Extração, Tarnformação e Carga (ETL) de dados do SUS via Protocolo de Transferência de Arquivos (FTP)

Veja materiais complementares em https://github.com/labxss/curso_r.

Bibliotecas do R

As operações a seguir baixam e carregam os pacotes necessários o para processamento dos dados.

```
if(!require(RCurl)) {install.packages("RCurl"); require(RCurl)}
# funcao getURL

if(!require(downloader)) {install.packages("downloader"); require(downloader)}
# funcao download

if(!require(stringr)) {install.packages("stringr"); require(stringr)}
# lpad str_pad

if(!require(gsubfn)) {install.packages("gsubfn"); require(gsubfn)}

if(!require(readr)) {install.packages("readr"); require(readr)}
# para ler csv diretamente do github
```

pacote read.dbc

> Saiba mais sobre o read.dbc em https://pt.linkedin.com/pulse/datasus-conhe%C3%A7a-nova-ferramenta-para-ler-arquivos-dbc-petruzalek

```
 if(!require(read.dbc)) {devtools::install_github("danicat/read.dbc"); require(read.dbc)}
 # le arquivo DBC da estrategia tabnet/tabwin de disseminacao
```

Parâmetros

A título de exemplo vamos trabalhar com o Protocolo Clínico e Diretriz Terapêutica **Espondilite Ancilosante**, disponível em https://www.gov.br/conitec/pt-br/midias/protocolos/20210428_pcdt-espondilite-ancilosante-1.pdf .

```
cid10=c("M45", "M468") # Espondilite Ancilosante
sigtap=c(
··"0601010019",·#·ADALIMUMABE·(A)·40·MG·INJETAVEL-·SERINGA·PREENCHIDA·(POR·TRATAMENTO·MENSAL)→Revogado·desde·06/2010
··"0604380011",·#·ADALIMUMABE·40·MG·INJETAVEL·(POR·SERINGA·PREENCHIDA)→
··"0604380062",·#·ADALIMUMABE·40·MG·INJETÁVEL·(POR·SERINGA·PREENCHIDA)→
··"0604380097",·#·ADALIMUMABE·40·MG·INJETÁVEL·(FRASCO·AMPOLA)→
··"0604380127",·#·ADALIMUMABE·40·MG·INJETÁVEL·(·POR·SERINGA·PREENCHIDA)(·BIOSSIMILAR·A)→
··"0604380135",·#·ADALIMUMABE·40·MG·INJETÁVEL·(POR·SERINGA·PREENCHIDA)·(BIOSSIMILAR·B)
··"0601010027",·#·ETANERCEPTE·(A)25·MG·INJETAVEL·-FRASCO-AMPOLA·(POR·TRATAMENTO·MENSAL)→Revogado·desde·06/2010
··"0601010051",·#·ETANERCEPTE·50MG·INJETAVEL-·FRASCO·AMPOLA·(POR·TRATAMENTO·MENSAL)→Revogado·desde·06/2010
··"0604380020",·#·ETANERCEPTE·25·MG·INJETÁVEL·(POR·FRASCO-AMPOLA·OU·SERINGA·PREENCHIDA)→
··"0604380038",·#·ETANERCEPTE·50MG·INJETAVEL·(POR·FRASCO-AMPOLA·OU·SERINGA·PREENCHIDA)(ORIGINADOR)→
··"0604380100",·#·ETANERCEPTE·50·MG·INJETÁVEL·(POR·FRASCO-AMPOLA·OU·SERINGA·PREENCHIDA)(BIOSSIMILAR·A)
··"0601010035",·#·INFLIXIMABE·(A)10·MG/ML·10·ML·INJETAVEL·(FRASCO-AMPOLA-·POR·TRATAMENTO·MENSAL)→ Revogado·desde·06/2010
··"0601010043",·#·INFLIXIMABE·10·MG/ML·INJETAVEL·(POR·FRASCO-AMPOLA·10·ML)→ Revogado·desde·06/2010
··"0604380046",·#·INFLIXIMABE·10·MG/ML·INJETAVEL·(POR·FRASCO-AMPOLA·COM·10·ML)→
··"0604380054",·#·INFLIXIMABE·10·MG/ML·INJETAVEL·(POR·FRASCO-AMPOLA·COM·10·ML)→
··"0604380119",·#·INFLIXIMABE·10·MG·/ML·INJETÁVEL·(POR·FRASCO-AMPOLA·COM·10·ML)·(BIOSSIMILAR·A)
··"0604380089",·#·GOLIMUMABE·50·MG·INJETÁVEL·(POR·SERINGA·PREENCHIDA
··"0604380070",·#·CERTOLIZUMABE·PEGOL·200·MG/ML·INJETÁVEL·(POR·SERINGA·PREENCHIDA)
··"0604690029"·#·SECUQUINUMABE·150·MG/ML·SOLUÇÃO·INJETÁVEL·(POR·SERINGA·PREENCHIDA)
) # biologicos


ufs=c('AC', 'AM', 'AP', 'PA', 'RO', 'RR', 'TO',
      'AL', 'BA', 'CE', 'MA', 'PB', 'PE', 'PI', 'RN', 'SE',
      'ES', 'MG', 'RJ', 'SP', 'PR', 'RS', 'SC', 'DF', 'GO', 'MS', 'MT')

ano=18:22
mes=str_pad(1:12, 2, pad="0")

url="ftp://ftp.datasus.gov.br/dissemin/publicos/SIASUS/200801_/Dados/"
```

Estrutura dos dados

Criar tabelas vazias no formato **data.frame** contendo as variáveis desejadas.

O dicionário de dados do Sistema de Informações Ambulatoriais está disponível em ftp://ftp.datasus.gov.br/dissemin/publicos/SIASUS/200801_/Doc/.

Como baixar o dicionário de dados

> **Alternativa 1**
>
> Os enderelos do tipo ftp:// usualmente não funcionam no navegador de internet. Cole o endereço no navegador de arquivos.

> **Alternativa 2**
>
> Baixe usando o R em sua máquina local com a função `download.file`.

```
# R
download.file(
  "ftp://ftp.datasus.gov.br/dissemin/publicos/SIASUS/200801_/Doc/Informe_Tecnico_SIASUS_2019_07.pdf",
  destfile = "Informe_Tecnico_SIASUS_2019_07.pdf"
)
```

```
 paam_estrutura=data.frame(
  PA_AUTORIZ = numeric(),
  PA_CMP = numeric(),
  PA_MVM = numeric(),
  PA_CIDPRI = character(),
  PA_CIDSEC = character(),
  PA_PROC_ID = character(),
  PA_QTDAPR = numeric(),
  PA_SEXO = character(),
  PA_IDADE = numeric(),
  PA_MUNPCN = numeric(),
  uf_processamento  = character(),
  AP_CNSPCN = character()
)

pa_estrutura=data.frame(
  PA_AUTORIZ = numeric(),
  PA_CMP = numeric(),
  PA_MVM = numeric(),
  PA_CIDPRI = character(),
  PA_CIDSEC = character(),
  PA_PROC_ID = character(),
  PA_QTDAPR = numeric(),
  PA_SEXO = character(),
  PA_IDADE = numeric(),
  PA_MUNPCN = numeric(),
  uf_processamento  = character()
)

am_estrutura=data.frame(
  AP_AUTORIZ = numeric(),
  AP_PRIPAL = character(),
  AP_CIDPRI = character(),
  AP_CNSPCN = character()
)
```

Exemplo de dado

O Sistema de Informação Ambulatorial (SIA) apreenta vários subsistemas, a saber:

- **PA Produção Ambulatorial**
- AB Laudo de Acompanhamento à Cirurgia Bariátrica
- ABO Acompanhamento Pós Cirurgia Bariátrica
- ACF Laudo de Confecção de Fístula
- AD Laudos Diversos
- **AM Laudo de Medicamentos**
- AMP Laudo de Acompanhamento Multiprofissional
- AN Laudo de Nefrologia
- AQ Laudo de Quimioterapia
- AR Laudo de Radioterapia
- ATD Laudo de Tratamento Dialítico
- BI Boletim Individual

Os arquivos estão no formado DBF, cujo nome é padronizado:

prefixo | UF | ano com dois dígitos | mês com dois dídigos.

Exemplos: PARR2301.dbc, AMPR2201.dbc, AQBA1801.dbc.

Vamos usar o `download.file` e o `read.dbc` para ler um arquivo direto do diretório FTP.

```
 download.file(
  paste0(url,'PAAC2212.dbc'),
  destfile = "arquivo.dbc"
)
paac2212=read.dbc("arquivo.dbc")
head(paac2212)
```

	PA_CODUNI	PA_GESTAO	PA_CONDIC	PA_UFMUN	PA_REGCT	PA_INCOUT	PA_INCURG
	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>
1	2001586	120000	EP	120040	0000	0000	0000
2	5336171	120000	EP	120020	0000	0000	0000
3	0128619	120000	EP	120040	0000	0000	0000
4	2000393	120000	EP	120070	0000	0000	0000
5	2000970	120000	EP	120034	0000	0000	0000
6	2001063	120000	EP	120040	0000	0000	0000

Lista de arquivos DBC

A lista completa dos arquivos dbc do SIA é obtida com a função `getURL`.

```
url="ftp://ftp.datasus.gov.br/dissemin/publicos/SIASUS/200801_/Dados/"

# lista arquivos dbc do diretorio FTP
aux <-
  getURL(
    url,
    verbose = TRUE,
    ftp.use.epsv = FALSE,
    dirlistonly = TRUE,
    crlf = TRUE
  )
listadbc=strsplit(aux, "\r*\n")[[1]]

length(listadbc) # numero de arquivos

listadbc[sample(1:length(listadbc),20, replace = TRUE)]
# amostra de arquivos DBC
```

```
43158
'ADPR1109.dbc' · 'PASC1803.dbc' · 'ADMT1204.dbc' · 'BIRJ1808.dbc' · 'AQTO2104.dbc' ·
'AMPSC2110.dbc' · 'PAPE0904.dbc' · 'ATDMA1811.dbc' · 'PACE1207.dbc' · 'AQGO1906.dbc' ·
'BIAC1208.dbc' · 'PAGO0902.dbc' · 'BIAP2107.dbc' · 'AMPI0904.dbc' · 'RSES1411.dbc' ·
```

Lista de UF por mês de competência

O controle da ordem de manipulação dos arquivos segundo estado e mês é fundamental na carga.

A função `expand.grid` é empregada para gerar as combinações de UF, ano e mês que integram o nome dos arquivos a serem manipulados.

```
 uf='GO'
 aux=as.matrix(expand.grid(uf,ano,mes))
 ufaamm=sort(paste0(
   aux[,1],
   aux[,2],
   aux[,3]
 ))
 ufaamm
```

```
'GO1801' · 'GO1802' · 'GO1803' · 'GO1804' · 'GO1805' · 'GO1806' · 'GO1807' · 'GO1808' ·
'GO1809' · 'GO1810' · 'GO1811' · 'GO1812' · 'GO1901' · 'GO1902' · 'GO1903' · 'GO1904' ·
'GO1905' · 'GO1906' · 'GO1907' · 'GO1908' · 'GO1909' · 'GO1910' · 'GO1911' · 'GO1912' ·
'GO2001' · 'GO2002' · 'GO2003' · 'GO2004' · 'GO2005' · 'GO2006' · 'GO2007' · 'GO2008' ·
```

Enriquecimento do subsistema principal do SIA

O subsistema principal do **SIA** é chamado de corpo da Autorização de Procedimentos Ambulatoriais (antiga sigla APAC) e são disseminados sob o prefixo **PA**.

Por meio do número de autorização, em geral, **PA_AUTORIZ** ou **AP_AUTORIZ**, é possível completar o arquivo principal.

A tarefa é fundamental para computar o número de usuários com quantidade aprovada de dado procedimento, sobretudo a partir do arquivo de medicamentos de prefixo **AM**.

Veja o exemplo ilustrado para o primeiro valor no índice do vetor `ufaamm`.

Processamento dos arquivos PA da mesma UF e mês

Eventualmente arquivos do mês mês que ultrapassam cerca de dois milhões de registros são fragmentados e mais de um arquivo, por exemplo, **PASP2301a.dbc**, **PASP2301b.dbc** e **PASP2301c.dbc**.

Por isso é utilizada a estrutura de repetição `for`.

Note que os filtros de **CID10** e procedimento **SIGTAP** são usados logo após a carga para reduzir a ocupação na memória e, portanto, o recurso computacional necessário.

Foram estabelecidos filtros segundo a doença selecionada em `PA_CIDPRI %in% cid10` e procedimento em `PA_PROC_ID %in% sigtap`.

Adicionalmente, o vetor de atributos foi aplicado a fim de desprezar os demais.

```
c("PA_AUTORIZ","PA_CMP","PA_MVM" ,"PA_CIDPRI", "PA_CIDSEC",
  "PA_PROC_ID", "PA_QTDAPR","PA_SEXO", "PA_IDADE", "PA_MUNPCN"
)
```

> **Nota:** É recomendável baixar previamente os arquivo DBC para o disco local Assim, basta substituir a linha de comando
>
> `download.file(paste0(url,listadbc_pa[j]), destfile = "arquivo.dbc")`
>
> por
>
> `arquivo=paste0(dirdbc,listadbc_pa[j])`

```
i=1
ufaamm[i]

# filtra apenas arquivos contendo o respectivo UFAAMM
listadbc_pa=subset(
      listadbc,
      grepl(paste0("^PA",ufaamm[i]), listadbc)
    )
listadbc_pa

pa=pa_estrutura

# incorpora o arquivo PA
    for (j in 1:length(listadbc_pa)) {
      arquivo = "arquivo.dbc"
      download.file(paste0(url,listadbc_pa[j]), destfile = arquivo)
      # arquivo=paste0(dirdbc,listadbc_pa[j])
      aux=subset(
        read.dbc(arquivo)[,c(
          "PA_AUTORIZ","PA_CMP","PA_MVM" ,"PA_CIDPRI", "PA_CIDSEC",
          "PA_PROC_ID", "PA_QTDAPR","PA_SEXO", "PA_IDADE", "PA_MUNPCN"
        )],
        PA_CIDPRI %in% cid10 &
          PA_PROC_ID %in% sigtap
      )

      if (nrow(aux)==0) {
        aux=pa
      } else {
        aux$uf_processamento = uf
        pa=rbind(pa,aux)
      }

    }
head(pa)
```

```
'GO1801'
'PAGO1801.dbc'
```

A data.frame: 6 × 11

Processamento do arquivo AM por UF e mês

Analogamente ao que ocorreu com o arquivo **PA**, é processado o arquivo **AM**.

```
listadbc_am=subset(
  listadbc,
  grepl(paste0("^AM",ufaamm[i]), listadbc)
)

   for (j in 1:length(listadbc_am)) {
    arquivo = "arquivo.dbc"
    download.file(paste0(url,listadbc_am[j]), destfile = arquivo)
    # arquivo=paste0(dirdbc,listadbc_am[j])
    if (file.exists(arquivo)) {
      aux=subset(
        read.dbc(arquivo)[,c(
          "AP_AUTORIZ", "AP_PRIPAL",
          "AP_CIDPRI",  "AP_CNSPCN"
        )],
        AP_CIDPRI %in% cid10 & AP_PRIPAL %in% sigtap
      )
      am=rbind(am,aux)
    }
  }
head(am)
```

```
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
ERROR while rich displaying an object: Error in gsub(chr, html_specials[[ch

Traceback:
1. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
2. tryCatchList(expr, classes, parentenv, handlers)
3. tryCatchOne(expr, names, parentenv, handlers[[1L]])
4. doTryCatch(return(expr), name, parentenv, handler)
5. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
6. repr::mime2repr[[mime]](obj)
7. repr_markdown.data.frame(obj)
8. repr_matrix_generic(obj, "\n%s\n\n%s%s\n", sprintf("|%%s\n|%s|\n",
 .     underline), NULL, " <!--/--> |", " %s |", "%s", "|%s\n",
 .     " %s |", " %s |", escape_fun = markdown_escape, rows = rows,
 .     cols = cols, ...)
9. lapply(seq_len(nrow(x)), function(r) {
 .     row <- escape_fun(slice_row(x, r))
 .     cells <- sprintf(cell, row)
 .     if (has_rownames) {
 .         row_head <- sprintf(row_head, escape_fun(rownames(x)[[r]]))
 .         cells <- c(row_head, cells)
 .     }
 .     sprintf(row_wrap, paste(cells, collapse = ""))
 . })
10. FUN(X[[i]], ...)
11. escape_fun(slice_row(x, r))
12. html_escape(values, do_spaces = FALSE)
13. gsub(chr, html_specials[[chr]], text, fixed = TRUE)
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
“input string 4 is invalid in this locale”
Warning message in FUN(X[[i]], ...):
```

```
“input string 4 is invalid in this locale”
```

Substituição dos caracteres especias do CNS criptografado

A fim de agilizar o processamento, os caracteres especiais do **Cartão Nacional de Saúde** (CNS) criptografado são substituídos por números.

```
“input string 4 is invalid in this locale”
```

```
am2=unique(am[,c("AP_AUTORIZ","AP_CNSPCN")])

 am2$AP_CNSPCN=
   gsubfn(
     ".",
     list(
       "{" = "0", "}" = "9", "~" = "5",
       "\177" = "7", "Ç" = "6", "ä" = "8",
       "ü" = "4", "é" = "1", "|" = "2", "â" = "3"
     ),
     iconv(am2$AP_CNSPCN, "CP861", "UTF-8")
   )
head(am2)
```

A data.frame: 6 × 2

	AP_AUTORIZ	AP_CNSPCN
	<fct>	<chr>
344	5217203327765	109801621286415
345	5217203183050	100405717722145
346	5217203322628	383007002133371
351	5218200226162	383009000960458
352	5218200062010	383007087503755
353	5217202866777	104304919345396

```
“input string 4 is invalid in this locale”
```

Junção do PA e AM

A junção ocorre segundo código da autorização. Note a incorporação do atributo `AP_CNSPCN`. Se mais atributos forem necessários basta incorporar ao vetor do arquivo `AM`.

```
Warning message in FUN(X[[i]], ...):
```

```
 paam = merge(
   pa, am2,
   by.x=c("PA_AUTORIZ"),
   by.y=c("AP_AUTORIZ"),
   all.x = TRUE
 )
head(paam)
```

A data.frame: 6 × 1

	PA_AUTORIZ	PA_CMP	PA_MVM	PA_CIDPRI	PA_CIDSEC	PA_PROC_ID	PA_QTDAPR
	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<int>
1	5217202706364	201801	201801	M45	0000	0604380011	2
2	5217202716407	201801	201801	M45	0000	0604380054	3
3	5217202751365	201801	201801	M45	0000	0604380054	3
4	5217202756403	201801	201801	M45	0000	0604380011	2
5	5217202763003	201801	201801	M45	0000	0604380054	0
6	5217202763400	201801	201801	M45	0000	0604380011	2

```
Warning message in FUN(X[[i]], ...):
```

Veja o código completo para baixar e integrar vários arquivos em

https://github.com/labxss/curso_r/blob/main/dbc2csv_cursoR.R

```
“input string 4 is invalid in this locale”
```

Modelagem de dados para cálculo do impacto orçamentário

Agregação anual para avaliação da quantidade aprovada e número de usuários

A título de exemplo vamos carregar os dados de RR de 2018 a 2022.

> A redução deve-se ao grande poder computacional necessário para processar os dados completos do território nacional, disponíveis a partir de 2008. O processamento completo em linguagem R requer um computador com maior capacidade computacional, em geral, 16 a 32GB de RAM e processador comparável ao i7 ou superior.

Para facilitar a análise vamos atribuir uma sigla comum para o mesmo medicamento dotado de diferentes códigos.

```
“input string 4 is invalid in this locale”
```

```
options(scipen=999) # para evitar notacao cientifica
paam_atendimento=read_csv("https://raw.githubusercontent.com/labxss/curso_r/main/csv/sia_pa_RR.csv")
paam_atendimento$ano=substr(paam_atendimento$PA_CMP,1,4)
head(paam_atendimento)
```

```
New names:
• `` -> `...1`
Rows: 11123 Columns: 13
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (6): PA_CIDPRI, PA_CIDSEC, PA_PROC_ID, PA_SEXO, PA_IDADE, uf_processame
dbl (7): ...1, PA_AUTORIZ, PA_CMP, PA_MVM, PA_QTDAPR, PA_MUNPCN, AP_CNSPCN

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this me
```

A tibble:

...1	PA_AUTORIZ	PA_CMP	PA_MVM	PA_CIDPRI	PA_CIDSEC	PA_PROC_ID	PA_QT
<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<chr>	<(
1	1417200234318	201801	201801	M45	0000	0604380038	
2	1417200234550	201801	201801	M45	0000	0604380011	
3	1417200234879	201801	201801	M45	0000	0604380011	
4	1417200235176	201801	201801	M45	0000	0604380011	
5	1417200236727	201801	201801	M45	0000	0604380011	
6	1417200234318	201801	201801	M45	0000	0604380038	

```
Warning message in FUN(X[[i]], ...):
```

Assinalar uma siga para os diferentes códigos do medicamento.

```
“input string 4 is invalid in this locale”
```

```
paam_atendimento$medicamento=""
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0601010019'] <- 'ADA'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0601010019'] <- 'ADA'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380127'] <- 'ADA'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380097'] <- 'ADA'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380011'] <- 'ADA'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380062'] <- 'ADA'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380135'] <- 'ADA'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380070'] <- 'CER'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0601010027'] <- 'ETA25'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380020'] <- 'ETA25'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380100'] <- 'ETA50'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380038'] <- 'ETA50'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0601010051'] <- 'ETA50'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380089'] <- 'GOL'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0601010035'] <- 'INF'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380119'] <- 'INF'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0601010043'] <- 'INF'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380046'] <- 'INF'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604380054'] <- 'INF'
paam_atendimento$medicamento[paam_atendimento$PA_PROC_ID == '0604690029'] <- 'SEC'

head(paam_atendimento)
```

...1	PA_AUTORIZ	PA_CMP	PA_MVM	PA_CIDPRI	PA_CIDSEC	PA_PROC_ID	PA_QT
<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<chr>	<chr>	<(
1	1417200234318	201801	201801	M45	0000	0604380038	
2	1417200234550	201801	201801	M45	0000	0604380011	
3	1417200234879	201801	201801	M45	0000	0604380011	
4	1417200235176	201801	201801	M45	0000	0604380011	
5	1417200236727	201801	201801	M45	0000	0604380011	
6	1417200234318	201801	201801	M45	0000	0604380038	

Uma vez a tabela carregada na variável **paam_atendimento**, onde cada linha representa um atendimento, é necessário extrair o ano do mês de competência **PA_CMP** em função do procedimento SIGTAP **PA_PROC_ID**, somando a quantidade aprovada **PA_QTDAPR** e contando o número de usuários distintos **AP_CNSPCN**.

```
quantidade=aggregate(
  PA_QTDAPR ~ ano + medicamento,
  data = paam_atendimento,
  FUN = sum,
  na.rm = TRUE
)
head(quantidade,20)
```

A data.frame: 16 × 3

	ano	medicamento	PA_QTDAPR
	<chr>	<chr>	<dbl>
1	2018	ADA	3358
2	2019	ADA	4227
3	2020	ADA	3356
4	2021	ADA	2812
5	2022	ADA	1514
6	2022	CER	118
7	2018	ETA50	876
8	2019	ETA50	872
9	2020	ETA50	1240
10	2021	ETA50	360
11	2022	ETA50	92
12	2022	GOL	18
13	2019	SEC	299
14	2020	SEC	596
15	2021	SEC	584
16	2022	SEC	114

```
usuarios=aggregate(
··AP_CNSPCN·~·ano·+·medicamento,·
··data·=·paam_atendimento,·
··function(x)·length(unique(x))
)
head(usuarios,20)
```

A data.frame: 16 × 3

ano	medicamento	AP_CNSPCN
.	.	.

```
paam_anual = merge(
  usuarios, quantidade,
  by.x=c("ano","medicamento"),
  by.y=c("ano","medicamento"),
  all.x = TRUE
)
paam_anual
```

A data.frame: 16 × 4

ano <chr>	medicamento <chr>	AP_CNSPCN <int>	PA_QTDAPR <dbl>
2018	ADA	5	3358
2018	ETA50	2	876
2019	ADA	10	4227
2019	ETA50	1	872
2019	SEC	1	299
2020	ADA	11	3356
2020	ETA50	1	1240
2020	SEC	1	596
2021	ADA	18	2812
2021	ETA50	1	360
2021	SEC	2	584
2022	ADA	24	1514
2022	CER	1	118
2022	ETA50	1	92
2022	GOL	1	18
2022	SEC	1	114

Agora vamos imputar o valor médio unitário multiplicado pela quantidade aprovada.

Aqui foi utilizada a média ponderada (soma da quantidade todal adquirida)

```
valor=read_csv("https://raw.githubusercontent.com/labxss/curso_r/main/bps_valor_unitario.csv")
valor
```

```
Rows: 7 Columns: 6
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (1): medicamento
dbl (5): bps2018, bps2019, bps2020, bps2021, bps2022
```

Agora vamos fazer a junção dos valores atribuídos com os dados de mundo real.

```
valor=read_csv("https://raw.githubusercontent.com/labxss/curso_r/main/bps_valor_unitario_tranposto.csv")

paam_anual_valor = merge(
        paam_anual, valor,
        by.x=c("ano","medicamento"),
        by.y=c("ano","medicamento"),
        all.x = TRUE
      )


# percentual de difusao ao ano do numero de usuarios
paam_anual_valor=transform(paam_anual_valor, usuario_perc = ave(AP_CNSPCN, ano, FUN = prop.table))
paam_anual_valor$usuario_perc=round(paam_anual_valor$usuario_perc*100,1)

# produto entre total aprovado e valor unitario
paam_anual_valor$total=round(paam_anual_valor$PA_QTDAPR*paam_anual_valor$unidade)

# percentual de difusao ao ano do total
paam_anual_valor=transform(paam_anual_valor, total_perc = ave(total, ano, FUN = prop.table))
paam_anual_valor$total_perc=round(paam_anual_valor$total_perc*100,1)

paam_anual_valor$unidade=round(paam_anual_valor$unidade,2)

paam_anual_valor
```

```
Rows: 35 Columns: 3
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (1): medicamento
dbl (2): ano, unidade

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this me
```

A data.frame: 16 × 8

ano	medicamento	AP_CNSPCN	PA_QTDAPR	unidade	usuario_perc	total	to
<chr>	<chr>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	
2018	ADA	5	3358	2532.56	71.4	8504336	
2018	ETA50	2	876	1435.36	28.6	1257375	
2019	ADA	10	4227	2555.51	83.3	10802141	
2019	ETA50	1	872	1493.86	8.3	1302646	
2019	SEC	1	299	2366.09	8.3	707461	
2020	ADA	11	3356	1628.34	84.6	5464709	
2020	ETA50	1	1240	1089.86	7.7	1351426	
2020	SEC	1	596	2786.01	7.7	1660462	
2021	ADA	18	2812	701.16	85.7	1971662	
2021	ETA50	1	360	685.86	4.8	246910	
2021	SEC	2	584	3205.94	9.5	1872269	
2022	ADA	24	1514	507.62	85.7	768537	
2022	CER	1	118	731.51	3.6	86318	
2022	ETA50	1	92	410.41	3.6	37758	
2022	GOL	1	18	2786.35	3.6	50154	
2022	SEC	1	114	2437.32	3.6	277854	

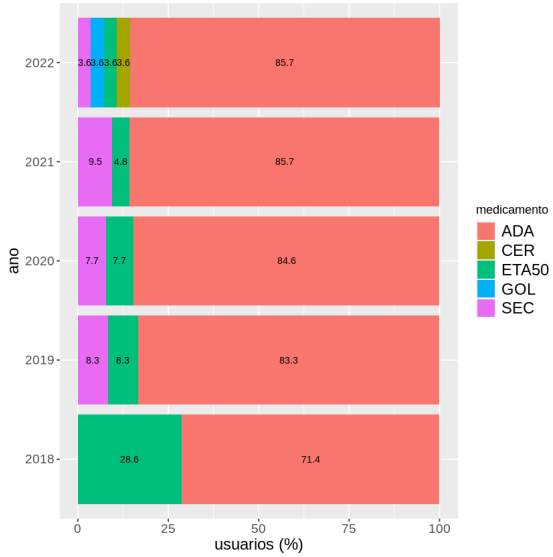
Cálculo da difusão de uso

Vamos explorar visualmente a difusão do percentual de uso com `ggplot`.

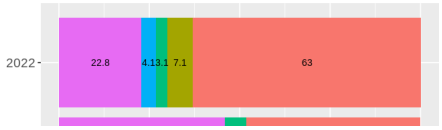
```
if(!require(ggplot2)) {install.packages("ggplot2"); require(ggplot2)} # funcao getURL


ggplot(
```

```
  paam_anual_valor,
  aes(
    fill = medicamento,
    y = usuario_perc,
    x = ano,
    label = usuario_perc
  )
) + geom_bar(stat = "identity") +
  xlab('ano') +
  ylab('usuarios (%)') +
  geom_text(size = 3, position = position_stack(vjust = 0.5)) +
  coord_flip() +
  theme(
    legend.text = element_text(size = rel(1.3)),
    axis.title.x = element_text(size = rel(1.3)),
    axis.title.y = element_text(size = rel(1.3)),
    axis.text.x = element_text(size = rel(1.3)),
    axis.text.y = element_text(size = rel(1.3))
  )
```




```
ggplot(
  paam_anual_valor,
  aes(
    fill = medicamento,
    y = total_perc,
    x = ano,
    label = total_perc
  )
) + geom_bar(stat = "identity") +
  xlab('ano') +
  ylab('valor (%)') +
  geom_text(size = 3, position = position_stack(vjust = 0.5)) +
  coord_flip() +
  theme(
    legend.text = element_text(size = rel(1.3)),
    axis.title.x = element_text(size = rel(1.3)),
    axis.title.y = element_text(size = rel(1.3)),
    axis.text.x = element_text(size = rel(1.3)),
    axis.text.y = element_text(size = rel(1.3))
  )
```




```
 paam_mensal_valor = merge(
        paam_atendimento, valor,
        by.x=c("ano","medicamento"),
        by.y=c("ano","medicamento"),
        all.x = TRUE
      )
# produto entre total aprovado e valor unitario
paam_mensal_valor$total=round(paam_mensal_valor$PA_QTDAPR*paam_mensal_valor$unidade)

    total=aggregate(
  total ~ PA_CMP + medicamento,
  data = subset(
      paam_mensal_valor,
      medicamento == 'ADA'
    ),
  FUN = sum,
  na.rm = TRUE
)
head(total,20)
```

A data.frame: 20 × 3

	PA_CMP	medicamento	total
	<dbl>	<chr>	<dbl>
1	201801	ADA	911700
2	201802	ADA	298835
3	201803	ADA	587540
4	201804	ADA	1154820
5	201805	ADA	835725
6	201806	ADA	557150
7	201807	ADA	820530
8	201808	ADA	536890
9	201809	ADA	790140
10	201810	ADA	258315
11	201811	ADA	759750
12	201812	ADA	992740
13	201901	ADA	1226640
14	201902	ADA	480434
15	201903	ADA	705318
16	201904	ADA	1149975
17	201905	ADA	899536
18	201906	ADA	439546
19	201907	ADA	643986
20	201908	ADA	838204

Projeção com ARIMA

Os dados de mundo real obtidos podem ser processados com técnicas de regressão ou séries temporais.

Vamos ilustrar com o método ARIMA.

```
if(!require(forecast)) {install.packages("forecast"); require(forecast)} # funcao getURL
if(!require(tseries)) {install.packages("tseries"); require(tseries)} # funcao getURL

arima_data=forecast(auto.arima(
    total$total,
    allowmean = TRUE
  ), h = 60)

plot(
  arima_data,
```

```
  xlab = "",
  ylab = "Total (R$)",
  main = "",
  las = 3  ,
  xaxt = 'n'
)

axis(1,
     at = 1:(nrow(total)+61),
     labels = 1:(nrow(total)+61),
     las = 3)

arima_data=as.data.frame(arima_data)
colnames(arima_data)=c("previsão","min80","min95","max80","max95")
tail(arima_data,60)
```

A data.frame: 60 × 5

	previsão	min80	min95	max80	max95
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
61	29811.22	-297012.4	356634.8	-470022.4	529644.8
62	29811.22	-312242.4	371864.9	-493314.7	552937.2
63	29811.22	-326822.7	386445.1	-515613.3	575235.7
64	29811.22	-340829.8	400452.3	-537035.3	596657.8
65	29811.22	-354326.5	413949.0	-557676.8	617299.2
66	29811.22	-367364.9	426987.3	-577617.2	637239.6
67	29811.22	-379988.6	439611.0	-596923.5	656545.9
68	29811.22	-392234.9	451857.3	-615652.6	675275.0
69	29811.22	-404135.7	463758.2	-633853.3	693475.8
70	29811.22	-415718.8	475341.2	-651568.1	711190.5
71	29811.22	-427008.2	486630.7	-668833.8	728456.3
72	29811.22	-438025.3	497647.8	-685683.0	745305.5
73	29811.22	-448788.9	508411.3	-702144.5	761766.9
74	29811.22	-459315.6	518938.1	-718243.8	777866.2
75	29811.22	-469620.6	529243.0	-734003.8	793626.3
76	29811.22	-479717.1	539339.6	-749445.2	809067.6
77	29811.22	-489617.5	549239.9	-764586.4	824208.9
78	29811.22	-499332.6	558955.1	-779444.5	839066.9
79	29811.22	-508872.6	568495.0	-794034.6	853657.0
80	29811.22	-518246.5	577869.0	-808370.8	867993.2
81	29811.22	-527462.8	587085.2	-822465.8	882088.3
82	29811.22	-536529.1	596151.5	-836331.6	895954.0
83	29811.22	-545452.5	605075.0	-849978.8	909601.2
84	29811.22	-554239.6	613862.1	-863417.5	923040.0
85	29811.22	-562896.5	622519.0	-876657.0	936279.5
86	29811.22	-571428.7	631051.2	-889705.9	949328.4
87	29811.22	-579841.6	639464.0	-902572.3	962194.7
88	29811.22	-588139.9	647762.3	-915263.4	974885.9
89	29811.22	-596328.2	655950.7	-927786.4	987408.8
90	29811.22	-604410.8	664033.3	-940147.7	999770.2
91	29811.22	-612391.7	672014.2	-952353.5	1011975.9
92	29811.22	-620274.7	679897.1	-964409.4	1024031.8
93	29811.22	-628063.2	687685.6	-976320.8	1035943.3
94	29811.22	-635760.5	695383.0	-988092.9	1047715.4

```
usuarios=aggregate(
  AP_CNSPCN ~ PA_CMP + medicamento,
  data = subset(
      paam_atendimento,
      medicamento == 'ADA'
    ),
  function(x) length(unique(x))
)
tail(usuarios,20)

arima_data=forecast(auto.arima(
    usuarios$AP_CNSPCN,
    allowmean = TRUE
  ), h = 60)

plot(
  arima_data,
  xlab = "",
  ylab = "Usuários",
  main = "",
  las = 3  ,
  xaxt = 'n'
```

```
)

axis(1,
     at = 1:(nrow(total)+61),
     labels = 1:(nrow(total)+61),
     las = 3)

arima_data=as.data.frame(arima_data)
colnames(arima_data)=c("previsão","min80","min95","max80","max95")
tail(arima_data,60)
```

A data.frame: 20 × 3

	PA_CMP	medicamento	AP_CNSPCN
	<dbl>	<chr>	<int>
41	202105	ADA	9
42	202106	ADA	9
43	202107	ADA	12
44	202108	ADA	8
45	202109	ADA	9
46	202110	ADA	7
47	202111	ADA	10
48	202112	ADA	11
49	202201	ADA	11
50	202202	ADA	13
51	202203	ADA	13
52	202204	ADA	13
53	202205	ADA	10
54	202206	ADA	9
55	202207	ADA	11
56	202208	ADA	8
57	202209	ADA	9
58	202210	ADA	9
59	202211	ADA	11
60	202212	ADA	11

A data.frame: 60 × 5

	previsão	min80	min95	max80	max95
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
61	10.68896	8.542094	12.83583	7.405610	13.97232
62	10.82371	8.579313	13.06811	7.391200	14.25623
63	10.95846	8.620599	13.29633	7.383007	14.53392
64	11.09322	8.665479	13.52095	7.380314	14.80612
65	11.22797	8.713570	13.74236	7.382529	15.07340
66	11.36272	8.764550	13.96088	7.389163	15.33627
67	11.49747	8.818148	14.17679	7.399801	15.59513
68	11.63222	8.874132	14.39031	7.414089	15.85035
69	11.76697	8.932304	14.60164	7.431723	16.10222
70	11.90172	8.992492	14.81095	7.452439	16.35100
71	12.03647	9.054543	15.01840	7.476005	16.59694
72	12.17122	9.118325	15.22412	7.502218	16.84023
73	12.30597	9.183719	15.42823	7.530898	17.08105
74	12.44072	9.250622	15.63083	7.561883	17.31956
75	12.57547	9.318937	15.83201	7.595030	17.55592
76	12.71023	9.388581	16.03187	7.630208	17.79024
77	12.84498	9.459477	16.23048	7.667301	18.02265
78	12.97973	9.531555	16.42790	7.706203	18.25325
79	13.11448	9.604752	16.62420	7.746815	18.48214
80	13.24923	9.679010	16.81945	7.789050	18.70941
81	13.38398	9.754276	17.01368	7.832827	18.93513
82	13.51873	9.830502	17.20696	7.878071	19.15939
83	13.65348	9.907641	17.39932	7.924713	19.38225
84	13.78823	9.985654	17.59081	7.972690	19.60378
85	13.92298	10.064501	17.78147	8.021943	19.82402

86	14.05773	10.144146	17.97132	8.072417	20.04305
87	14.19249	10.224556	18.16041	8.124062	20.26091
88	14.32724	10.305701	18.34877	8.176829	20.47764
89	14.46199	10.387551	18.53642	8.230674	20.69330
90	14.59674	10.470079	18.72340	8.285557	20.90792
91	14.73149	10.553259	18.90972	8.341438	21.12154
92	14.86624	10.637069	19.09541	8.398281	21.33420
93	15.00099	10.721485	19.28050	8.456051	21.54593
94	15.13574	10.806485	19.46500	8.514716	21.75677
95	15.27049	10.892052	19.64893	8.574245	21.96674
96	15.40524	10.978164	19.83232	8.634610	22.17588
97	15.53999	11.064805	20.01518	8.695784	22.38420
98	15.67474	11.151958	20.19753	8.757740	22.59175
99	15.80950	11.239607	20.37938	8.820454	22.79854
100	15.94425	11.327736	20.56076	8.883903	23.00459
101	16.07900	11.416332	20.74166	8.948065	23.20993
102	16.21375	11.505379	20.92212	9.012919	23.41458
103	16.34850	11.594867	21.10213	9.078445	23.61855

✓ 0s conclusão: 17:59