Usando o **plantR** para atualizar informações entre duplicatas

Renato A. F. de Lima (raflima at usp.br)

26 April 2021

Sumário

1 Introdução

Esse tutorial foi pensado para auxiliar curadores de herbário que pretendem atualizar suas coleções usando informações oriundas de duplicatas depositadas em outras coleções. Como realizar esse processo para a coleção inteira demandaria obter as bases de dados de todas as coleções biológicas, os códigos abaixo foram preparados para serem executados por famílias individualmente.

Agradecimentos especiais ao André L. de Gasper pelos testes do pacote e pela sugestão de criar esse tutorial mais dirigido aos curadores de herbário.

2 Obtendo os registros

Antes de começar precisamos carregar o pacote usando a função `library()`, assumindo que você tem o pacote instalado junto ao seu `R`. Se você ainda não tem o pacote instalado em seu computador, siga as instruções do tutorial geral do pacote (i.e. 'plantr_tutorial.html').

```
library("plantR")
```

2.1 Baixando os registros do speciesLink e GBIF

Iremos usar a família Blechnaceae para esse exemplo, e baixando dados tanto do speciesLink (INCT) quanto do GBIF. O GBIF possui mais de 430,000 registros para a família no mundo todo. Assumindo que o curador seja de uma coleção do Brasil, iremos baixar apenas os coletas realizadas no Brasil (i.e. campo `country` igual a 'BR').

Para downloads do GBIF, o limite do número de registros baixados é controlado pelo argumento `n.records` e o padrão é baixar até 5000 registros por item buscado. Estão o limite foi modificado para 450,000 registros, pois o GBIF contém aproximadamente 430,000 registros no total para a família.

Note que o campo `species` da função `rgbif2()` aceita não apenas nomes ao nível de espécie, mas em qualquer nível taxonômico. Para a função `rspeciesLink()`, há um campo específico para familia (i.e. `family`).

```
familia <- "Blechnaceae"
occs_splink <- rspeciesLink(family = familia)
occs_gbif <- rgbif2(species = familia,
                    country = "BR",
                    n.records = 450000)
dim(occs_splink)
```

```
## [1] 24080    51
```

```
dim(occs_gbif)
```

```
## [1] 22803   164
```

Foram encontrados aproximadamente 24,000 registros para a família no speciesLink e aprox. 23,000 registros para o Brasil no GBIF.

2.2 Unindo os registros

Em seguida, combinamos os registros dos dois repositórios em apenas um objeto usando a função `formatDwc()` e removendo colunas desnecessárias para a busca de duplicatas:

```
occs <- formatDwc(splink_data = occs_splink,
                  gbif_data = occs_gbif, drop = TRUE)
dim(occs)
```

```
## [1] 46883    47
```

Veja que o objeto `occs` contém agora a soma dos registros vindos do speciesLink e GBIF (i.e. aprox. 47,000 em Abril de 2021).

3 Padronização e validação de campos relevantes

Mesmo que você esteja interessado apenas em atualizar um tipo de informação (e.g. nome da espécie), é importante padronizar a notação e formato de campos que são importantes para a busca de duplicatas. Essa padronização não é perfeita, mas ela aumenta as chances de recuperar duplicatas com pequenas diferenças de notação em campos como nome e número do coletor, a localidade da coleta, nome da família e a grafia e/ou formato do nome da espécie.

3.1 Padronização das informações

O pacote **plantR** possui funções que executam várias dessas edições de uma só vez. As funções `formatOcc()`, `formatLoc()`, `formatCoord()` e `formatTax()` fazem a padronização dos campos relacionados à coleta em si (e.g. código da coleção, nome e número do coletor), à localidade da coleta, às coordenadas geográficas associadas à coleta e à informação taxonômica.

A verificação ortográfica e de sinônimos via o The Plant List é relativamente demorada. Portanto, iremos fazer a conferência e validação dos nomes apenas usando a Flora do Brasil (i.e. campo `db` igual a 'fbo')

```
occs <- formatOcc(occs)
occs <- formatLoc(occs)
occs <- formatCoord(occs)
occs <- formatTax(occs, db = "fbo")
```

```
## The following family names were automatically replaced:
##
## |Genus         |Old fam.    |New fam.         |
## |:-------------|:-----------|:----------------|
## |Blechum       |Blechnaceae |Acanthaceae      |
## |Didymoglossum |Blechnaceae |Hymenophyllaceae |
## |Phyllitis     |Blechnaceae |Aspleniaceae     |
## |Pteris        |Blechnaceae |Pteridaceae      |
```

Note que as funções não editam as colunas originais dos registros. Elas criam novas colunas que armazenam as informações editadas que em geral tem o mesmo nome da coluna original seguido do sufixo '.new'.

3.2 Validação das informações padronizadas

Para cada conjunto de duplicatas, a tomada de decisão de qual espécimen será usado para obter a informação válida ou a mais atual sobre localidade, coordenadas geográficas ou identificação taxonômica depende de um processo de validação. Esse processo de validação não é essencial para encontrar as duplicatas entre coleções, mas ela ajuda no processo de tomada de decisão sobre qual é a melhor informação disponível.

Assim como para a padronização dos campos, o pacote **plantR** oferece funções que executam essas validações de uma só vez. As funções `validateLoc()` e `validateCoord()` fazem a validação dos campos de localidade e coordenadas geográficas, respectivamente.

```
occs <- validateLoc(occs)
```

```
## [1] "Locality resolution in the original data vs. edited data:"
##                original
## edited          country locality municipality no_info stateProvince
##   country          4363      234            8       0            34
##   locality           10     5933            0       0             0
##   municipality        2    22953         4239       0             0
##   no_info             7        7            2      80             0
##   stateProvince       0     7473          221       0          1317
```

```
occs <- validateCoord(occs)
```

A função `validateTax()` faz uma classificação da confiabilidade da identificação taxonômica dos registros. Essa avaliação é feita em uma base de dados global de nomes de taxonomistas por família (veja o help da função digitando ?validateTax no seu console do R).

Contudo, essa base de dados não é completa. Assim, pode ser importante executar a função de maneira preliminar para avaliar se não há nomes de taxonomistas que sejam relevantes à essa avaliação e que não estejam na base de dados. Vamos portanto visualizar os 15 nomes com mais determinações para a família usado no exemplo, e que não estão listados na base de dados como especialistas da família:

```
temp <- validateTax(occs, top.det = 15)
## Top people with many determinations but not in the taxonomist list:
##
## |Identifier            | Records|
## |:---------------------|-------:|
## |Gonzatti, F.          |    1135|
## |Matos, F.B.           |     959|
## |Almeida, T.E.         |     844|
## |Pietrobom, M.R.       |     842|
## |Kazmirczak, C.        |     646|
## |Sehnem, A.            |     495|
## |(uc, A.R.S.           |     467|
## |Pietrobom-Silva, M.R. |     444|
## |Senna, R.M.           |     441|
## |Smith, A.R.           |     430|
## |Moran, R.C.           |     391|
## |Mickel, J.T.          |     381|
## |Michelon, C.          |     341|
## |Athayde Filho, F.P.   |     333|
## |Prado, J.             |     329|
rm(temp)
```

Note que entre os nomes listados há vários especialistas do grupo, mas que não são exatamente especialistas da família em questão (e.g. 'Gonzatti, F.'). Outros nomes podem estar listados porque a base de dados usada está incompleta.

Para incluir nomes faltantes de taxonomistas, é possível usar o campo `miss.taxonomist`. Nesse exemplo vamos incluir dois nomes 'Kazmirczak, C.' e 'Prado, J.', que tem que ser fornecidos em um formato específico: nome da 'família + underline + nome do determinador.

```
falta.tax <- paste(familia, c("Kazmirczak, C.", "Prado, J."), sep = "_")
occs <- validateTax(occs, miss.taxonomist = falta.tax,
                    top.det = 15)
## Top people with many determinations but not in the taxonomist list:
##
## |Identifier            | Records|
## |:---------------------|-------:|
## |Gonzatti, F.          |    1135|
## |Matos, F.B.           |     959|
## |Almeida, T.E.         |     844|
## |Pietrobom, M.R.       |     842|
## |Sehnem, A.            |     495|
## |(uc, A.R.S.           |     467|
## |Pietrobom-Silva, M.R. |     444|
## |Senna, R.M.           |     441|
## |Smith, A.R.           |     430|
## |Moran, R.C.           |     391|
## |Mickel, J.T.          |     381|
## |Michelon, C.          |     341|
## |Athayde Filho, F.P.   |     333|
## |Caxambu, M.G.         |     291|
## |Barros, I.C.L.        |     289|
```

Note que agora os nomes incluídos já não são mais listados entre os nomes com mais identificações, mas que não são especialistas da família.

4 Preparando os registros para a busca de duplicatas

4.1 Definindo os identificadores únicos

Para a busca de duplicatas, precisamos de algumas informações adicionais. Uma delas é ter um identificador único para cada registro (i.e. número de tombo). Para isso podemos concatenar o código padronizado da coleção e seu número de registro ou tombo na coleção correspondente. Isso pode ser feito usando a função `getTombo()`:

```
occs$numTombo <- getTombo(occs[, "collectionCode.new"],
                          occs[, "catalogNumber"])
```

Como muitas coleções tem suas bases de dados tanto no speciesLink/INCT quanto no GBIF é normal termos números de tombo repetidos nos dados, as chamadas duplicatas virtuais. Como nosso interesse é pelas informações contidas nas duplicatas físicas entre coleções, é indicado remover números de tombo repetidos. Isso não afeta a busca por duplicatas, mas simplifica a sua checagem final, isto é, as duplicatas virtuais não estarão entre as duplicatas encontradas.

Como alguns registros não tem número de tombo (tombo igual a 'US_NA'), usamos o código abaixo para realizar a remoção apenas dos números de tombo repetidos:

```
dim(occs)[1]
```

```
## [1] 46883
```

```
occs <- occs[!duplicated(occs$numTombo) |
               grepl("_NA$", occs$numTombo, perl = TRUE),]
dim(occs)[1]
```

```
## [1] 31553
```

Note que cerca de um terço dos registros representavam duplicatas virtuais.

Note também que existe a possibilidade de coleções terem um mesmo número de tombo para registros diferentes, devido a algum tipo um erro. Mas após o código acima, não estamos mais considerando esse tipo de erro.

4.2 Definindo os campos a serem usados na busca

Uma informação essencial para a busca de duplicatas é a definição de quais campos serão usados na busca por duplicatas. No **plantR** é possível usar qualquer combinação dos campos contendo a informação do nome da família ou espécie, nome ou sobrenome do coletor, número do coletor, ano de coleta e localidade da coleta.

Como há muita variação de notação entre coleções no nome do coletor, é possível usar apenas o seu sobrenome. Além disso, como o uso apenas do sobrenome e número de coletor pode retornar falsas duplicatas para nomes muito comuns e números de coletor baixos (e.g. Lima 10), é aconselhado incluir outros campos na busca. Neste exemplo iremos usar três combinações de campos, que são geradas usando a função `prepDup()`:

```
dups <- prepDup(occs,
                comb.fields = list(c("species","col.name","col.number"),
                                   c("col.last.name","col.number","col.loc"),
                                   c("col.last.name","col.number","col.year")))
```

Note que salvamos os strings concatenados de busca em uma tabela à parte, pois essa informação não será necessária após a busca, evitando aumentar ainda mais o número de novas colunas.

Note também que o padrão da função é não incluir na busca por duplicatas registros com informação faltante de coletor, data ou localidade, etc. Se você quiser incluir esses registros o campo `ignore.miss` deverá ser modificado de `TRUE` para `FALSE`. Mas tome cuidado pois isso pode resultar em muitas falsas duplicatas!!

5 Obtendo as indicações de duplicatas

Finalmente, podemos usar a nova tabela gerada para buscar os registros com alguma indicação de serem um mesmo espécimen com material depositado em diferentes coleções.

```
dups <- getDup(dups)
head(dups[order(dups$dup.ID),], 4)
##          numTombo                                dup.srch.str1                dup.srch.str2
## 1374    ALCB_1558            Blechnaceae sp._Hurbath, F._133           Hurbath_133_mucuge
## 30566 ALCB_100715 Telmatoblechnum serrulatum_Hurbath, F._133           Hurbath_133_mucuge
## 1412    ALCB_9580       Blechnaceae sp._Crestani, A.C.V._201 Crestani_201_licinio almeida
## 7326  ALCB_100921  Blechnum occidentale_Crestani, A.C.V._201 Crestani_201_licinio almeida
##           dup.srch.str3 dup.numb          dup.prop                  dup.ID
## 1374   Hurbath_133_2011        2 0.666666666666667 [ALCB_100715|ALCB_1558]
## 30566  Hurbath_133_2011        2 0.666666666666667 [ALCB_100715|ALCB_1558]
## 1412  Crestani_201_2011        2 0.666666666666667 [ALCB_100921|ALCB_9580]
## 7326  Crestani_201_2011        2 0.666666666666667 [ALCB_100921|ALCB_9580]
```

5.1 Inspecionando os resultados de número de duplicatas encontradas

Novas colunas foram criadas. A coluna 'dup.prop' contém a força de indicação das duplicatas (i.e. quanto mais próximo de 1 mais segurança na indicação). Vamos inspecionar as categorias geradas:

```
table(dups$dup.prop)
##
##                 0 0.333333333333333 0.666666666666667                 1                cc
##             11820              1441              6940              6322              5030
```

Em seguida, precisamos reunir as colunas de interesse da busca de duplicatas com a nossa planilha geral:

```
occs <- cbind.data.frame(occs,
                         dups[,c("dup.ID", "dup.numb", "dup.prop")],
                         stringsAsFactors = FALSE)
```

Para esses dados, cerca de 20% dos registros possuem forte indicação de duplicatas e cerca de 40% não tiveram nenhuma indicação (i.e. prováveis unicatas). Além desses, 22% foram classificados com 'dup.prop' > 0.5 (e.g. 2 sobre 3 combinações iguais) e outros 15% foram marcados como 'cc' de 'cannot check' por falta de informações associadas aos campos escolhidos. Para registros com 'dup.prop' <0.5 (apenas 1 combinação em comum) sugere-se um 'pente-fino', pois esses casos podem representar problemas ou falsas duplicatas.

6 Homogeneizando as informações dentre grupos de duplicatas

Por fim, podemos usar para todas as duplicatas de cada grupo as informações mais confiáveis ou recentes sobre os registros. Atualmente, o **plantR** faz essa homogeneização para as informações de localidade, coordenadas geográficas e/ou taxonomia. Ela é feita usando a função `mergeDup()` cujo padrão é fazer apenas o merge para registros com 'dup.prop' > 0.75. Isso pode ser alterado usando o campo `prop` da função, como abaixo.

```
occs <- mergeDup(occs, prop = 0.65)
```

O default dessa função usa os campos já editados pelo **plantR** e cria novas colunas com um '1' adicionado ao final do nome da coluna (e.g. coluna 'family.new' após a homogeneização será armazenada na nova coluna 'family.new1'). Lembre-se, contudo, que os nomes das colunas que serão usadas podem ser definidas pelo usuário (e.g. usar a coluna 'family' ao invés da 'family.new').

6.1 Inspecionando os resultados da homogeneização

Podemos comparar as mudanças obtidas para um dos campos, como por exemplo, o campo que define o nível de confiança na identificação do registro. Note que mais de cerca de 700 registros que possuíam nível definido como 'unknown' e 'low' passaram a ter um nível definido como 'high'.

```
table(occs$tax.check, occs$tax.check1, dnn = c("antes", "depois"))
##          depois
## antes      high   low unknown
##   high     5813     0       0
##   low       446 13849       0
##   unknown   248     0   11197
```

7 Preparando os dados para a atualização da coleção

7.1 Filtrando campos para atualização de coleções ou de sua base de dados

Para esse tutorial, vamos supor que o André L. de Gasper, atual curador do herbário FURB, queira comparar os materiais depositados no FURB com as demais coleções. Mais especificamente, ele quer atualizar as informações relacionadas à identificação taxonômica e aos campos de determinação.

7.1.1 Qual são os registros da coleção que podem/devem ser atualizadas?

Primeiro, precisamos definir o nome da coleção focal e filtrar quais registros possuem direta ou indiretamente informações que podem ser relevantes. No caso, iremos definir o FURB como a coleção focal.

```
colecao <- "FURB"
occs1 <- occs[grepl(colecao, occs$numTombo, perl = TRUE) |
                grepl(colecao, occs$dup.ID, perl = TRUE), ]
dim(occs1)
## [1] 1737  103
```

Podemos inspecionar quantos registros há por coleção:

```
table(occs1$collectionCode.new)
##
##    ALCB     ASE    BHCB BHCB-SL    CESJ    CGMS     CRI    CTBS    EAFM     EVB    FCAB    FLOR
##       2       1      18      18      24       1      15       1       1       6       7      37
##     FPS    FUEL    FURB     HAS     HCF    HEPH    HSTM    HUCS    HURB    IRAI     JOI    LUSC
##      20      21    1273       1      28      11       2      70      22       1      43       2
##     MBM      MO      NX      NY    PACA      RB     RON   SAMES    SLUI     UEC    UPCB     VIC
##      38       1       3       1       2      30       1       4       5       2      13       7
##    VIES       W
##       4       1
```

Note que há cerca de 1700 registros de Blechnaceae no nosso filtro, dos quais aprox. 1300 são de registros do FURB mesmo, e o restante dos registros estão espalhados em cerca de 40 outras coleções.

Se você quiser fazer o mesmo, mas para outra coleção basta trocar o nome da coleção que é atribuída ao objeto chamado `colecao`.

7.2 Atualização taxonômica

7.2.1 Quais campos podem ser usados na atualização taxonômica?

Podemos filtrar apenas as colunas que são de interesse para a checagem alvo que o André quer realizar, a checagem taxonômica. Para cada um dos campos de interesse iremos filtrar as informações básicas de cada registros, mais a informação original (em formato DwC), a editada pelo **plantR** (sufixo '.new') e a

homogeneizada a partir dos grupos de duplicatas (sufixo '.new1') relativas à identificação taxonômica dos registros.

Note que como usamos os defaults do **plantR** o filtro da colunas também seguem esse padrão. Caso você tenha selecionado outras colunas, os nomes abaixo têm que ser adaptados.

```
colunas <- c("numTombo", "recordedBy.new", "recordNumber.new",
             "identifiedBy", "identifiedBy.new", "identifiedBy.new1",
             "yearIdentified", "yearIdentified.new", "yearIdentified.new1",
             "family", "family.new", "family.new1",
             "scientificName", "scientificName.new", "scientificName.new1",
             "scientificNameStatus", "tax.notes",
             "tax.check","tax.check1","dup.ID")
occs1.tax <- occs1[ , colunas]
```

7.2.2 Algum nome não encontrado ou com notas em relação à Flora do Brasil?

```
table(occs1[grepl("FURB_", occs1$numTombo, fixed = TRUE) &
              !occs1$tax.notes %in% "", c("scientificName","tax.notes")])
##                                                    tax.notes
## scientificName                                      not found was misspelled
##   Blechnum ×caudatum Cav.                                   0              0
##   Blechnum ×leopoldense (Dutra) V.A.O.Dittrich & Salino     0              1
##   Blechnum australe L.                                      1              0
```

Sem grandes problemas para o FURB! Algumas notas em relação a diferença como os híbridos são anotados na Flora do Brasil e a uma espécie que não ocorre no Brasil.

7.2.3 Alguma atualização taxonômica feita nas duplicatas em outras coleções?

Para realizar essa operação, podemos realizar uma filtragem mais, para ficarmos apenas com os registro com indicações de duplicata (i.e. coluna 'dup.ID' não vazia):

```
occs2.tax <- occs1.tax[!is.na(occs1.tax$dup.ID), ]
occs2.tax <- occs2.tax[order(occs2.tax$dup.ID), ]
head(occs2.tax, 7)
##         numTombo    recordedBy.new recordNumber.new    identifiedBy identifiedBy.new identifiedBy.ne
## 893   FURB_50519        Lima, L.V.              112        Lima, LV       Lima, L.V.        Lima, L.
## 6748  CESJ_67491        Lima, L.V.              112       L.V. Lima       Lima, L.V.        Lima, L.
## 791   FURB_54226 Kassner Filho, A.              391    Gasper, A.L.     Gasper, A.L.  Dittrich, V.A.
## 6580  CESJ_71184 Kassner Filho, A.              391 V.A.O. Dittrich Dittrich, V.A.O.  Dittrich, V.A.
## 3745   FURB_6419  Falkenberg, D.B.             6751 Gasper, A.L. de     Gasper, A.L.      Gasper, A.
## 24194 FLOR_24680  Falkenberg, D.B.             6751 Gasper, A.L. de     Gasper, A.L.      Gasper, A.
## 25474 FURB_41755  Falkenberg, D.B.             6751 Gasper, A.L. de     Gasper, A.L.      Gasper, A.
##       yearIdentified yearIdentified.new yearIdentified.new1      family  family.new family.new1
## 893             <NA>               n.d.                n.d. Blechnaceae Blechnaceae Blechnaceae
## 6748            <NA>               2015                2015 Blechnaceae Blechnaceae Blechnaceae
## 791             <NA>               2017                2018 Blechnaceae Blechnaceae Blechnaceae
## 6580            <NA>               2018                2018 Blechnaceae Blechnaceae Blechnaceae
## 3745            <NA>               2017                2017 Blechnaceae Blechnaceae Blechnaceae
## 24194           2010               2010                2017 Blechnaceae Blechnaceae Blechnaceae
## 25474           <NA>               2017                2017 Blechnaceae Blechnaceae Blechnaceae
##                            scientificName   scientificName.new  scientificName.new1
## 893                           Blechnaceae      Blechnaceae sp.      Blechnaceae sp.
## 6748            Blechnum occidentale L. Blechnum occidentale Blechnum occidentale
## 791                           Blechnaceae      Blechnaceae sp. Blechnum auriculatum
```

```
## 6580            Blechnum auriculatum Cav. Blechnum auriculatum Blechnum auriculatum
## 3745                          Blechnaceae      Blechnaceae sp.  Lomaridium plumieri
## 24194                 Blechnum binervatum  Blechnum binervatum  Lomaridium plumieri
## 25474 Lomaridium plumieri (Desv.) C.Presl  Lomaridium plumieri  Lomaridium plumieri
##       scientificNameStatus              tax.notes tax.check tax.check1
## 893        family_as_genus                                  low        low
## 6748        name_w_authors                                  low        low
## 791        family_as_genus                                 high       high
## 6580        name_w_authors                                 high       high
## 3745       family_as_genus                                 high       high
## 24194          possibly_ok check no accepted name          high       high
## 25474       name_w_authors                                 high       high
##                                  dup.ID
## 893             [CESJ_67491|FURB_50519]
## 6748            [CESJ_67491|FURB_50519]
## 791             [CESJ_71184|FURB_54226]
## 6580            [CESJ_71184|FURB_54226]
## 3745  [FLOR_24680|FURB_41755|FURB_6419]
## 24194 [FLOR_24680|FURB_41755|FURB_6419]
## 25474 [FLOR_24680|FURB_41755|FURB_6419]
```

A função `head()` imprimiu os 7 primeiros resultados de duplicatas dos registros do FURB. Os dois primeiros casos (i.e. 'dup.ID' = 'CESJ_67491|FURB_50519' e 'CESJ_71184|FURB_54226') tratam-se de registros identificados a nível de família no FURB mas que tem a identificação completa no CESJ. O primeiro caso, como o determinador 'Lima, L.V.' não está entre os especialistas de Blechnaceae, suas informações não foram homogeneizadas para o registro do FURB (colunas terminando em '.new1'). No segundo caso, a determinação foi feita por Dittrich, V.A.O., especialista da família. Portanto, as novas colunas foram preenchidas automaticamente pelo **plantR** e podem ser usadas pelo curador do FURB para atualização.

No terceiro caso (i.e. FLOR_24680|FURB_41755|FURB_6419), a duplicata do FLOR estava (em 25/04/2021) identificada como *Blechnum binervatum* pelo próprio Gasper, A.L., uma determinação de 2010. No FURB há duas duplicatas, com identificações diferentes, também feitas pelo Gasper, A.L.: uma apenas como Blechnaceae e outra como *Lomaridium plumieri*. Isso é comum pois as duplicatas vêm de um mesmo indivíduo, mas podem ter atributos bem diferentes representados na exsicata (e.g. partes reprodutivas diagnósticas). Ambas as identificações datam de 2017, então nesse caso a identificação com a melhor resolução taxonômica é usada e preenchidas automaticamente. Nesse caso, o curador do FURB pode atualizar a identificação feita ao nível de família.

7.2.4 Salvando os resultados da atualização taxonômica

Para facilitar o trabalho do curador, podemos marcar essas duplicatas com indicação de alteração da identificação:

```
occs2.tax$checar.taxon <-
  occs2.tax$scientificName.new != occs2.tax$scientificName.new1 &
    grepl("FURB_", occs2.tax$numTombo, fixed = TRUE)
```

Em seguida, podemos salvar em uma planilha à parte para conferência, usando a função `saveData()`:

```
nome.arquivo <- paste(colecao, familia, "taxonomia", sep = "_")
saveData(occs2.tax,
         file.name = nome.arquivo,
         dir.name = "checagem_plantR",
         file.format = "csv",  compress = FALSE)
```

Como resultado final, há cerca de 500 registros do FURB que possuem uma duplicata com identificação

mais recente ou em um nível taxonômica mais fino. Esses registros estão marcados como 'TRUE' na coluna 'checar.taxon' que foi salva na pasta 'checagem_plantR' do seu diretório local. Por fins de organização, usamos a combinação do nome da família e da coleção no nome do arquivo '.csv' salvo localmente.

7.3 Atualização de coordenadas geográficas

[EM BREVE]

7.4 Atualização de informações de localidade

[EM BREVE]