus" "genus" ...
##  $ family.correct              : chr  "" "" "Asparagaceae" "Poaceae" ...
##  $ name.short.correct          : chr  NA NA "Chlorophytum" "Echinochloa" ...
##  $ rank.short.correct          : chr  "higher" "higher" "genus" "genus" ...
```

```
backbone.splot.try3 <- join(splot.try3.code, backbone.splot.try3, by = "Name_number")
str(backbone.splot.try3)
```

```
table(backbone.splot.try3$Status.correct)
```

```
##
##                   Accepted                 No opinion
##                      95485                       4591
## No suitable matches found.                    Synonym
##                       1692                      20952
##                 Unresolved
##                        181
```

For constistency, assign `Unresolved` to `status = Unresolved`:

```
backbone.splot.try3$Status.correct[backbone.splot.try3$Status.correct == "No opinion"] <-
    "Unresolved"

write.csv(backbone.splot.try3, file = "backbone.splot.try3.csv")
save(backbone.splot.try3, file = "backbone.splot.try3.Rdata")
```

6.5 Some statistics

```
table(backbone.splot.try3$sPlot.TRY)
```

```
## < table of extent 0 >
```

```
table(backbone.splot.try3$Manual.matching)
```

```
##
##    x
## 1675
```

```
table(backbone.splot.try3$Status.correct)
```

```
##
##                   Accepted                 No opinion
##                      95485                       4591
## No suitable matches found.                    Synonym
##                       1692                      20952
##                 Unresolved
##                        181
```

```
length(unique(backbone.splot.try3$name.correct))-1
```

```
## [1] 90696
```

```
length(unique(backbone.splot.try3$family.correct))-1
```

```
## [1] 665
```

```
length(unique(backbone.splot.try3$name.short.correct))-1
```

```
## [1] 86528
```

```
table(backbone.splot.try3$rank.short.correct)
```

```
##
##  family   genus  higher species
##    1880   13383    1211  105818
```

7 Resolve the 7,701 additional species from the ad-hoc version sPlot2.1

Get the species in the sPlot-July2015-version (`sPlot_2015_07_29_species`) that do not match with the backbone in the April2015_Version (sPlot 2.0, `sPlot_14_4_2015_species`) that was use in matching procedure above. Apply the cleaning procedure, similar to the one used for `sPlot 2.0`.

```
miss.new <- read.csv("/home/oliver/Dokumente/PhD/PostPhD/IDiv/sDiv/sPlot/Analyses/Code/
Mismatches_29_07_2015_new.csv")
```

```
dim(miss.new)
```

```
## [1] 7701    1
```

There are 7,701 names in the `sPlot version 29_07_2015` that were not in splot 2.0 (April 2015).

7.1 Name cleaning (spelling of ranks, name additions etc.)

```
OriginalNames <- as.character(miss.new$x)

OriginalNames <- gsub('*', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('cf. ', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('Cf. ', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('[', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(']', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(' x ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('Ã-', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('aff ', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('(', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(')', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(' cf ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(' aff. ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('c,e', 'ceae', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('    ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('   ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('  ', ' ', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('x-', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('X-', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub('like ', '', OriginalNames, fixed=TRUE)
OriginalNames <- gsub(',', '', OriginalNames, fixed=TRUE)

library(stringr)
firstWordWithNumbers <- grepl('[0-9]', word(OriginalNames, 1))
numberOfWords <- sapply(gregexpr("\\W+", OriginalNames), length) + 1
OriginalNames[firstWordWithNumbers & numberOfWords > 1] <- sapply(OriginalNames[
    firstWordWithNumbers & numberOfWords > 1], function(x) substr(x, start=regexpr(
                                                                         pattern =' ',
                                                                         text=x)+1,
                                                                  stop=nchar(x)))

CleanedNames <- taxname.abbr(OriginalNames)
write.csv(CleanedNames, file = "CleanedNames.csv")
save(CleanedNames, file = "CleanedNames.Rdata")
```

7.2 Upload cleaned names to TRNS

On the TNRS application webpage, use all six sources (`TPL` first, see procedure above). Subsequently, use `Tropicos` first.

```r
tnrs.tpl.new <- read.csv("/home/oliver/Downloads/missing_tpl.txt", sep = "\t",
                          stringsAsFactors = FALSE, skip = 0, head = T)[-1, ]
str(tnrs.tpl.new)
tnrs.tpl <- read.csv("tnrs.tpl.csv")
```

7.3 Select correctly resolved names (TPL first)

Assign index to first column:

```r
tnrs.tpl$Name_number <- 1:length(tnrs.tpl$Name_number)

tnrs.tpl <- tnrs.tpl[,-1]
```

7.3.1 Family level

```r
index.family <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "family" &
                               (tnrs.tpl$Taxonomic_status == "Accepted" |
                                tnrs.tpl$Taxonomic_status == "Synonym") &
                               tnrs.tpl$Family_score > 0.88), 1]
length(index.family)
```

Repeat selection procedure for `Name_matched_rank` ==:

- `forma`
- `genus`
- `infraspecies`
- `species`
- etc.

Create respective index:

7.3.2 Forma level

```r
index.forma <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "forma"), 1]
length(index.forma)

### Genus level
```

```r
index.genus <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "genus" &
                              tnrs.tpl$Taxonomic_status == "Accepted" &
                              tnrs.tpl$Genus_score > 0.83), 1]
length(index.genus)
```

7.3.3 Infraspecific level

```r
index.infraspec <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "infraspecies"), 1]
length(index.infraspec)
```

7.3.4 Species level

```r
index.species <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "species" &
                                (tnrs.tpl$Taxonomic_status == "Accepted" |
                                 tnrs.tpl$Taxonomic_status == "Synonym") &
                                tnrs.tpl$Genus_score > 0.78 & tnrs.tpl$Name_score > 0.93),1]
length(index.species1)

index.species <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "species" &
                                (tnrs.tpl$Taxonomic_status == "Accepted" |
                                 tnrs.tpl$Taxonomic_status == "Synonym") &
                                tnrs.tpl$Specific_epithet_score > 0.78), 1]
length(index.species2)

index.species <- unique(c(index.species1, index.species2))
length(index.species)
```

7.3.5 Subspecies level

```r
index.subspec <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "subspecies" &
                                (tnrs.tpl$Taxonomic_status == "Accepted" |
                                 tnrs.tpl$Taxonomic_status == "Synonym")), 1]
length(index.subspec)
```

7.3.6 Variety level

```r
index.variety <- tnrs.tpl[which(tnrs.tpl$Name_matched_rank == "variety" &
                                (tnrs.tpl$Taxonomic_status == "Accepted" |
                                 tnrs.tpl$Taxonomic_status == "Synonym")), 1]
length(index.variety)
```

7.3.7 Identifying non-matched species that are spermatophyta

```r
index.spermatophyt <- tnrs.tpl[which(tnrs.tpl$Name_matched == "No suitable matches found." &
                                     word(tnrs.tpl$Name_submitted, 1) == "Spermatophyta"), 1]
length(index.spermatophyt)
```

7.3.8 Identify species that do not fulfill the search criteria

```r
index.tpl <- c(index.family, index.forma, index.genus, index.species, index.subspec,
               index.variety, index.spermatophyt)
length((index.tpl))

tnrs.tpl.certain <- tnrs.tpl[index.tpl,]
dim(tnrs.tpl.certain)
save(tnrs.tpl.certain, file = "tnrs.tpl.certain.Rdata")
write.csv(tnrs.tpl.certain, file = "tnrs.tpl.certain.csv")
```

```r
tnrs.tpl.uncertain <- tnrs.tpl[tnrs.tpl$Name_number %in% index.tpl == F, ]
dim(tnrs.tpl.uncertain)
save(tnrs.tpl.uncertain, file = "tnrs.tpl.uncertain.Rdata")
write.csv(tnrs.tpl.uncertain, file = "tnrs.tpl.uncertain.csv")

write.csv(tnrs.tpl.uncertain[,2], file = "tnrs.tpl.uncertain.upload.csv")
```

7.4 Select correctly resolved names (Tropicos first)

```r
tnrs.trop <- read.csv("/home/oliver/Downloads/tnrs.trop.txt", sep = "\t")
str(tnrs.trop)
```

```r
str(tnrs.trop)
tnrs.trop$Name_number <- 1:length(tnrs.trop$Name_number)
```

7.4.1 Reduce to the uncertain species from tnrs.tpl

```r
tnrs.trop.small <- tnrs.trop[tnrs.trop$Name_number %in% index.tpl == F, ]
str(tnrs.trop.small)

save(tnrs.trop.small, file = "tnrs.trop.small.Rdata")
write.csv(tnrs.trop.small, file = "tnrs.trop.small.csv")
```

7.4.2 Family level

```r
index.family <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "family" &
                                      (tnrs.trop.small$Taxonomic_status == "Accepted"|
                                       tnrs.trop.small$Taxonomic_status == "No opinion"))
                                , 1]
length(index.family)
```

7.4.3 Forma level

```r
index.forma <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "forma" &
                                     (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                      tnrs.trop.small$Taxonomic_status == "Synonym")),
                               1]
length(index.forma)
```

7.4.4 Genus level

```r
index.genus <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "genus" &
                                     (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                      tnrs.trop.small$Taxonomic_status == "Synonym" |
                                      tnrs.trop.small$Taxonomic_status == "No opinion") &
```

```
                                    tnrs.trop.small$Genus_score > 0.83), 1]
length(index.genus)
```

7.4.5 Species level

```
index.species <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "species" &
                                      (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                       tnrs.trop.small$Taxonomic_status == "Synonym" |
                                       tnrs.trop.small$Taxonomic_status == "No opinion") &
                                      tnrs.trop.small$Specific_epithet_score > 0.77), 1]
length(index.species)

index.species2 <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "species" &
                                       (tnrs.trop.small$Taxonomic_status == "Accepted" |
                                        tnrs.trop.small$Taxonomic_status == "Synonym") &
                                       tnrs.trop.small$Genus_score > 0.88 &
                                       tnrs.trop.small$Name_score > 0.49), 1]
length(index.species2)

index.species <- unique(c(index.species1, index.species2))
length(index.species)
```

7.4.6 Subspecies level

```
index.subspec <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "subspecies"),
                                 1]
length(index.subspec)
```

7.4.7 Subvariety level

```
index.subvariety <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "subvariety"),
                                    1]
length(index.subvariety)
```

7.4.8 Variety level

```
index.variety <- tnrs.trop.small[which(tnrs.trop.small$Name_matched_rank == "variety" &
                                      tnrs.trop.small$Taxonomic_status == "No opinion"),
                                 1]
length(index.variety)

index.all <- c(index.family, index.genus, index.species, index.subspec, index.variety,
               index.subvariety)
length((index.all))

tnrs.trop.small.certain <- tnrs.trop.small[tnrs.trop.small$Name_number %in%
                                           index.all == T,]
dim(tnrs.trop.small.certain)
```

```r
save(tnrs.trop.small.certain, file = "tnrs.trop.small.certain.Rdata")
write.csv(tnrs.trop.small.certain, file = "tnrs.trop.small.certain.csv")

tnrs.trop.small.uncertain <- tnrs.trop.small[tnrs.trop.small$Name_number %in%
                                             index.all == F, ]
dim(tnrs.trop.small.uncertain)
save(tnrs.trop.small.uncertain, file = "tnrs.trop.small.uncertain.Rdata")
write.csv(tnrs.trop.small.uncertain, file = "tnrs.trop.small.uncertain.csv")
```

```r
backbone.tpl.trop.certain <- rbind(tnrs.tpl.certain, tnrs.trop.small.certain)
str(backbone.tpl.trop.certain)
```

7.4.9 Kick out some columns that should not be in the final backbone

```r
backbone.tpl.trop.certain.2 <- backbone.tpl.trop.certain[ ,c(1:6,12:17,25:35)]
colnames(backbone.tpl.trop.certain.2)
```

7.4.10 Add extra columns to match the old backbone

```r
backbone.tpl.trop.certain.2$Manual.matching <- NA
backbone.tpl.trop.certain.2$Status.correct <- NA
backbone.tpl.trop.certain.2$name.correct <- NA
backbone.tpl.trop.certain.2$rank.correct <- NA
backbone.tpl.trop.certain.2$family.correct <- NA
backbone.tpl.trop.certain.2$name.short.correct <- NA
backbone.tpl.trop.certain.2$rank.short.correct <- NA
backbone.tpl.trop.certain.2$names.sPlot.TRY <- NA
backbone.tpl.trop.certain.2$names.corr.string <- NA
backbone.tpl.trop.certain.2$sPlot.TRY <- NA
```

7.4.11 Bring names in the same order as in the old backbone

```r
backbone.tpl.trop.certain.3 <- backbone.tpl.trop.certain.2[,c(1,31:33,2:30)]
match(colnames(backbone.tpl.trop.certain.3), colnames(backbone.splot.try3))
identical(colnames(backbone.tpl.trop.certain.3), colnames(backbone.splot.try3)) # TRUE
```

7.5 Fill in empty columns

7.5.1 `Status.correct`

```r
backbone.tpl.trop.certain.3$Status.correct <- backbone.tpl.trop.certain.3$Taxonomic_status
index <- which(backbone.tpl.trop.certain.3$Name_matched == "No suitable matches found.")
backbone.tpl.trop.certain.3$Status.correct[index] <- "No suitable matches found."
```

Rename `no.opinion` to `unresolved`

```r
index <- which(backbone.tpl.trop.certain.3$Status.correct == "No opinion")
backbone.tpl.trop.certain.3$Status.correct[index] <- "Unresolved"
```

7.5.2 Name.correct

Use “Accepted name”, if unresolved take “Name_matched”)

```r
backbone.tpl.trop.certain.3$name.correct <- backbone.tpl.trop.certain.3$Accepted_name
index <- which(backbone.tpl.trop.certain.3$Taxonomic_status != "Accepted" |
               backbone.tpl.trop.certain.3$Taxonomic_status != "Synonym")
backbone.tpl.trop.certain.3$name.correct[index] <- backbone.tpl.trop.certain.3$
    Name_matched[index]
index <- which(backbone.tpl.trop.certain.3$Name_matched == "No suitable matches found.")
backbone.tpl.trop.certain.3$name.correct[index] <- "No suitable matches found."
```

7.5.3 Rank.correct

```r
backbone.tpl.trop.certain.3$rank.correct <- backbone.tpl.trop.certain.3$Accepted_name_rank
index <- which(backbone.tpl.trop.certain.3$Taxonomic_status != "Accepted" |
               backbone.tpl.trop.certain.3$Taxonomic_status != "Synonym")
backbone.tpl.trop.certain.3$rank.correct[index] <-
    backbone.tpl.trop.certain.3$Name_matched_rank[index]
index <- which(backbone.tpl.trop.certain.3$Name_matched == "No suitable matches found.")
backbone.tpl.trop.certain.3$rank.correct[index] <- NA
```

7.5.4 Family.correct

```r
backbone.tpl.trop.certain.3$family.correct <- backbone.tpl.trop.certain.3$
    Accepted_name_family
index <- which(backbone.tpl.trop.certain.3$Taxonomic_status != "Accepted" |
               backbone.tpl.trop.certain.3$Taxonomic_status != "Synonym")
backbone.tpl.trop.certain.3$family.correct[index] <-
    backbone.tpl.trop.certain.3$Name_matched_accepted_family[index]
index <- which(backbone.tpl.trop.certain.3$Name_matched == "No suitable matches found.")
backbone.tpl.trop.certain.3$family.correct[index] <- NA
```

7.5.5 Name.short.correct

```r
backbone.tpl.trop.certain.3$name.short.correct <- backbone.tpl.trop.certain.3$name.correct

index <- which(backbone.tpl.trop.certain.3$rank.correct == "subspecies" |
               backbone.tpl.trop.certain.3$rank.correct == "variety" |
               backbone.tpl.trop.certain.3$rank.correct == "forma"|
               backbone.tpl.trop.certain.3$rank.correct == "subvariety" |
               backbone.tpl.trop.certain.3$Name_matched_rank == "subspecies" |
               backbone.tpl.trop.certain.3$Name_matched_rank == "variety")

library(stringr)
backbone.tpl.trop.certain.3$name.short.correct[index] <-
    word(string = backbone.tpl.trop.certain.3$name.short.correct[index], start = 1, end = 2)
```

7.5.6 `Rank.short.correct`

```
backbone.tpl.trop.certain.3$rank.short.correct <- backbone.tpl.trop.certain.3$rank.correct

index <- which(backbone.tpl.trop.certain.3$rank.correct == "subspecies" |
               backbone.tpl.trop.certain.3$rank.correct == "variety" |
               backbone.tpl.trop.certain.3$rank.correct == "forma"|
               backbone.tpl.trop.certain.3$rank.correct == "subvariety")

backbone.tpl.trop.certain.3$rank.short.correct[index] <- "species"
```

7.5.7 Kick out some columns that should not be in the final backbone

```
tnrs.trop.small.uncertain.2 <- tnrs.trop.small.uncertain[ ,c(1:6,12:17,25:35)]
colnames(tnrs.trop.small.uncertain.2)
```

Add extra columns to match the old backbone

```
tnrs.trop.small.uncertain.2$Manual.matching <- NA
tnrs.trop.small.uncertain.2$Status.correct <- NA
tnrs.trop.small.uncertain.2$name.correct <- NA
tnrs.trop.small.uncertain.2$rank.correct <- NA
tnrs.trop.small.uncertain.2$family.correct <- NA
tnrs.trop.small.uncertain.2$name.short.correct <- NA
tnrs.trop.small.uncertain.2$rank.short.correct <- NA

tnrs.trop.small.uncertain.2$names.sPlot.TRY <- NA
tnrs.trop.small.uncertain.2$names.corr.string <- NA
tnrs.trop.small.uncertain.2$sPlot.TRY <- NA
```

7.5.7.1 Bring names in the same order as in the old backbone

```
tnrs.trop.small.uncertain.3 <- tnrs.trop.small.uncertain.2[,c(1,31:33,2:30)]
match(colnames(tnrs.trop.small.uncertain.3), colnames(backbone.splot.try3))
identical(colnames(tnrs.trop.small.uncertain.3), colnames(backbone.splot.try3)) # TRUE

tnrs.trop.small.uncertain.3[,c(28:29)] <- "No suitable matches found."
```

7.5.7.2 Merge with backbone for certain species

```
backbone.tpl.trop <- rbind(backbone.tpl.trop.certain.3, tnrs.trop.small.uncertain.3)
head(backbone.tpl.trop)
dim(backbone.tpl.trop)
write.csv(backbone.tpl.trop, file = "backbone.tpl.trop.csv")
backbone.tpl.trop <- read.csv("backbone.tpl.trop.csv")
```

7.5.8 Fill in remaining first three columns

Assing all entries to sPlot ("S")

```
backbone.tpl.trop$sPlot.TRY <- "S"
```

Assing all entries to sPlot-version 2.1 (“sPlot2b”)

```
backbone.tpl.trop$Manual.matching <- "sPlot2b"
miss.clean.2 <- cbind(1:length(miss.clean[,1]), as.data.frame(miss.clean))
colnames(miss.clean.2)[1] <- "Name_number"
```

7.5.9 Join the TRNS results

```
backbone.tpl.trop <- left_join(miss.clean.2, backbone.tpl.trop, by = "Name_number")
head(backbone.tpl.trop)

colnames(backbone.tpl.trop)

backbone.tpl.trop.2 <- backbone.tpl.trop[,-c(4:6)]
colnames(backbone.tpl.trop.2)[2:3] <- c("names.sPlot.TRY", "names.corr.string")
match(colnames(backbone.tpl.trop.2), colnames(backbone.splot.try3))
identical(colnames(backbone.tpl.trop.2), colnames(backbone.splot.try3))

write.csv(backbone.tpl.trop.2, file = "backbone.tpl.trop.2.csv")
save(backbone.tpl.trop.2, file = "backbone.tpl.trop.2.Rdata")
```

7.5.10 Merge additional species with the big backbone for `sPlot 2.0` and `TRY 3.0`

```
backbone.splot2b.try3 <- rbind(backbone.splot.try3, backbone.tpl.trop.2)
dim(backbone.splot2b.try3)
```

Check whether the new backbone matches the `sPlot 2.1` species data:

```
str(splot.species)
length(unique(splot.species$Matched_concept))
which(unique(splot.species$Matched_concept) %in% backbone.splot2b.try3$names.sPlot.TRY == F)
backbone.splot2.1.try3 <- backbone.splot2b.try3
write.csv(backbone.splot2.1.try3, file = "backbone.splot2.1.try3.csv")
save(backbone.splot2.1.try3, file = "backbone.splot2.1.try3.Rdata")
```

8 Tag vascular species in `backbone.splot2.1.try3`

Because sPlot, and to some extent TRY, contain a bunch of names belonging to non-vascular, those need to be tagged.

```
load("../backbone.splot2.1.try3.is.vascular.Rdata")
```

```
colnames(backbone.splot2.1.try3)
```

```
##  [1] "Name_number"                "names.sPlot.TRY"
##  [3] "names.corr.string"          "sPlot.TRY"
##  [5] "Name_submitted"             "Overall_score"
##  [7] "Name_matched"               "Name_matched_rank"
##  [9] "Name_score"                 "Family_score"
## [11] "Name_matched_accepted_family" "Genus_matched"
## [13] "Genus_score"                  "Specific_epithet_matched"
## [15] "Specific_epithet_score"       "Unmatched_terms"
## [17] "Taxonomic_status"             "Accepted_name"
## [19] "Accepted_name_author"         "Accepted_name_rank"
## [21] "Accepted_name_url"            "Accepted_name_species"
## [23] "Accepted_name_family"         "Selected"
## [25] "Source"                       "Warnings"
## [27] "Manual.matching"              "Status.correct"
## [29] "name.correct"                 "rank.correct"
## [31] "family.correct"               "name.short.correct"
## [33] "rank.short.correct"           "is.vascular.species"
## [35] "sPlot2.1.TRY"
```

8.1 Get families

```
fam <- unique(backbone.splot2.1.try3$family.correct)
head(fam)
```

```
## [1] ""              "Asparagaceae" "Poaceae"      "Fabaceae"
## [5] "Polygalaceae" "Orchidaceae"
```

```
gbiffam <- sapply(fam[2:5], function(x) name_usage(name=x,
                                                    rank = 'FAMILY,', limit = 1)$data$phylum)
gbiffam[which(sapply(gbiffam, function(x) is.null(x)))] <- 'unknown'
fam$phylum <- unlist(gbiffam, use.names = TRUE)
table(fam$phylum)
write.csv(fam, file='family_affiliation_gbif.csv')
family_affiliation_gbif <- read.csv('../Florian_TaxStand/family_affiliation_gbif.csv')
```

```
table(family_affiliation_gbif$phylum)
```

```
##
## Anthocerotophyta       Ascomycota    Basidiomycota        Bryophyta
##                1               35               15               57
##       Charophyta      Chlorophyta    Cyanobacteria      Glaucophyta
##                3                6                2                1
##  Marchantiophyta       Ochrophyta       Rhodophyta     Tracheophyta
##               45               17                8              474
##          unknown
##                5
```

8.1.1 Add column "is.vascular.species" and set all families that correspond to Tracheophyta (vascular plants == TRUE)

```
table(is.na(backbone.splot2.1.try3$is.vascular.species))
```

```
##
##  FALSE   TRUE
## 123589   7013
```

Select families that belong to `Tracheophyta`:

```r
fam.trach <- family_affiliation_gbif$Var1[family_affiliation_gbif$phylum == "Tracheophyta"]
dim(fam.trach)
ind.vasc <- backbone.splot2.1.try3$family.correct %in% fam.trach
backbone.splot2.1.try3$is.vascular.species <- ind.vasc
backbone.splot2.1.try3$is.vascular.species[backbone.splot2.1.try3$is.vascular.species ==
                                          FALSE] <- NA
backbone.splot2.1.try3.is.vascular <- backbone.splot2.1.try3
```

To obtain proper stats for `sPlot2.1`, kick out (or tag) the species that were only found in `sPlot 2.0` but not in `sPlot 2.1`:

```r
splot.species.july2015 <- read.csv("/home/oliver/Dokumente/PhD/PostPhD/IDiv/sDiv/sPlot/
Analyses/Data/Species/sPlot/sPlot_2015_07_29/sPlot_2015_07_29_species.csv", sep = "\t")
gc()

splot.species.july2015.spec <- as.character(unique(splot.species.july2015$Matched.concept))
str(splot.species.july2015.spec)
write.csv(splot.species.july2015.spec, file = "splot.species.july2015.spec.csv")
```

There are 86,432 unique, uncorrected names in `sPlot 2.1`. Check whether all of these names are in the new backbone?

```r
load("backbone.splot2.1.try3.is.vascular.Rdata")
```

Export small backbone:

```r
backbone.splot2.1.try3.small <- backbone.splot2.1.try3[,c(2,32)]
write.csv(backbone.splot2.1.try3.small, file = "backbone.splot2.1.try3.small.csv")

load("backbone.splot2.1.try3.Rdata")
ind <- (splot.species.july2015.spec %in% backbone.splot2.1.try3$names.sPlot.TRY)
splot.species.july2015.spec[ind==F]
table(ind)
```

But there are only 86,427 names here. Five species are missing! Why?

The following species are in `sPlot2.1` (`splot.species.july2015.spec`) but not in the backbone (`backbone.splot2.1.try`):

- "Strauch like Oraniquelocarpus"
- "Buchenavia [GUNDINGA]"
- "[ms554 Triumfetta Ahorn]"
- "Dolichos holosericea"
- "[Aeschynomene latifolia]"

Actually, they are in the backbone but just spelled differently, e.g. `Strauch like ""Oraniquelocarpus""` or `[ms554 Triumfetta "Ahorn"]`. Probably those names got messed up, generate a vector of that 5 species that is consistent with the spelling used in the `missing-species-list` and in the final backbone.

```r
miss7701 <- read.csv("/home/oliver/Dokumente/PhD/PostPhD/IDiv/sDiv/sPlot/Analyses/Code/
Mismatches_29_07_2015_new.csv", stringsAsFactors = F)
vec5 <- miss7701[c(49, 3165, 5793, 5814, 5823), 1]
```

Add these five species to the `sPlot 2.1` species list:

```r
splot.species.july2015.spec.5 <- c(splot.species.july2015.spec, vec5)
```

In the big backbone, identify species that are in "S" and "ST" and that are in our `sPlot 2.1` species list (should be 86,432).

```r
ind <- (splot.species.july2015.spec.5 %in% backbone.splot2.1.try3$names.sPlot.TRY)
splot.species.july2015 <- splot.species.july2015.spec.5[ind]
ind <- (splot.species.july2015 %in% backbone.splot2.1.try3$names.sPlot.TRY)
table(ind)
```

Great, these are the species in `sPlot 2.1`.

8.1.2 Select species in backbone that are in sPlot2.1 (`splot.species.july2015`)

```r
ind.splot2.1 <- which(backbone.splot2.1.try3$names.sPlot.TRY %in% splot.species.july2015)
```

There are 86,432 species in TRY, that are both only in TRY as well as in sPlot and TRY.

8.1.3 Select species that are in TRY

```r
ind.try <- which(backbone.splot2.1.try3$sPlot.TRY=="T" | backbone.splot2.1.try3$sPlot.TRY=="ST")
```

60,273 species that are in TRY.

8.1.4 Species that are both in sPlot and TRY

```r
intersect_all <- function(a,b,...){
  Reduce(intersect, list(a,b,...))
}

inter <- intersect_all(ind.splot2.1, ind.try)
```

24,844 names shared between sPlot and TRY.

Check whether the three numbers above are correct, and add new column `sPlot2.1.TRY` to backbone, that tags the entries that belong to `sPlot 2.1` and `TRY 3.0`, respectively.

```r
backbone.splot2.1.try3$ST <- NA

backbone.splot2.1.try3$ST[ind.splot2.1] <- "S"
backbone.splot2.1.try3$ST[ind.try] <- "T"
backbone.splot2.1.try3$ST[inter] <- "ST"
colnames(backbone.splot2.1.try3)[35] <- "sPlot2.1.TRY"
save(backbone.splot2.1.try3, file = "backbone.splot2.1.try3.is.vascular.Rdata")
```

9 Statistics for backbone combining names in `sPlot2.1` and `TRY3.0`

9.1 All taxon name entries

```r
load("../backbone.splot2.1.try3.is.vascular.Rdata")
```

How many entries in the backbone were only found in the old sPlot-version 2.0 (`sPlot_14_4_2015`) but not in `sPlot 2.1` `sPlot_2015_07_19`?

```
table(!is.na(backbone.splot2.1.try3$sPlot2.1.TRY))
```

```
##
##  FALSE   TRUE
##   8741 121861
```

There are 8,741 entries that do not belong to `sPlot 2.1`. In total 121,861 (unstandardized) name entries in `sPlot 2.1` and `TRY 3.0` combined. Index those species that were exclusiely found in the sPlot 2.0 but not in version 2.1, and remove them from the backbone (`back2.1`) to update the match statistics.

```
ind2.1 <- !is.na(backbone.splot2.1.try3$sPlot2.1.TRY)
back2.1 <- backbone.splot2.1.try3[ind2.1, ]
```

Database affiliations (`sPlot 2.1` and `TRY 3.0`).

```
kable(t(table(back2.1$sPlot2.1.TRY)), caption = "Number of (standardized) name entries
unique to, or shared between TRY (S) and sPlot (T).")
```

Table 1: Number of (standardized) name entries unique to, or shared between TRY (S) and sPlot (T).

S	ST	T
61588	24844	35429

60,273 of the total number of entries belong to TRY (incl. names that are only in TRY as well as in sPlot and TRY). 86,432 name entries belong to sPlot (incl. names that are only in TRY as well as in sPlot and TRY).

```
table(back2.1$Manual.matching)
```

```
##
## sPlot2b       x
##    7694    1432
```

1,432 name entries were matched manually (see above).

Taxonomic ranks:

```
kable(t(table(back2.1$rank.short.correct)), caption = "Number of (standardized) name entries
per taxonomic rank.")
```

Table 2: Number of (standardized) name entries per taxonomic rank.

family	genus	higher	species
1745	12373	1020	105777

Taxonomic status:

```
kable(t(table(back2.1$Status.correct)), caption = "Number of (standardized) name entries
that correspond to `Accepted`, `Synonyms` or Unresolved species, respecively.")
```

Table 3: Number of (standardized) name entries that correspond to `Accepted`, `Synonyms` or Unresolved species, respecively.

Accepted	No suitable matches found.	Synonym	Unresolved
94068	1902	21028	4863

Total number of unique standardized taxon names and families:

```r
length(unique(back2.1$name.short.correct))-1 # minus 1 for NA
```

```
## [1] 86760
```

```r
length(unique(back2.1$family.correct))-2 # minus 2 for "" and NA
```

```
## [1] 663
```

Number of entries corresponding to vascular plant species:

```r
table(back2.1$is.vascular.species)
```

```
## 
##   TRUE 
## 115678
```

Number of duplicated entries after taxonomic standardization: Frequency of original (non-standardized) species names per resolved (non-standardized) name (excluding non-vascular and non-matched species).

```r
df.count <- back2.1 %>%
    dplyr::filter(is.vascular.species == TRUE, !is.na(name.short.correct)) %>%
    dplyr::group_by(name.short.correct) %>%
    dplyr::summarise(n = n()) %>%
    dplyr::arrange(desc(n))

kable(df.count[c(1:3, 21:25, 101:110), ], , caption = "Number of unresolved, original name
enties per resolved name.")
```

9.2 Based on unique standardized names

Generate version of the backbone that only includes the unique resolved names in `name.short.correct`, and for the non-unique names, the first rows of duplicated name:

```r
back2.1.uni <- back2.1[!duplicated(back2.1$name.short.correct), ]
```

Remove the first entry, which is NA:

```r
back2.1.uni <- back2.1.uni[-1, ]
```

```r
length(unique(back2.1.uni$name.short.correct))
```

```
## [1] 86761
```

There are 86,760 unique taxon names the in backbone. Exclude the non-vascular plant and non-matching taxon names:

```r
df.uni <- back2.1.uni %>%
    dplyr::filter(is.vascular.species == TRUE, !is.na(name.short.correct))
```

`df.uni` had two names less than `df.count`, as they were accidentially tagged as non-vascular species names. Resolve that issue:

```
miss <- df.count$name.short.correct[(df.count$name.short.correct %in% df.uni$name.short.correct) == F]
which(df.count$name.short.correct %in% c("Arabis stellulata", "Coptidium pallasii"))
which(df.uni$name.short.correct %in% c("Arabis stellulata", "Coptidium pallasii"))
df.count[c(20319, 31478), ]
ind <- which(back2.1$name.short.correct %in% c("Arabis stellulata", "Coptidium pallasii"))
```

```
back2.1[ind, 31:34]
```

```
##       family.correct name.short.correct rank.short.correct
## 8853    Brassicaceae  Arabis stellulata            species
## 8932    Brassicaceae  Arabis stellulata            species
## 30191  Ranunculaceae Coptidium pallasii            species
## 96184  Ranunculaceae Coptidium pallasii            species
##       is.vascular.species
## 8853                 TRUE
## 8932                 TRUE
## 30191                TRUE
## 96184                TRUE
```

```
back2.1[ind[1], 31:34] <- back2.1[ind[2], 31:34]
back2.1[ind[3], 31:34] <- back2.1[ind[4], 31:34]

df.uni <- back2.1.uni %>%
    dplyr::filter(is.vascular.species == TRUE, !is.na(name.short.correct))
```

Now, run the stats for unique resolved names (excluding non-vascular and non-matching taxa):

```
length(df.uni$name.short.correct)
```

```
## [1] 83677
```

There are 83,679 unique (non-vascular plant) taxon names:

```
kable(t(table(df.uni$sPlot2.1.TRY)), caption = "Number of (standardized) vascular plant
taxon names per unique to, and shared between TRY (S) and sPlot (T).")
```

Table 4: Number of (standardized) vascular plant taxon names per unique to, and shared between TRY (S) and sPlot (T).

S	ST	T
34105	20414	29158

```
table(df.uni$Manual.matching)
```

```
##
## sPlot2b       x
##    1637     490
```

490 of those names were matched manually (see above).

Taxonomic ranks:

```
kable(t(table(df.uni$rank.short.correct)), caption = "Number of (standardized) name entries
per taxonomic rank.")
```

Table 5: Number of (standardized) name entries per taxonomic rank.

family	genus	higher	species
152	4343	13	79169

Taxonomic status:

```
kable(t(table(df.uni$Status.correct)), caption = "Number of (standardized) name entries that
correspond to `Accepted`, `Synonyms` or Unresolved species, respecively.")
```

Table 6: Number of (standardized) name entries that correspond to `Accepted`, `Synonyms` or Unresolved species, respecively.

Accepted	Synonym	Unresolved
69173	9998	4506

Total number of unique standardized taxon names and families:

```
length(unique(back2.1$name.short.correct))-1 # minus 1 for NA
```

```
## [1] 86760
```

```
length(unique(back2.1$family.correct))-2 # minus 2 for "" and NA
```

```
## [1] 663
```

Number of entries corresponding to vascular plant species:

```
table(back2.1$is.vascular.species)
```

```
##
##   TRUE
## 115678
```

Number of duplicated entries after taxonomic standardization: Frequency of original (non-standardized) species names per resolved (non-standardized) name (excluding non-vascular and non-matched species).

```
df.count <- back2.1 %>%
    dplyr::filter(is.vascular.species == TRUE, !is.na(name.short.correct)) %>%
    dplyr::group_by(name.short.correct) %>%
    dplyr::summarise(n = n()) %>%
    dplyr::arrange(desc(n))

kable(df.count[c(1:3, 21:25, 101:110), ], , caption = "Number of unresolved, original
name enties per resolved name.")
```

Table 7: Number of unresolved, original name enties per resolved name.

name.short.correct	n
Poaceae	254
Lauraceae	175
Asteraceae	146

name.short.correct	n
Myrcia	47
Taraxacum	42
Pouteria	41
Malpighiaceae	39
Guarea	37
Annonaceae	17
Antimima	17
Brosimum	17
Centaurium erythraea	17
Chrysophyllum	17
Cinnamomum	17
Elymus hispidus	17
Euphorbia	17
Lamium galeobdolon	17
Mollinedia	17

9.3 Based on `unique` standardized names

Generate version of the backbone that only includes the unique resolved names in `name.short.correct`, and for the non-unique names, the first rows of duplicated name:

```
back2.1.uni <- back2.1[!duplicated(back2.1$name.short.correct), ]
```

Remove the first entry, which is NA:

```
back2.1.uni <- back2.1.uni[-1, ]
length(unique(back2.1.uni$name.short.correct))
```

There are 86,760 unique taxon names the in backbone. Exclude the non-vascular plant and non-matching taxon names:

```
df.uni <- back2.1.uni %>%
    dplyr::filter(is.vascular.species == TRUE, !is.na(name.short.correct))
```

`df.uni` had two names less than `df.count`, as they were accidentially tagged as non-vascular species names. Correct that:

```
miss <- df.count$name.short.correct[(df.count$name.short.correct %in% df.uni$name.short.correct) == F]
which(df.count$name.short.correct %in% c("Arabis stellulata", "Coptidium pallasii"))
which(df.uni$name.short.correct %in% c("Arabis stellulata", "Coptidium pallasii"))
df.count[c(20319, 31478), ]
ind <- which(back2.1$name.short.correct %in% c("Arabis stellulata", "Coptidium pallasii"))
```

```
back2.1[ind, 31:34]
```

```
##       family.correct name.short.correct rank.short.correct
## 8853    Brassicaceae  Arabis stellulata            species
## 8932    Brassicaceae  Arabis stellulata            species
## 30191  Ranunculaceae Coptidium pallasii            species
## 96184  Ranunculaceae Coptidium pallasii            species
##       is.vascular.species
## 8853                 TRUE
## 8932                 TRUE
## 30191                TRUE
## 96184                TRUE
```

```
back2.1[ind[1], 31:34] <- back2.1[ind[2], 31:34]
back2.1[ind[3], 31:34] <- back2.1[ind[4], 31:34]

df.uni <- back2.1.uni %>%
    dplyr::filter(is.vascular.species == TRUE, !is.na(name.short.correct))
```

Now, run the stats for unique resolved names (excluding non-vascular and non-matching taxa):

```
length(df.uni$name.short.correct)
```

There are 83,679 unique (non-vascular plant) taxon names:

Database affiliations (`sPlot 2.1` and `TRY 3.0`).

```
kable(t(table(df.uni$sPlot2.1.TRY)), caption = "Number of (standardized) vascular
plant taxon names per unique to, and shared between TRY (S) and sPlot (T).")
```

Table 8: Number of (standardized) vascular plant taxon names per unique to, and shared between TRY (S) and sPlot (T).

S	ST	T
34105	20414	29158

```
table(df.uni$Manual.matching)
```

```
##
## sPlot2b       x
##    1637     490
```

1,432 of those names were matched manually (see above). **Taxonomic ranks:**

```
kable(t(table(df.uni$rank.short.correct)), caption = "Number of (standardized) vascular
plant taxon names per taxonomic rank.")
```

Table 9: Number of (standardized) vascular plant taxon names per taxonomic rank.

family	genus	higher	species
152	4343	13	79169

Taxonomic status:

```
kable(t(table(df.uni$Status.correct)), caption = "Number of (standardized) vascular plant
taxon names that correspond to `Accepted`, `Synonyms` or Unresolved species, respecively.")
```

Table 10: Number of (standardized) vascular plant taxon names that correspond to `Accepted`, `Synonyms` or Unresolved species, respecively.

Accepted	Synonym	Unresolved
69173	9998	4506

9.4 Stats for the corrected names in sPlot only:

```r
df.uni.splot <- df.uni %>%
    dplyr::filter(is.vascular.species == TRUE, !is.na(name.short.correct), df.uni$sPlot2.1.TRY!= "T")
```

```r
length((df.uni.splot$name.short.correct))
```

```
## [1] 54519
```

Database affiliations

```r
kable(t(table(df.uni.splot$sPlot2.1.TRY)), caption = "Number of (standardized) vascular
plant taxon names per unique to sPlot (S), and shared between TRY and sPlot (ST).")
```

Table 11: Number of (standardized) vascular plant taxon names per unique to sPlot (S), and shared between TRY and sPlot (ST).

S	ST
34105	20414

```r
table(df.uni.splot$Manual.matching)
```

```
##
## sPlot2b       x
##    1637     266
```

266 uniquenames in sPlot were matched manually (see above).

Taxonomic ranks:

```r
kable(t(table(df.uni.splot$rank.short.correct)), caption = "Number of (standardized)
vascular plant taxon names per taxonomic rank.")
```

Table 12: Number of (standardized) vascular plant taxon names per taxonomic rank.

family	genus	higher	species
133	2447	9	51930

Taxonomic status:

```r
kable(t(table(df.uni.splot$Status.correct)), caption = "Number of (standardized) vascular
plant taxon names that correspond to `Accepted`, `Synonyms` or Unresolved species, respecively.")
```

Table 13: Number of (standardized) vascular plant taxon names that correspond to `Accepted`, `Synonyms` or Unresolved species, respecively.

Accepted	Synonym	Unresolved
46009	5967	2543

Number of families in sPlot:

```r
length(unique(df.uni.splot$family.correct))
```

```
## [1] 439
```

Done!

---

10 R-settings

```r
sessionInfo()
```

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.5 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libopenblas.so.0
## LAPACK: /usr/lib/lapack/liblapack.so.3.0
##
## locale:
##  [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=de_DE.UTF-8        LC_COLLATE=de_DE.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=de_DE.UTF-8
##  [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] bindrcpp_0.2      rmarkdown_1.6     rgbif_0.9.8
##  [4] Taxonstand_2.0    pbapply_1.3-3     plyr_1.8.4
##  [7] dplyr_0.7.2.9000  doParallel_1.0.10 iterators_1.0.8
## [10] foreach_1.4.3     vegdata_0.9       foreign_0.8-69
## [13] knitr_1.16        stringr_1.2.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.12      highr_0.6         compiler_3.4.1
##  [4] bindr_0.1         tools_3.4.1       digest_0.6.12
##  [7] evaluate_0.10.1   jsonlite_1.5      tibble_1.3.3
## [10] gtable_0.2.0      lattice_0.20-35   pkgconfig_2.0.1
## [13] rlang_0.1.1.9000  yaml_2.1.14       xml2_1.1.1
## [16] httr_1.2.1.9000   rgeos_0.3-23      rprojroot_1.2
## [19] grid_3.4.1        glue_1.1.1        data.table_1.10.4
## [22] geoaxe_0.1.0      R6_2.2.2          oai_0.2.2
## [25] XML_3.98-1.9      sp_1.2-5          whisker_0.3-2
## [28] ggplot2_2.2.1     magrittr_1.5      backports_1.1.0
## [31] htmltools_0.3.6   scales_0.4.1      codetools_0.2-15
## [34] assertthat_0.2.0  colorspace_1.3-2  stringi_1.1.5
## [37] lazyeval_0.2.0    munsell_0.4.3
```

References

Boyle, B., Hopkins, N., Lu, Z., Garay, J.A.R., Mozzherin, D., Rees, T., Matasci, N., Narro, M.L., Piel, W.H., Mckay, S.J. & others. (2013) The taxonomic name resolution service: An online tool for automated standardization of plant names. *BMC Bioinformatics*, **14**, 16.

Chase, M. & Reveal, J. (2009) A phylogenetic classification of the land plants to accompany apgiii. *Botanical Journal of the Linnean Society*, **161**, 122–127.

Federhen, S. (2010) The NCBI Handbook [Internet]. (eds J. McEntyre & J. Ostell) Available from: http://www.ncbi.nlm.nih.gov/guide/taxonomy/; National Center for Biotechnology Information, Bethesda, MD, USA, [Accessed: 25 Oct 2011].

Flann, C. (2009) *Global Compositae Checklist. [Accessed 2 Apr 2013].* Available from: http://compositae.landcareresearch.co.nz/.

International legume database and information service. (2006) *[Internet, Accessed 21 Aug 2015].* Available from: http://www.ildis.org/LegumeWeb.

iPlant Collaborative. (2015) *The Taxonomic Name Resolution Service [Internet]. Version 4.0 [Accessed: 20 Sep 2015].* Available from: http://tnrs.iplantcollaborative.org/; National Center for Biotechnology Information.

Missouri Botanical Garden. (2013) *Tropicos.org [Internet, Accessed 19 Dec 2014].* Available from: http://www.tropicos.org.

The Plant List. (2013) *[Internet]. Version 1.1. [Accessed: 19 Aug 2015].* Available from: http://www.theplantlist.org/.

USDA, NRCS. (2012) *The Plants Database [Internet]. [Accessed: 17 Jan 2015].* Available from: http://plants.usda.gov; National Plant Data Team, Greensboro, NC, USA.