INSTRUCTIVO DE INSTALACIÓN para RStudio
PAQUETE {uefi} (todas las versiones)

Mini instructivo para acelerar el proceso de instalación del paquete `{uefi}`

Paso 1: Descargar archivo fuente

Archivos del tipo *uefi_0.0.0.9xxx.tar.gz*

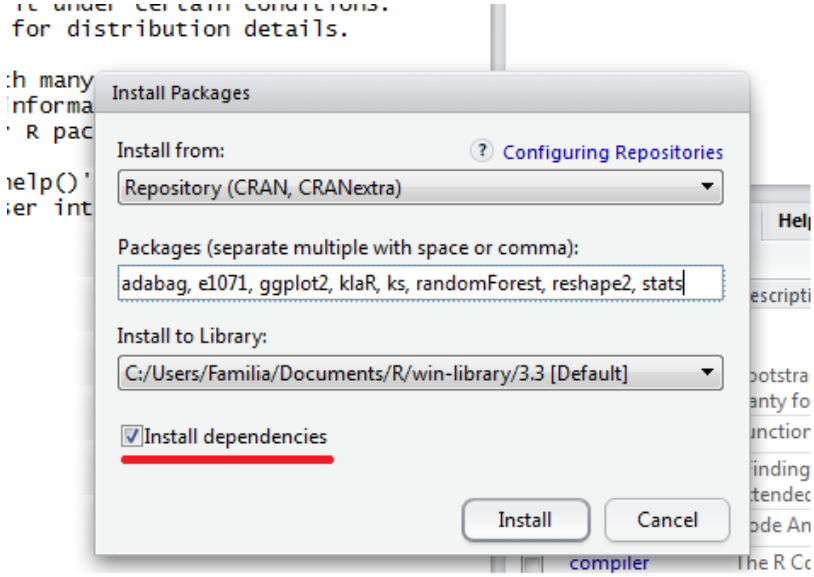
Si estás en Dropbox:

- Entrar, ir a 'Tesis/Actual', bajar archivo "tar.gz/zip" (*uefi_0.0.0.9xxx.tar.gz*) que contiene el *código fuente* del paquete `{uefi}`
- NO descomprimir/usar 'uefi' desde Dropbox; es necesario instalar el paquete en la máquina que estén usando de forma adecuada (ver Pasos 2 a 5)

Paso 2: Instalar estos paquetes (y sus dependencias):

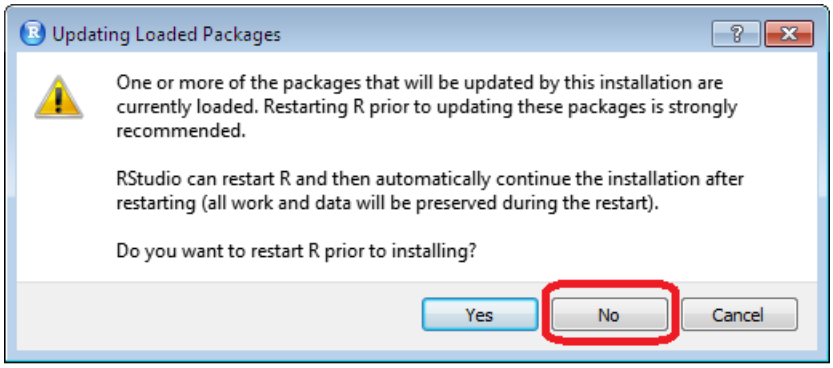
Válido para R >= 3.3.1: (copiar/pegar el siguiente texto tal cual en la ventana indicada)

```
adabag, e1071, ggplot2, klaR, ks, randomForest, reshape2, stats, devtools
```



Otras cosas a considerar

- darle que **NO** a reinciar RStudio e dejar instalar (como próxima imagen)
- constatamos que `{devtools}` es *fundamental* para instalar `{rgl,ks}` sin problemas (al menos en Windows)
- instalacion de todos estos paquetes (y dependencias) demora < 2'



Paso 3: Bajar e instalar paquete {rgl}

Levemente diferente según sistema operativo

Para Windows instalar desde Bioconductor (unica instalacion que funciono en mi caso) con estas dos lineas de comando en la consola:

```
R> source("http://bioconductor.org/biocLite.R")
R> biocLite("rgl")
```

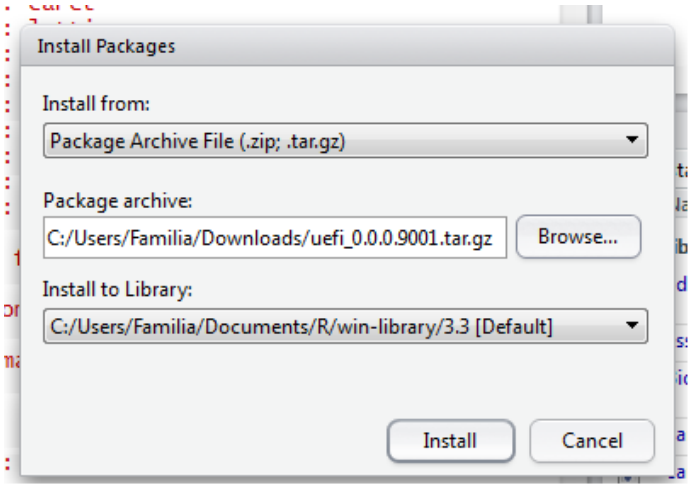
Para MAC/OS instalar *XQuartz* (`https://www.xquartz.org/`), que parece ser un programita de ventanas (fuente `http://stackoverflow.com/questions/33634871/installing-rgl-package-in-r-mac-osx-el-captian-fixed`)

Paso 4: Instalar {ks} y dependencias

Usando «modo normal»: poner `ks` en Packages de la ventana del Paso 2

Paso 5: Instalar archivo fuente de {uefi}

Buscar opción 'instalar desde archivo comprimido':



Paso 6: Corridas

Usar archivo diseñado especialmente para correr funciones ASM y relacionadas:

- En Dropbox: bajar `ve_CorridaEj_vR57.R`
- En otras instancias: usar archivos del tipo 've_ASM_vRnn_xxxx.R' (donde 'nn_xxxx' varían según el número de reunión y otros, ej: `ve_ASM_vR36_1_asmxs.R`)

Paso 7: Ejecución paso a paso

Para entender paso a paso que hace la función `asm2()` y todas las relacionadas a ella, ir mirando 'Doc_ASM_Rutas.txt' detalladamente (apartado al final que explica a asm2()), a medida que se ejecuta una corrida en un archivo de los mencionados en el Paso 6. Se recomienda utilizar la función `debug()` o preferentemente `debugonce()` en RStudio.

Uso debugonce() Esta función permite ver la ejecución paso a paso de cualquier función, además de hacerlo una sola vez (a diferencia de `debug()` que permanece con la función en este modo hasta que se desactive usando `undebug()`)

1. Solicitar el debugging de la función requerida, luego ejecutarla:

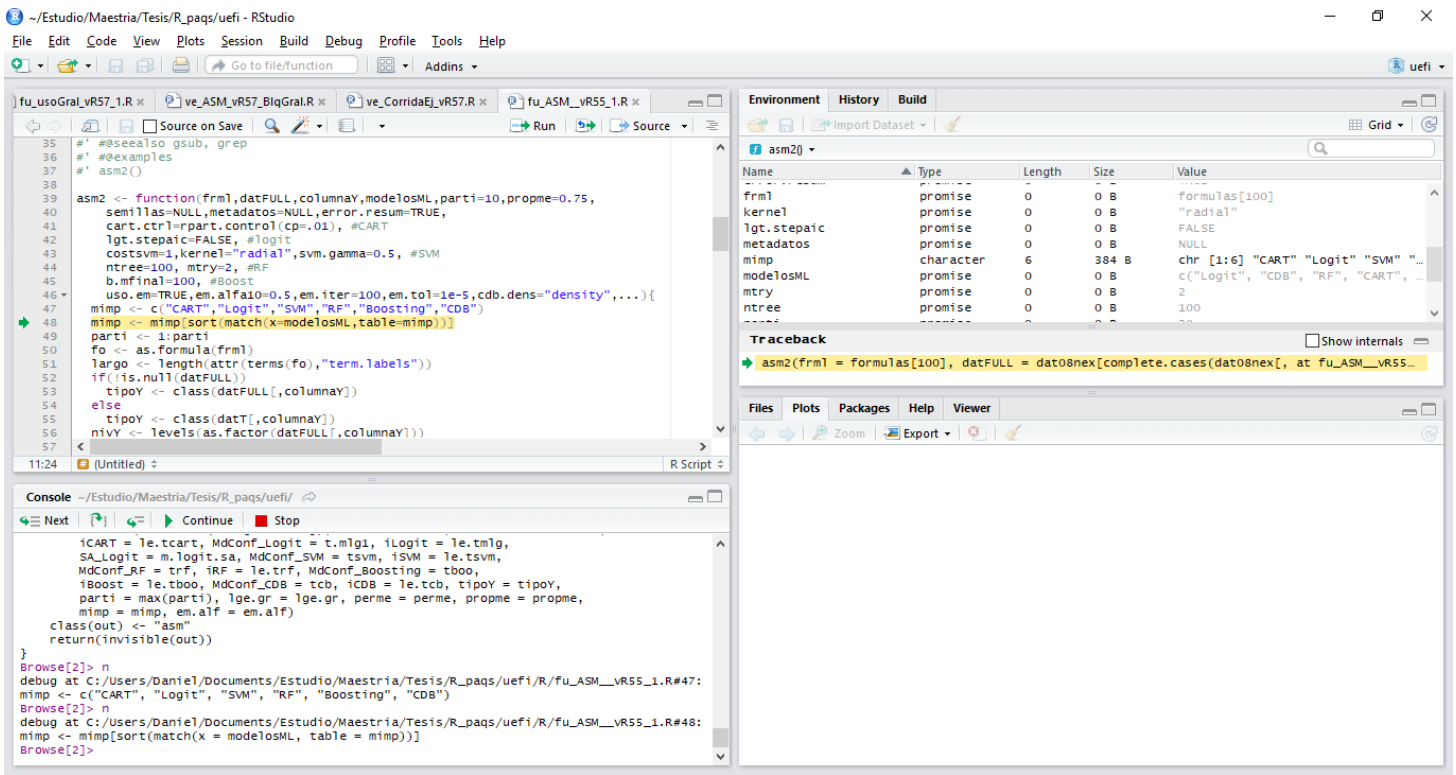
   ```
   R> debugonce(asm2)
   ```

   ```
   R> aju08_ej <- asm2(frml=formulas[100],...)  #resto igual que en script ve_CorridaEj_vR57.R
   ```

2. Ir paso a paso con ejecución: RStudio marcará en amarillo el lugar aproximado donde se encuentre la ejecución de la función

   (a) Presionar F10 para continuar con siguiente paso

   (b) En loops: para correr todo el loop, presionar dentro del mismo f luego Enter; pasará a siguiente paso fuera del loop



3. Para terminar: Q luego Enter

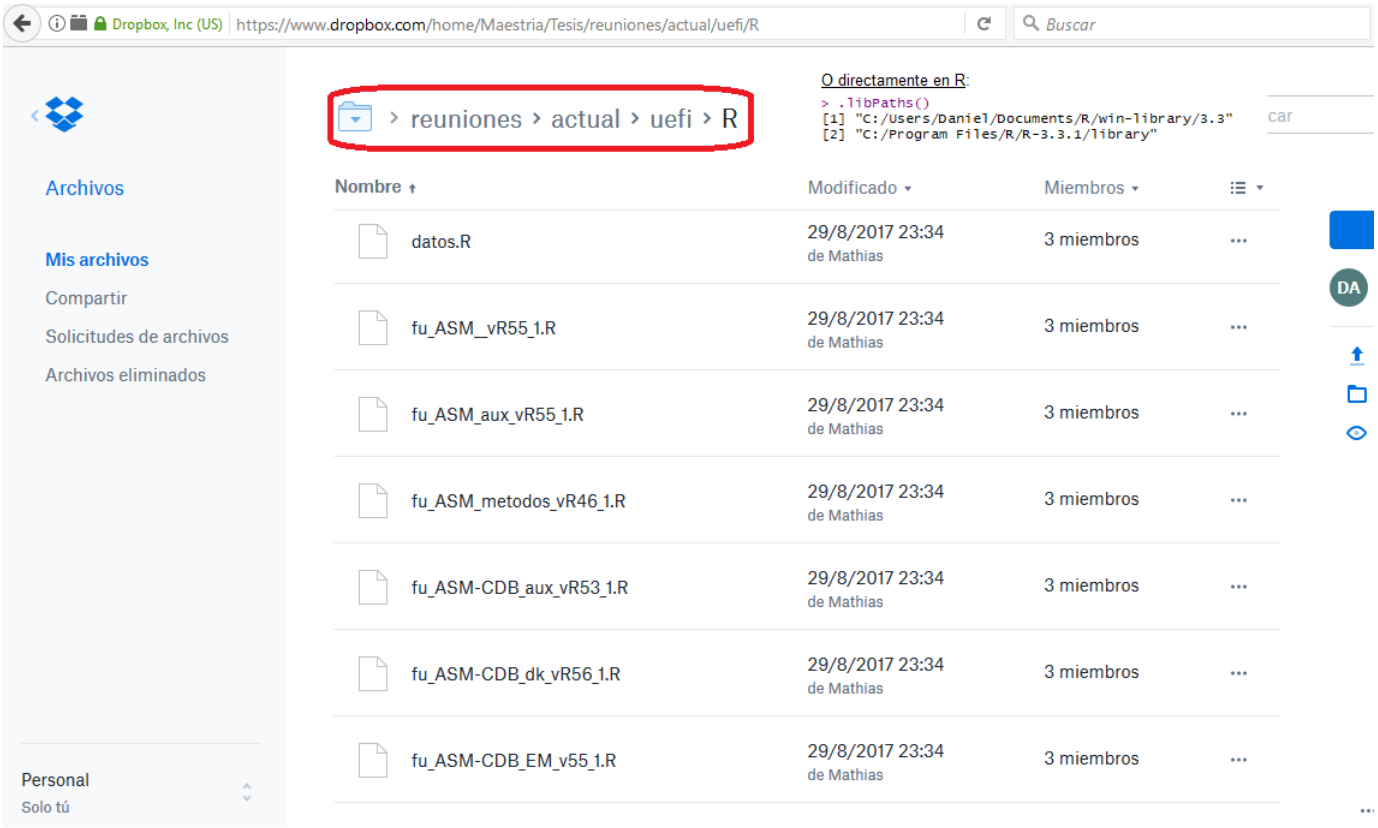
Dejo dos fuentes adicionales de consulta:

- Página oficial de RStudio: `https://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio`
- Grupo MilanoR (más viejo pero interesante): `http://www.milanor.net/blog/visual-debugging-with-rstudio/`

Anexo

Scripts

- En '~[Dropbox]home/Maestria/Tesis/Reuniones/Actual/uefi/R' o en carpeta local de instalación de paquetes (usando la función `.libPaths()` pueden saberlo), similar a '~/{instalacionlocalR}/uefi/R' están todos los scripts utilizados dentro del paquete `{uefi}`



- Breve descripción de archivos y componentes:
  - `datos.R`: contiene los datos a utilizar (sociodemográficos + HDI + CEAM generaciones 2005 y 2008)
  - Los correspondientes al **Ajuste Simultaneo de Modelos (ASM)**, con 'xx' variable (último número de reunión donde el script fue modificado)
    - `fu_ASM__vRxx_1.R`: contiene la principal función del paquete, `asm2()`, que ajusta simultáneamente varios modelos utilizados en el Aprendizaje Automático, considerando sus peculiaridades y manejando cuidadosamente sus partes en común
    - `fu_ASM_metodos_vRxx_1.R`: contiene métodos asociados a asm2: `plot.asm2()` (`print.asm()` suspendido)
    - `fu_ASM_aux_vRxx_2.R`: funciones relacionadas a asm2(): `errp()`(para calcular «error promedio» desde varias permutaciones de datos de entrenamiento/testing), `mdcmed()` (genera indicadores de matriz de confusión), `bestinfo()` (función auxiliar, extrae mejores «promedios» de cada bloque de ejecución (bloques definidos por una secuencia de fórmulas posibles a ejecutar)
    - `fu_ASM-CDB_dk_vRxx_1.R`: función `dk()` –que calcula los valores de las densidades kernel– y metodos (por ahora vacios)
    - `fu_ASM-CDB_pape_vRxx_1.R`: `pape()` –que calcula las probabilidades a posteriori del clasificador bayesiano– y metodos (por ahora vacios)

    * `fu_ASM-CDB_EM_vxx_1.R`: función `emalg2k()` que estima los valores de α para el clasificador bayesiano, mediante el algoritmo EM
    * `fu_ASM-CDB_aux_vRxx_1.R`: contiene muchas funciones auxiliares, en general necesarias para complementar a ASM:
        · Auxiliares para estimación de densidades para variables discretas: `nif()` (genera f_k)
        · Predicción de histogramas o densidades: `predict.nif()/predict.nifM()` (predict para "histograma" categorico uni/multivariado), `predict.hist.x()/predict.hist()` (predict para histograma/densidad variables numericas uni/multivariada), `predict.hm()` (predictor "histograma" mixto: version conjunta multivariada), `plot.cdb_dens()` gráfico "histograma" numérico
        · Separación de muestra de entrenamiento y muestra de prueba o testing: `mute()`
        · Generación de datos con distribución Normal: `gedn()` (genera Normales), `mezcla()` (genera mezcla de Normales),

– Otros archivos con funciones generales:

    * `fu_usoGral_vRxx_1.R`: funciones que sirven para cosas mas generales
        · Generadores de fórmulas: `vdm()` (genera vector formulas Y~X todas las combinaciones posibles)
        · Para data.frames: `ldcmeta()` (genera Libro de Códigos "superficial", es decir con mínima información), `jyo_df()` (junta y ordena data.frame según ranking de primer df usado), `ord_df()` (saca info rankeable más fácilmente de objetos 'asm')
        · Ajuste de parámetros por modelo: `ajp()` (ajuste parametros considerando SVM, CART, Logit y RF simultáneamente)
        · Otras: `vespe()` (frecuencias esperadas en TDEs), `reductable()` (uso futuro MdConf >2 categ), `infoxcol()` (info de variables en df)
    * `fu_Z_otras_vRxx_1.R` (pedido MBo para "apartar funciones viejas")