API BDE desde R

Banco Central de Chile, 10 de junio de 2020

Cómo acceder a datos de la Base de datos Estadísticos del BCCh

Introducción

En este documento se explicará cómo acceder a datos de la Base de Datos Estadísticos (BDE) del Banco usando R.

Acceso a documentos y formularios:
https://si3.bcentral.cl/estadisticas/Principal1/Web_Services/index.htm

Para acceder a la BDE a través de la API deben solicitarse credenciales de acceso (usuario y contraseña) al correo electrónico contacto_ws@bcentral.cl , y completar lo siguiente: Formulario

Ejemplo

Cómo obtener una base de datos (DataFrame) que incluirá:

- Códigos de serie
- Frecuencia y nombre en español (obtenidos desde el Webservice "SearchSeries")
- Las observaciones existentes dentro de un rango de fechas (obtenidas desde el Webservice "GetSeries")

Requisitos

- Acceso a Internet
- Conocimiento básico de R
- Credenciales de acceso (usuario y contraseña)
- Fecha de inicio (como string en formato aaaa-mm-dd, ej: "2017-01-01")
- Fecha de término (como string en formato aaaa-mm-dd, ej: "2019-01-01")
- Vector con códigos de series a consultar

El Banco provee de dos Webservices que contienen la información necesaria para la elaboración del Dataframe mencionado:

SearchSeries

Utilizando **usuario, contraseña** y una frecuencia de serie **("Daily","Monthly","Quarterly" o "Annual")**, retorna un dataframe con todas las series correspondientes a la frecuencia ingresada (por ejemplo, para todas las series anuales) con los siguientes datos:

- SeriesId
- Frequency
- FrequencyCode

- Observed
- ObservedCode
- SpanishTitle
- EnglishTitle
- firstObservation
- lastObservation
- updatedAt
- createdAt

GetSeries

Utilizando **usuario, contraseña, fecha de inicio, fecha de término y código de serie**, devuelve un dataframe con todas las observaciones en el período comprendido entre **fecha de inicio y fecha de término** para el código de serie consultado. Formalmente, la consulta arrojará los siguientes campos:

- IndexDateString
- KeyFamilyId
- LastModified
- LastModifiedUser
- SeriesId
- DataStage
- Exists
- Description
- DescripIng
- DescripEsp
- StatusCode
- Value

Para este ejercicio en particular, se utilizará sólo los campos "indexDateString", "seriesKey" y "value".

A continuación, se importarán las librerías necesarias para implementar el código. En caso de un error en la siguiente celda, se sugiere verificar la instalación:

```
# 1.
# Importar las librerías necesarias
# httr crea, recibe y procesa las respuestas del Webservice
library(httr)
# XML y xml2, para leer y crear documentos XML y HTML
library(XML)
library(xml2)
# plyr ofrece un set de herramientas para manipulación de datos
library(plyr)
# stringi, para trabajar con cadenas de texto
library(stringi)
```

Luego, se definen los inputs a usar:

```
# 2.
# Creación de inputs para la conexión al Webservice
```

```r
user <- "usuario"
password <- "password"

firstDate <- "2019-12-31"
lastDate <- "2020-05-14"

# A modo de ejemplo, se consultarán 2 series:
# - 'Tasa de política monetaria (TPM) (porcentaje)'
# - 'Tipo de cambio del Dólar observado'
# Cuyos códigos son, respectivamente:
lista_serie <- c("F022.TPM.TIN.D001.NO.Z.D", "F073.TCO.PRE.Z.M")
lista_serie <- unique(lista_serie)
lista_serie <- toupper(lista_serie)
print(lista_serie)
```

```
## [1] "F022.TPM.TIN.D001.NO.Z.D" "F073.TCO.PRE.Z.M"
```

Un listado con todas las series disponibles en Webservice puede ser descargado desde:

https://si3.bcentral.cl/estadisticas/Principal1/Web_Services/Webservices/series.xls

A continuación, se revisa la validez de los códigos de serie ingresados. Luego de esto, se asignan las variables necesarias para crear y hacer la primera consulta al servicio "SearchSeries". Con el objetivo de hacer la consulta más eficiente, se busca una vez, cada una de las distintas frecuencias presentes en la lista de series a consultar:

```r
# 3.
# Se revisa si hay un código inválido. Todos debiesen terminar en d, m, t, o a
# (correspondiente a las frecuencias). En caso de ser inválido, se avisa al
# usuario y se retira de la lista a consultar
lista_serie <- as.vector(lista_serie)
lista_serie_aux <- stri_sub(lista_serie, -1)
lista_serie_aux <- unlist(lapply(lista_serie_aux, tolower))

right_freq <- c("d", "m", "t", "a")
bool_right_ser <- lista_serie_aux %in% right_freq

if (sum(!bool_right_ser) >= 1) {
  print(paste("Las series ", paste(lista_serie[!bool_right_ser], collapse = ", "),
    " son inexistentes. Por favor chequea los codigos",
    sep = ""
  ))
}

lista_serie <- lista_serie[bool_right_ser]

# Se identifican las distintas frecuencias de series que pudieran existir y se
# clasifican por cada tipo
# FrequencyCode será la lista que contendrá los valores únicos de frecuencia
```

```r
FrequencyCode <- unique(stri_sub(lista_serie, -1))
# Ahora, se convierte la inicial de la frecuencia a su nombre, que es el que se
# necesita para hacer la consulta
FrequencyCode <- gsub("A", "ANNUAL", FrequencyCode)
FrequencyCode <- gsub("T", "QUARTERLY", FrequencyCode)
FrequencyCode <- gsub("M", "MONTHLY", FrequencyCode)
FrequencyCode <- gsub("D", "DAILY", FrequencyCode)

print(FrequencyCode)
```

```
## [1] "DAILY"   "MONTHLY"
```

En el siguiente cuadro se realiza la consulta al Webservice "SearchSeries". Nótese que el resultado obtenido muestra las variables de interés únicamente ("seriesId","frequency","spanishTitle"):

```r
# 4.
# Se genera la consulta en formato XML (1/2)
headerFields <- c("Content-Type" = "text/xml; charset=utf-8")

# dfFinal1 es un DataFrame que contendrá los datos del título de la serie y su
# frecuencia
dfFinal1 <- data.frame()

# Se itera dentro del vector FrequencyCode para consultar las distintas
# frecuencias de interés:
for (Frec in FrequencyCode) {
  # Se genera la consulta en formato XML (2/2) usando el usuario, password y
  # frecuencia
  body1 <- paste('<?xml version="1.0" encoding="utf-8"?>
                 <soap:Envelope
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
                 <soap:Body>
                 <SearchSeries xmlns="http://bancocentral.org/">
                 <user>', user, "</user>
                 <password>", password, "</password>
                 <frequencyCode>", Frec, "</frequencyCode>
                 </SearchSeries>
                 </soap:Body>
                 </soap:Envelope>", sep = "")
  # Se realiza la consulta para extraer la información de interés: Títulos,
  # códigos y frecuencias de las series, con 4 intentos en caso de que se
  # pierda la conexión con el Webservice. Estos datos quedan contenidos en
  # dfFinal1
  for (intento1 in 1:4) {
    tryCatch(
      {
```

```r
# Se realiza la consulta usando las cadenas de texto en formato XML
# generadas previamente
response1 <- httr::POST(
  url = "https://si3.bcentral.cl/SieteWs/SieteWS.asmx",
  body = body1,
  add_headers(headerFields)
)
# Se extrae la respuesta en formato XML entregada por el Webservice
response1 <- content(response1)
test1 <- xmlTreeParse(response1)[["doc"]]
# Se convierte el XML extraído en una lista o matriz (dependiendo de su
# forma)
xmltest <- xmlToList(test1[[1]][[1]][[1]][[1]])
# Si hay error en la consulta
if (xmltest$Codigo == "-1") {
  error_gen / 2
}
# En caso de que el objeto xmltest$SeriesInfos sea una matriz, se
# convierte a lista
if (class(xmltest$SeriesInfos) == "matrix") {
  xmltest$SeriesInfos <- apply(xmltest$SeriesInfos, 2, as.list)
}
# Se extraen los nombres de las series, en caso de no estar disponibles
# se rellena con NA
SpanishTitle <- unlist(lapply(xmltest$SeriesInfos, function(x) {
  if (is.null(x$spanishTitle)) {
    NA
  } else {
    x$spanishTitle
  }
}))
# Se extraen las frecuencias, en caso de no estar disponibles se rellena
# con NA
Frequency <- unlist(lapply(xmltest$SeriesInfos, function(x) {
  if (is.null(x$frequencyCode)) {
    NA
  } else {
    x$frequencyCode
  }
}))
# Se extraen los códigos de las series, en caso de no estar disponibles
# se rellena con NA
Series1 <- unlist(lapply(xmltest$SeriesInfos, function(x) {
  if (is.null(x$seriesId)) {
    NA
  } else {
    x$seriesId
```

```
          }
        }))

        # A partir de los datos extraídos, se crea un dataframe (df_aux1) que
        # luego se agrega al dataframe que contendrá todos los datos (dfFinal1)
        df_aux1 <- data.frame(SpanishTitle = SpanishTitle, Frequency = Frequency)
        rownames(df_aux1) <- Series1
        # Se incluyen únicamente las series presentes en el vector lista_serie
        valid_row1 <- intersect(rownames(df_aux1), lista_serie)
        df_aux1 <- df_aux1[valid_row1, ]
        dfFinal1 <- rbind(dfFinal1, df_aux1)
        print(paste("Frecuencia ", Frec, " encontrada. Agregando", sep = ""))
        break
      },
      error = function(e) {
        message(
          "Intento ", intento1, ": La frecuencia ", Frec,
          " no fue encontrada"
        )
        print(e)
        # En caso de error, se espera 20 segundos antes de volver a consultar la
        # frecuencia
        Sys.sleep(20)
        if (intento1 == 4) {
          stop(paste("Frecuencia ", Frec, " no fue encontrada. Deteniendo ",
            "ejecución",
            sep = ""
          ))
        }
      }
    )
  }
}
```

```
## [1] "Frecuencia DAILY encontrada. Agregando"
## [1] "Frecuencia MONTHLY encontrada. Agregando"
```

```
print(dfFinal1)
```

```
##                                                          SpanishTitle
## F022.TPM.TIN.D001.NO.Z.D Tasa de polÃtica monetaria (TPM) (porcentaje)
## F073.TCO.PRE.Z.M                    Tipo de Cambio del DÃ³lar Observado
##                           Frequency
## F022.TPM.TIN.D001.NO.Z.D     DAILY
## F073.TCO.PRE.Z.M           MONTHLY
```

El resultado final de la ejecución anterior es la variable dfFinal1, un dataframe en que cada una de sus filas es un código de serie, y en sus columnas contiene las variables “spanishTitle” y “frequency”. En el siguiente cuadro se consultará el servicio “GetSeries”.

```r
# 5.
# Creación del DataFrame dfFinal, que incluirá los datos numéricos de todas las
# series que se consultarán
dfFinal <- data.frame()
# Creación del vector row_name para asignar los códigos de serie como índices del
# data_frame
row_name <- c()

# Iteración por cada una de las series para consultar los datos numéricos:
for (serie in lista_serie) {
  # Se genera la consulta en formato XML (2/2) usando el usuario, password,
  # fecha de inicio, fecha de término y código de series para obtener los datos
  # numéricos contenidos entre estas fechas
  body <- paste('<?xml version="1.0" encoding="utf-8"?>
                <soap:Envelope
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
                <soap:Body>
                <GetSeries xmlns="http://bancocentral.org/">
                <user>', user, "</user>,
                <password>", password, "</password>
                <firstDate>", firstDate, "</firstDate>
                <lastDate>", lastDate, "</lastDate>
                <seriesIds>
                <string>", serie, "</string>
                </seriesIds>
                </GetSeries>
                </soap:Body>
                </soap:Envelope>", sep = "")

  # Se genera un loop para hacer 4 intentos de consulta por serie. Si tiene
  # éxito, continúa con la siguiente serie
  for (intento in 1:4) {
    tryCatch(
      {
        # Se realiza la consulta usando las cadenas de texto en formato XML
        # generadas anteriormente. Se guarda la respuesta en la variable
        # response
        response <- httr::POST(
          url = "https://si3.bcentral.cl/SieteWs/SieteWS.asmx",
          body = body,
          add_headers(headerFields)
        )
        response <- content(response)
        # Se analiza el código de respuesta, para revisar si esta fue exitosa
        if (xml_text(xml_find_all(response, "//*")[5]) != "0") {
```

```
          error_gen / 2
        }

        # Se extraen las fechas de las observaciones
        Fecha <- xml_text(xml_find_all(response, "//obs/indexDateString"))
        # Se extraen los datos numéricos de las observaciones
        Value <- xml_text(xml_find_all(response, "//obs/value"))

        # Si no hay observaciones disponibles, continúa con la siguiente serie
        if (length(Fecha) == 0) {
          print(paste("Serie ", serie, " no tiene datos para el período de ",
            "interés. Omitiendo",
            sep = ""
          ))
          break
        }
        # Se ordena la información obtenida, dejando como nombre de fila el
        # código de serie y, como columnas, las fechas en formato dd-mm-aaaa
        df_aux <- data.frame(t(data.frame(Value)))
        colnames(df_aux) <- Fecha
        dfFinal <- rbind.fill(dfFinal, df_aux)
        row_name <- append(row_name, serie)
        print(paste("Serie ", serie, " encontrada. Agregando", sep = ""))
        break
      },
      error = function(e) {
        # En caso de error, se envía un mensaje e intenta nuevamente luego de
        # 20 segundos
        print(paste("Intento ", intento, ": La serie ", serie, " no fue",
          "encontrada",
          sep = ""
        ))
        Sys.sleep(20)
        if (intento == 4) {
          print(paste("La serie ", serie, " no fue encontrada. Omitiendo",
            sep = ""
          ))
        }
      }
    )
  }
}
```

```
## [1] "Serie F022.TPM.TIN.D001.NO.Z.D encontrada. Agregando"
## [1] "Serie F073.TCO.PRE.Z.M encontrada. Agregando"
```

```
# En caso de haber sido guardado como lista, se transforma row_name a vector y se
# asigna a los nombres de fila de dfFinal
```

```r
row_name <- unlist(row_name)
rownames(dfFinal) <- row_name
```

La última parte de este tutorial consiste en revisar que las extracciones estén ordenadas por fecha (desde la más antigua a la más nueva) y juntar en una lista de dataframes toda la información extraída, en dfList

```r
# Se ordenan las columnas de dfFinal por fecha, desde la más antigua a la más
# reciente
dfFinal <- dfFinal[, order(as.Date(colnames(dfFinal), format = "%d-%m-%Y"))]
# Se intersecan las series de dfFinal y dfFinal1
rownames_inter <- intersect(rownames(dfFinal), rownames(dfFinal1))
# Se unen los resultados de dfFinal1 con dfFinal. Se guardan en dfList
dfList <- cbind(dfFinal1[rownames_inter, ], dfFinal[rownames_inter, ])
# Se definen los tipos de columnas en el dataframe. String para las primeras 2
# columnas y numéricos desde la 3 en adelante
colsstr <- c(1, 2)
colsnum <- 3:ncol(dfList)
dfList[, colsstr] <- apply(dfList[, colsstr], 2, function(x) as.character(x))
dfList[, colsnum] <- apply(dfList[, colsnum], 2, function(x) {
  as.numeric(x)
})
# Finalmente, se define el encoding de los caracteres. Se utiliza UTF-8
Encoding(dfList[, 1]) <- "UTF-8"
dfList <- split(dfList, f = dfList$Frequency)
dfList <- lapply(dfList, function(x) {
  Filter(function(x) !all(is.na(x)), x)
})
# Se imprime el resultado de dfList
print(dfList)
```

```
## $DAILY
##                                                    SpanishTitle
## F022.TPM.TIN.D001.NO.Z.D Tasa de política monetaria (TPM) (porcentaje)
##                          Frequency 31-12-2019 02-01-2020 03-01-2020 06-01-2020
## F022.TPM.TIN.D001.NO.Z.D     DAILY       1.75       1.75       1.75       1.75
##                          07-01-2020 08-01-2020 09-01-2020 10-01-2020 13-01-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                          14-01-2020 15-01-2020 16-01-2020 17-01-2020 20-01-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                          21-01-2020 22-01-2020 23-01-2020 24-01-2020 27-01-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                          28-01-2020 29-01-2020 30-01-2020 31-01-2020 03-02-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                          04-02-2020 05-02-2020 06-02-2020 07-02-2020 10-02-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                          11-02-2020 12-02-2020 13-02-2020 14-02-2020 17-02-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
```

```
##                           18-02-2020 19-02-2020 20-02-2020 21-02-2020 24-02-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                           25-02-2020 26-02-2020 27-02-2020 28-02-2020 02-03-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                           03-03-2020 04-03-2020 05-03-2020 06-03-2020 09-03-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                           10-03-2020 11-03-2020 12-03-2020 13-03-2020 16-03-2020
## F022.TPM.TIN.D001.NO.Z.D       1.75       1.75       1.75       1.75       1.75
##                           17-03-2020 18-03-2020 19-03-2020 20-03-2020 23-03-2020
## F022.TPM.TIN.D001.NO.Z.D          1          1          1          1          1
##                           24-03-2020 25-03-2020 26-03-2020 27-03-2020 30-03-2020
## F022.TPM.TIN.D001.NO.Z.D          1          1          1          1          1
##                           31-03-2020 01-04-2020 02-04-2020 03-04-2020 06-04-2020
## F022.TPM.TIN.D001.NO.Z.D          1        0.5        0.5        0.5        0.5
##                           07-04-2020 08-04-2020 09-04-2020 13-04-2020 14-04-2020
## F022.TPM.TIN.D001.NO.Z.D        0.5        0.5        0.5        0.5        0.5
##                           15-04-2020 16-04-2020 17-04-2020 20-04-2020 21-04-2020
## F022.TPM.TIN.D001.NO.Z.D        0.5        0.5        0.5        0.5        0.5
##                           22-04-2020 23-04-2020 24-04-2020 27-04-2020 28-04-2020
## F022.TPM.TIN.D001.NO.Z.D        0.5        0.5        0.5        0.5        0.5
##                           29-04-2020 30-04-2020 04-05-2020 05-05-2020 06-05-2020
## F022.TPM.TIN.D001.NO.Z.D        0.5        0.5        0.5        0.5        0.5
##                           07-05-2020 08-05-2020 11-05-2020 12-05-2020 13-05-2020
## F022.TPM.TIN.D001.NO.Z.D        0.5        0.5        0.5        0.5        0.5
##                           14-05-2020
## F022.TPM.TIN.D001.NO.Z.D        0.5
##
## $MONTHLY
##                                          SpanishTitle Frequency 01-12-2019
## F073.TCO.PRE.Z.M Tipo de Cambio del Dólar Observado   MONTHLY   770.3905
##                   01-01-2020 01-02-2020 01-03-2020 01-04-2020 01-05-2020
## F073.TCO.PRE.Z.M   772.6477     796.38   839.3755    853.379   821.8053
```

Este es el resultado final del tutorial: Una lista de dataframes por frecuencia, que por cada fila contiene un código de serie, nombre, frecuencia, y las observaciones existentes entre las fechas especificadas. Si se quisiera agregar más series simplemente se debe agregar el código de serie, como string, a la lista "lista_serie", definida en la segunda celda. A su vez, si se quisiera cambiar las fechas basta con re-definir firstDate y lastDate.

También se incluye este código en formato función para comenzar a hacer consultas de inmediato a la BDE.

Se recuerda que, según los términos y condiciones de uso del Webservice, el usuario podrá requerir un máximo de **5 series por segundo**, correspondientes a 5 consultas al Webservice, por cuenta habilitada, independiente desde la o las direcciones IP que realicen las consultas.