Installation et administration de R

Version 4.3.1 (2023-06-16)

Équipe de base R

Ce manuel concerne R, version 4.3.1 (2023-06-16).

Table des matières

1Obtention de R

Les sources, les binaires et la documentation pour R peuvent être obtenus via CRAN, le « Comprehensive R Archive Network » dont les membres actuels sont répertoriés sur https://CRAN.R-project.org/mirroirs.html .

1.1 Récupération et décompression des sources Le moyen le plus

simple est de télécharger le fichier Rx.yztar.gz le plus récent et de le décompresser avec

```
tar -xf Rx.yztar.gz
```

sur les systèmes sur lesquels un tar approprié1 est installé. Sur d'autres systèmes, vous devez installer le programme gzip, alors que vous pouvez

```
utiliser gzip -dc Rx.yztar.gz | tar -xf - Le chemin
```

du répertoire dans lequel les sources sont décompressées ne doit pas contenir d'espaces, car la plupart des programmes make (et en particulier GNU make) n'attendent pas d'espaces.

Si vous souhaitez que la version soit utilisable par un groupe d'utilisateurs, définissez umask avant de le décompresser afin que les fichiers soient lisibles par le groupe cible (par exemple, umask 022 soit utilisable par tous les utilisateurs). Conservez ce paramètre d'umask lors de la construction et de l'installation.

Si vous utilisez une version GNU assez récente de tar et que vous le faites en tant que compte root (qui sous Windows inclut les comptes avec des privilèges d'administrateur), vous pouvez voir de nombreux avertissements concernant le changement de propriétaire. Dans ce cas vous pouvez utiliser

```
tar --no-same-owner -xf Rx.yztar.gz
```

et peut-être également inclure l'option --no-same-permissions. (Ces options peuvent également être définies dans la variable d'environnement TAR_OPTIONS : si plusieurs options sont incluses, elles doivent être séparées par des espaces.)

1.2 Obtenir les versions corrigées et de développement

Une version corrigée de la version actuelle, « r-patched », et la version de développement actuelle, « r-devel », sont disponibles sous forme d'archives tar quotidiennes et via l'accès au référentiel R Subversion. (Pendant les deux semaines précédant la sortie d'une version mineure (4.x.0), les archives tar « r-patchées » peuvent faire référence aux versions bêta/release candidates de la version à venir, la version corrigée de la version actuelle étant disponible via Subversion. .)

Les archives tar sont disponibles sur https://stat.ethz.ch/R/daily/. Télécharger R-patched.tar.gz ou R-devel.tar.gz (ou les versions .tar.bz2) et décompressez comme décrit dans la section précédente. Ils sont construits exactement de la même manière que les distributions des versions R.

1.2.1 Utilisation de Subversion et rsync Les sources

sont également disponibles via https://svn.R-project.org/R/, le référentiel R Subversion. Si vous disposez d'un client Subversion (voir https://subversion.apache.org/), vous pouvez consulter et mettre à jour le 'r-devel' actuel depuis https://svn.r-project.org/R/trunk/ et le 'r-patched' actuel de 'https://svn.r-project.org/R/branches/Rxy-branch/' (où x et y sont les numéros majeur et mineur de la version actuelle de R) . Par exemple, utilisez

```
svn checkout https://svn.r-project.org/R/trunk/ chemin
```

pour extraire 'r-devel' dans le chemin du répertoire (qui sera créé si nécessaire).
Les versions alpha, bêta et RC d'une prochaine version xy0 sont disponibles sur « https://svn.r-project.org/R/branches/Rxy-branch/ » dans les quatre semaines précédant la sortie.

---

[1] par exemple GNU tar version 1.15 ou ultérieure, ou celui des distributions 'libarchive' (telle qu'utilisée sur macOS) ou 'Heirloom Toolchest'.

Notez que 'https:' est requis2 et que le certificat SSL du serveur Subversion du projet R doit être reconnu comme provenant d'une source fiable.

Notez que récupérer les sources par exemple avec wget -r ou svn export à partir de cette URL ne fonctionnera pas (et donnera une erreur au début du processus de création) : les informations Subversion sont nécessaires pour construire R.

Le référentiel Subversion ne contient pas les sources actuelles des packages recommandés, qui peuvent être obtenus par rsync ou téléchargés depuis CRAN. Pour utiliser rsync afin d'installer les sources appropriées pour les packages recommandés, exécutez ./tools/rsync-recommended à partir du répertoire de niveau supérieur des sources R.

Si vous téléchargez manuellement depuis CRAN, assurez-vous que vous disposez des versions correctes des packages recommandés : si le numéro dans le fichier VERSION est 'xyz', vous devez télécharger le contenu de 'https://CRAN.R-project.org/ src/contrib/dir', où dir est 'xyz/Recommended' pour r-devel ou xy-patched/ Recommended pour r-patched, respectivement, dans le répertoire src/library/Recommended dans les sources que vous avez décompressées. Après le téléchargement manuel, vous devez exécuter tools/link-recommended à partir du niveau supérieur des sources pour créer les liens requis dans src/library/Recommended. Une incantation appropriée du niveau supérieur des sources R utilisant wget pourrait être (pour la valeur correcte de dir)

```
wget -r -l1 --no-parent -A\*.gz -nd -P src/library/Recommended \
    https://CRAN.R-project.org/src/contrib/dir ./tools/link-
recommended
```

[2] pour certains clients Subversion, « http : » peut sembler fonctionner, mais nécessite une redirection continue.

2 Installer R sous Unix-alikes

R configurera et construira sous la plupart des plates-formes Unix et similaires, y compris 'cpu-*-linux-gnu' pour 'alpha', 'arm64', 'hppa', 'ix86', 'm68k', 'mips', 'mipsel'#, 'ppc64', Processeurs 's390x', 'sparc64' et 'x86_64' , 'x86_64-apple-darwin' et 'aarch64-apple-darwin' ainsi que peut-être (il est testé moins fréquemment sur ces plates-formes) 'i386-sun-solaris' , 'i386- *-freebsd', 'x86_64-*-freebsd', 'i386-*-netbsd', 'x86_64-*-openbsd' et 'powerpc-ibm-aix6*' [1]

De plus, des distributions binaires sont disponibles pour certaines distributions Linux courantes (voir la FAQ pour les détails actuels) et pour macOS. Ceux-ci sont installés de manière spécifique à la plate-forme, donc pour le reste de ce chapitre, nous considérons uniquement la construction à partir des sources.

La construction croisée n'est pas possible : l'installation de R crée une version minimale de R, puis exécute plusieurs Scripts R pour terminer la construction.

2.1 Compilation simple Examinez d'abord les

outils et bibliothèques essentiels et utiles dans l'Annexe A [Autres programmes essentiels et utiles sous Unix-alike], page 39, et installez ceux que vous voulez ou dont vous avez besoin. Assurez-vous que la variable d'environnement TMPDIR n'est pas définie (et que /tmp existe et peut être écrit et que les scripts peuvent être exécutés) ou qu'elle pointe vers le chemin absolu vers un répertoire temporaire valide (un répertoire à partir duquel l'exécution de scripts est autorisée) qui ne le fait pas. ne contient pas d'espaces.[2]

Choisissez un répertoire pour installer l'arborescence R (R n'est pas seulement un binaire, mais contient des ensembles de données supplémentaires, des fichiers d'aide, des métriques de polices, etc.). Appelons cet endroit R HOME. Décompressez le code source. Cela devrait créer des répertoires src, doc et plusieurs autres sous un répertoire de niveau supérieur : accédez à ce répertoire de niveau supérieur (à ce stade, les lecteurs nord-américains devraient consulter la section B.3.1 [Définition du format de papier], page 52.) Émettez le commandes suivantes : ./configure

faire

(Voir Section B.5 [Utilisation de make], page 53, si votre make ne s'appelle pas « make ».) Les utilisateurs de systèmes 64 bits basés sur Debian[3] peuvent avoir besoin

de ./configure LIBnn=lib
faire

Vérifiez ensuite que le système construit fonctionne correctement en

faire un chèque

Les échecs ne sont pas nécessairement des problèmes car ils peuvent être causés par des fonctionnalités manquantes, mais vous devez examiner attentivement toute anomalie signalée. (Certaines erreurs non fatales sont attendues dans les paramètres régionaux qui ne prennent pas en charge Latin-1, en particulier dans les paramètres régionaux True C et les paramètres régionaux non UTF-8 non européens.) Un échec dans tests/ok-errors.R peut indiquer limites de ressources inadéquates (voir Chapitre 5 [Exécution de R], page 22).

Des tests plus complets peuvent être effectués par

faire un contrôle-développement

ou

faire tout vérifier

voir Section 2.8 [Test d'une installation de type Unix], page 14, pour les possibilités de faire cela en parallèle. Notez que ces vérifications ne sont exécutées complètement que si les packages recommandés sont

---

[1] alias « Apple Silicon », connu par certains sous le nom de « arm64-apple-darwin ».

[2] Les espaces étaient déconseillés mais autorisés avant R 4.3.0. qui

[3] utilisent lib plutôt que lib64 pour leurs répertoires de bibliothèques primaires 64 bits : des tentatives sont faites pour détecter de tels systèmes.

installée. Si vous disposez d'un miroir CRAN local, ces vérifications peuvent être accélérées soit en définissant la variable d'environnement R_CRAN_WEB sur son URL, soit en ayant un fichier .R/repositories la spécifiant (voir ?setRepositories).

La création parallèle est prise en charge pour la construction de R mais pas pour les cibles « vérifier » (car la sortie est susceptible d'être entrelacé de manière illisible, bien que là où il est pris en charge, l'option -O de GNU make puisse aider).

Si les commandes configure et make s'exécutent avec succès, un frontal de script shell appelé R sera créé et copié dans R_HOME/bin. Vous pouvez lier ou copier ce script à un endroit où les utilisateurs peuvent l'invoquer, par exemple dans /usr/local/bin/R. Vous pouvez également copier la page de manuel R.1 à un endroit où votre lecteur man la trouve, comme /usr/local/man/man1. Si vous souhaitez installer l'arborescence R complète sur, par exemple, /usr/local/lib/R, voir Section 2.4 [Installation], page 6. Remarque : vous n'avez pas besoin d'installer R : vous pouvez l'exécuter à partir de l'endroit où il se trouvait. construit.

Vous n'êtes pas nécessairement obligé de créer R dans le répertoire source de niveau supérieur (par exemple, TOP_SRCDIR). Pour construire dans BUILDDIR, exécutez

```
cd RÉPERTOIRE CONSTRUCTION
TOP_SRCDIR/configurer la
marque
```

et ainsi de suite, comme décrit plus loin. Cela présente l'avantage de toujours garder votre arborescence source propre et est particulièrement recommandé lorsque vous travaillez avec une version de R issue de Subversion.
(Vous aurez peut-être besoin de GNU make pour autoriser cela, et vous n'aurez besoin d'aucun espace dans le chemin d'accès au répertoire de construction. Il est peu probable que cela fonctionne si le répertoire source a déjà été utilisé pour une construction.)

De nombreux paramètres peuvent être personnalisés lors de la construction de R et la plupart sont décrits dans le fichier config.site dans le répertoire source de niveau supérieur. Cela peut être modifié, mais pour une installation utilisant BUILDDIR, il est préférable de mettre les paramètres modifiés dans un fichier config.site nouvellement créé dans le répertoire de construction.

Maintenant, répétez si nécessaire, tapez R, et lisez les manuels R et la FAQ R (fichiers FAQ ou doc/manual/R-FAQ.html, ou https://CRAN.R-project.org/doc/FAQ/R- FAQ.html qui contient toujours la version de la dernière version de R).

Remarque : si R est déjà installé, vérifiez que l'endroit où vous avez installé R remplace ou apparaît plus tôt dans votre chemin que l'installation précédente. Certains systèmes sont configurés pour avoir /usr/bin (l'emplacement standard pour une installation système) avant /usr/local/bin (l'emplacement par défaut pour l'installation de R) dans leur chemin par défaut, et certains n'ont pas /usr/ local/bin sur le chemin par défaut.

2.2 Options d'aide R fournit par

défaut les pages d'aide sous forme de texte brut affiché dans un pager, avec les options (voir l'aide pour l'aide) d'affichage de l'aide au format HTML ou PDF.

Par défaut, les pages d'aide HTML sont créées en cas de besoin plutôt que d'être construites au moment de l'installation.

Si vous devez désactiver le serveur et souhaitez de l'aide HTML , il existe la possibilité de créer des pages HTML lorsque les packages sont installés (y compris ceux installés avec R). Ceci est activé par l'option de configuration --enable-prebuilt-html. Le fait que R CMD INSTALL (et donc install.packages) pré-construit les pages HTML est déterminé en examinant l'installation de R et est signalé par R CMD INSTALL --help : il peut être remplacé en spécifiant l'une des options INSTALL --html ou

--no-html.

Le serveur est désactivé en définissant la variable d'environnement R_DISABLE_HTTPD sur une valeur non vide, soit avant le démarrage de R, soit dans la session R avant que l' aide HTML (y compris help.start) ne soit utilisée. Il est également possible que des mesures de sécurité du système empêchent le démarrage du serveur, par exemple si l'interface de bouclage a été désactivée. Voir ?tools::startDynamicHelp pour plus de détails.

---

[4] pas par la version fournie par macOS.

2.3 Réalisation des manuels

Il existe un ensemble de manuels qui peuvent être construits à partir des sources,

'plein de gras'
: Versions imprimées de toutes les pages d'aide des packages de base et recommandés (environ 3750 pages).

'refman' Versions imprimées des pages d'aide pour les packages de base sélectionnés (environ 2 200 pages)

'R-FAQ' R FAQ

'R-intro' « Une introduction à R ».

'R-data' « Importation/Exportation de données R ».

'R-admin' « Installation et administration de R », ce manuel.

'R-exts' « Écriture d'extensions R ».

'R-lang' « La définition du langage R ».

Pour les réaliser (avec 'fullrefman' plutôt que 'refman'), utilisez make pdf
: pour créer des versions PDF

faire des informations
: pour créer des fichiers d'informations (pas 'refman' ni 'fullrefman').

Vous ne pourrez créer aucun de ces éléments à moins d'avoir installé texi2any version 5.1 ou ultérieure, et pour les PDF, vous devez avoir installé texi2dvi et texinfo.tex (qui font partie de la distribution GNU texinfo mais sont, en particulier texinfo.tex, souvent fait partie du package TEX dans les redistributions). Le chemin vers texi2any peut être défini par la macro 'TEXI2ANY' dans config.site.
NB : texi2any nécessite Perl.

Les versions PDF peuvent être consultées à l'aide de n'importe quel visualiseur PDF récent : elles comportent des hyperliens qui peuvent être suivis. Les fichiers d'informations peuvent être lus en ligne avec Emacs ou le programme d'information GNU autonome. Les versions PDF seront créées en utilisant le format de papier sélectionné lors de la configuration (ISO a4 par défaut) : cela peut être remplacé en définissant R_PAPERSIZE sur la ligne de commande make, ou en définissant R_PAPERSIZE dans l'environnement et en utilisant make -e. (Si vous refaites les manuels pour un format de papier différent, vous devez d'abord supprimer le fichier doc/manual/version.texi. La valeur habituelle pour l'Amérique du Nord serait « lettre ».)

Il y a quelques problèmes avec la création du manuel de référence PDF, fullrefman.pdf ou refman.pdf. Les fichiers d'aide contiennent à la fois des caractères non-ASCII (par exemple dans text.Rd) et des guillemets droits, dont aucun n'est contenu dans les polices standard LATEX Computer Modern. Nous avons proposé les alternatives suivantes :

fois
: (Par défaut.) Utilisation des polices PostScript standard, Times Roman, Helvetica et Courier. Cela fonctionne bien à la fois pour la visualisation à l'écran et pour l'impression. Un inconvénient est que les sections d'utilisation et d'exemples peuvent être assez larges : ceci peut être surmonté en utilisant en plus l'une des options inconsolata (sur un système Unix uniquement si trouvée par configure) ou beramono, qui remplacent le courrier monospaced. police par Inconsolata ou Bera Sans mono respectivement. (Vous aurez besoin du package LATEX inconsolata[5] ou bera installé.)

  Notez que dans la plupart des installations LATEX, cela n'utilisera pas réellement les polices standard pour PDF, mais intégrera plutôt les clones URW NimbusRom, NimbusSans et (pour Courier, si utilisé) NimbusMon.

  Cela nécessite l'installation des packages LATEX times, helvetic et (si utilisé) courier.

[5] Les instructions sur la façon d'installer la dernière version se trouvent sur https://www.ctan.org/tex-archive/fonts/inconsolata/ .

lm Utilisation des polices Latin Modern. Ceux - ci ne sont pas souvent installés dans le cadre d'une distribution TEX, mais peuvent être obtenus auprès de https://www.ctan.org/tex-archive/fonts/ps-type1/lm/ et de miroirs. Cela utilise des polices plutôt similaires à celles de Computer Modern, mais elles ne sont pas aussi bonnes à l'écran que d'habitude.

La valeur par défaut peut être remplacée en définissant la variable d'environnement R_RD4PDF. (Sur les systèmes Unix, cela sera récupéré au moment de l'installation et stocké dans etc/Renviron, mais peut toujours être remplacé lors de la construction des manuels, en utilisant make -e.) La valeur par défaut habituelle[6] pour R_RD4PDF est 'times,inconsolata, hyper' : omettez 'inconsolata' si vous n'avez pas installé le paquet LATEX inconsolata. Depuis R 4.2.0, 'hyper' est toujours activé (avec une solution de secours si le package LATEX hyperref n'est pas installé).

D'autres options, par exemple pour hyperref, peuvent être incluses dans un fichier Rd.cfg quelque part sur votre chemin de recherche LATEX. Par exemple, si vous préférez créer un lien hypertexte vers le texte et non vers le numéro de page dans la table des matières, utilisez

```
\ifthenelse{\boolean{Rd@use@hyper}}{\hypersetup{linktoc=section}}{}
```

ou

```
\ifthenelse{\boolean{Rd@use@hyper}}{\hypersetup{linktoc=all}}{}
```

pour créer un lien hypertexte à la fois vers le texte et le numéro de page.

Tous les manuels PDF générés peuvent être compactés par

```
faire un compact-pdf
```

à condition que qpdf et gs soient disponibles (voir ?tools::compactPDF pour savoir comment les spécifier s'ils ne sont pas sur le chemin).

Les versions électroniques de la plupart des manuels dans l'un ou les deux formats .epub et .mobi peuvent être réalisé en exécutant dans le doc/manuel l'un des

```
faire des ebooks
faire epub
faire mobi
```

Cela nécessite une conversion de livre électronique depuis Calibre (https://calibre-ebook.com/download) ou depuis la plupart des distributions Linux. Si nécessaire, le chemin d'accès à ebook-convert peut être défini comme make macro EBOOK en éditant doc/manual/Makefile (qui contient une valeur commentée adaptée à macOS) ou en utilisant make -e.

2.4Installation

Pour vous assurer que l'arborescence installée est utilisable par le bon groupe d'utilisateurs, définissez umask de manière appropriée (peut-être sur '022') avant de décompresser les sources et tout au long du processus de construction.

Après

```
./configurer
faire
faire un chèque
```

(ou, lors de la construction en dehors de la source, TOP_SRCDIR/configure, etc.) a été terminé avec succès, vous pouvez installer l'arborescence R complète sur votre système en tapant

```
faire installer
```

Un make parallèle peut être utilisé (mais exécutez make avant make install). Ceux qui utilisent GNU make 4.0 ou version ultérieure voudront peut-être utiliser make -jn -O pour éviter l'entrelacement des sorties.

Cela installera dans les répertoires suivants :

prefix/bin ou bindir le script
shell frontal et d'autres scripts et exécutables

[6] sous Unix, 'inconsolata' est omis s'il n'est pas trouvé par configure.

prefix/man/man1 ou mandir/man1 la page de
manuel

prefix/LIBnn/R ou libdir/R pour tout le
reste (bibliothèques, système d'aide en ligne, . soyez · .). Ici, LIBnn est généralement « lib », mais peut
'lib64' sur certains systèmes Linux 64 bits. Ceci est connu sous le nom de répertoire personnel R.

où le préfixe est déterminé lors de la configuration (généralement /usr/local) et peut être défini en exécutant configure avec l'option --prefix, comme dans

./configure --prefix=/where/you/want/R/to/go

où la valeur doit être un chemin absolu. Cela amène make install à installer le script R dans /where/you/want/R/to/go/bin, et ainsi de suite. Le préfixe des répertoires d'installation est visible dans le message d'état affiché à la fin de la configuration. L'installation devra peut-être être effectuée par le propriétaire du préfixe, souvent un compte root.

Il existe la possibilité d'utiliser make install-strip (voir Section 2.7.1 [Symboles de débogage], page 10).

Vous pouvez installer dans une autre arborescence de répertoires en

utilisant make prefix=/path/to/here install

au moins avec la marque GNU (mais pas avec certaines autres marques Unix).

Un contrôle plus précis est disponible au moment de la configuration via les options : voir configure --help pour plus de détails. (Cependant, la plupart des options 'Réglage fin des répertoires d'installation' ne sont pas utilisées par R.)

Les options de configuration --bindir et --mandir sont prises en charge et déterminent où une copie du R
Le script et la page de manuel sont installés.

L'option de configuration --libdir contrôle où les principaux fichiers R sont installés : la valeur par défaut est 'eprefix/LIBnn', où eprefix est le préfixe utilisé pour installer les fichiers dépendants de l'architecture, la valeur par défaut est le préfixe et peut être définie via l'option de configuration - -préfixe-exec.

Chacun de bindir, mandir et libdir peut également être spécifié dans la commande make install
ligne (au moins pour GNU make).

Les variables configure ou make rdocdir et rsharedir peuvent être utilisées pour installer les répertoires doc et share indépendants du système ailleurs que libdir. Les fichiers d'en-tête C peuvent être installés à la valeur rin Includedir : notez que comme les en-têtes ne sont pas installés dans un sous-répertoire, vous voulez probablement quelque chose comme rin Includedir=/usr/local/include/R-4.3.1.

Si vous souhaitez que le home R soit autre chose que libdir/R, utilisez rhome : par exemple

make install rhome=/usr/local/lib64/R-4.3.1

utilisera un home R spécifique à la version sur un système non Debian Linux 64 bits.

Si vous avez créé R en tant que bibliothèque partagée/statique, vous pouvez l'installer dans le répertoire de bibliothèque de votre système

en make prefix=/path/to/here install-libR

où le préfixe est facultatif et libdir donnera un contrôle plus précis.7 Cependant, vous ne devez pas installer dans un répertoire mentionné dans LDPATHS (par exemple /usr/local/lib64) si vous avez l'intention de travailler avec plusieurs versions de R, car ce répertoire peut avoir la priorité sur le répertoire lib des autres installations R.

créer une bande d'installation

installera les exécutables supprimés, et sur les plates-formes où cela est pris en charge, les bibliothèques supprimées dans les répertoires lib et modules et dans les packages standard.

7 Cela sera nécessaire si plusieurs sous-architectures doivent être installées.

Notez que l'installation de R dans un répertoire dont le chemin contient des espaces n'est pas prise en charge et que certains aspects (tels que l'installation des packages sources) ne fonctionneront pas.

Pour installer les versions info et PDF des manuels, utilisez l'un ou les deux

```
créer des informations d'installation
faire installer-pdf
```

Encore une fois, il est facultatif de spécifier prefix, libdir ou rhome (les manuels PDF sont installés sous le répertoire personnel R).

Un contrôle plus précis est possible. Pour info, le paramètre utilisé est celui d'infodir (préfixe/info par défaut, défini par l'option de configuration --infodir). Les fichiers PDF sont installés dans l'arborescence R doc, définie par la variable make rdocdir.

Une installation par étapes est possible, c'est-à-dire qu'elle installe R dans un répertoire temporaire afin de déplacer l'arborescence installée vers sa destination finale. Dans ce cas, le préfixe (et ainsi de suite) doit refléter la destination finale, et DESTDIR doit être utilisé : voir https://www.gnu.org/prep/standards/ html_node/DESTDIR.html.

Vous pouvez éventuellement installer les tests d'exécution qui font partie de make check-all en

```
faire des tests d'installation
```

qui remplit un répertoire de tests dans l'installation.

2.5 Désinstallation

Vous pouvez désinstaller R en

```
faire la désinstallation
```

en spécifiant éventuellement le préfixe, etc. de la même manière que spécifié pour l'installation.

Cela désinstallera également tous les manuels installés. Il existe des cibles spécifiques pour désinstaller les informations et Manuels PDF dans le fichier doc/manual/Makefile.

Target uninstall-tests désinstallera tous les tests installés, ainsi que le répertoire tests contenant les résultats des tests.

Une libR partagée/statique installée peut être désinstallée en

```
make prefix=/path/to/here uninstall-libR
```

2.6 Sous-architectures

Certaines plates-formes peuvent prendre en charge des versions étroitement liées de R qui peuvent partager tout sauf les exécutables et les objets dynamiques. Les exemples incluent les versions sous Linux pour différents processeurs ou les versions 32 et 64 bits.

R prend en charge l'idée de builds spécifiques à l'architecture, spécifiées en ajoutant 'r_arch=name' à la ligne de configuration. Ici, le nom peut être n'importe quoi non vide et est utilisé pour nommer les sous-répertoires de lib, etc., inclure et les sous-répertoires du package libs. Des exemples de noms provenant d'autres logiciels sont l'utilisation de sparcv9 sur Sparc Solaris et 32 par gcc sur Linux 'x86_64'.

Si vous disposez de deux ou plusieurs versions de ce type, vous pouvez les installer les unes sur les autres (et pour les versions 32/64 bits sur une architecture, une version peut être réalisée sans « r_arch »). Les économies d'espace peuvent être considérables : sur Linux 'x86_64', une installation de base (sans symboles de débogage) prenait 74 Mo, et l'ajout d'une version 32 bits ajoutait 6 Mo. Si vous avez installé plusieurs versions, vous pouvez sélectionner la version à exécuter.

```
R --arch=nom
```

et le simple fait d'exécuter « R » exécutera la dernière version installée.

R CMD INSTALL détectera si plusieurs versions sont installées et tentera d'installer les packages avec les objets de bibliothèque appropriés pour chacun. Cela ne sera pas fait si le package possède un script de configuration exécutable ou un fichier src/Makefile. Dans de tels cas, vous pouvez installer des versions supplémentaires en

```
R --arch=nom CMD INSTALL --libs-only pkg1 pkg2 ...
```

Si vous souhaitez mélanger des sous-architectures compilées sur différentes plates-formes (par exemple « x86_64 » Linux et « i686 » Linux), il est judicieux d'utiliser des noms explicites pour chacune, et vous devrez peut-être également définir libdir pour vous assurer qu'elles s'installent dans le même endroit.

Lorsque des sous-architectures sont utilisées, la version de Rscript dans par exemple /usr/bin sera la dernière installée, mais des versions spécifiques à l'architecture seront disponibles par exemple dans /usr/lib64/R/bin/exec${R_ ARCH}. Normalement, toutes les architectures installées fonctionneront sur la plate-forme, donc l'architecture de Rscript elle-même n'a pas d'importance. L'exécutable Rscript exécutera le script R, et à ce moment-là, le paramètre de la variable d'environnement R_ARCH détermine l'architecture qui est exécutée.

Lors de l'exécution de tests post-installation avec des sous-architectures, utilisez

```
R --arch=nom CMD make check[-devel|all]
```

pour sélectionner une sous-architecture à vérifier.

Les sous-architectures sont également utilisées sous Windows, mais en sélectionnant les exécutables dans le répertoire bin approprié, R_HOME/bin/x64. Pour une compatibilité ascendante, il existe des exécutables R_HOME/bin/R.exe et R_HOME/bin/Rscript.exe : ceux-ci exécuteront un exécutable depuis l'un des sous-répertoires, lequel étant extrait d'abord de la variable d'environnement R_ARCH, puis de --arch option de ligne de commande [8] et enfin à partir de la valeur par défaut de l'installation (qui est 32 bits pour une installation R combinée 32/64 bits). R 4.2.0 suit le schéma, mais prend en charge et inclut uniquement

Versions 64 bits.

2.6.1 Multilib

Pour certaines distributions Linux[9], il existe un mécanisme alternatif permettant de mélanger des bibliothèques 32 bits et 64 bits, appelé multilib. Si la distribution Linux prend en charge multilib, alors les versions parallèles de R peuvent être installées dans les sous-répertoires lib (32 bits) et lib64 (64 bits). Le build à exécuter peut ensuite être sélectionné à l'aide de la commande setarch. Par exemple, une version 32 bits peut être exécutée par

```
setarch i686 R
```

La commande setarch n'est opérationnelle que si les versions 32 bits et 64 bits sont installées. S'il n'y a qu'une seule installation de R, celle-ci sera toujours exécutée quelle que soit l'architecture spécifiée par la commande setarch.

Il peut y avoir des problèmes lors de l'installation de packages sur l'architecture non native. C'est une bonne idée d'exécuter par exemple setarch i686 R pour les sessions dans lesquelles les packages doivent être installés, même si c'est la seule version de R installée (puisque cela indique au code d'installation du package l'architecture nécessaire).

Il existe un problème potentiel avec les packages utilisant Java, car la post-installation d'un RPM 'i686' sur Linux 'x86_64' reconfigure Java et trouvera le Java 'x86_64'. Si vous savez où Java 32 bits est installé, vous pourrez peut-être exécuter (en tant que root)

```
export JAVA_HOME=<chemin vers le répertoire jre de Java 32 bits> setarch i686 R CMD
javareconf pour obtenir un paramètre approprié.
```

Lorsque ce mécanisme est utilisé, la version de Rscript dans par exemple /usr/bin sera la dernière installée, mais une version spécifique à l'architecture sera disponible par exemple dans /usr/lib64/R/bin. Normalement, toutes les architectures installées fonctionneront sur la plate-forme, c'est donc le cas de l'architecture de Rscript.
pas important.

[8] avec les valeurs possibles 'i386', 'x64', '32' et '64'.

[9] principalement sur RedHat et Fedora, dont la mise en page est décrite ici.

2.7 Autres options

Il existe de nombreuses autres options d'installation, dont la plupart sont répertoriées par configure --help.
Presque toutes celles qui ne sont pas répertoriées ailleurs dans ce manuel sont soit des options autoconf standard non pertinentes pour R, soit destinées à des utilisations spécialisées par les développeurs R.

Celle qui peut être utile lorsque vous travaillez sur R lui-même est l'option --disable-byte-compiled-packages, qui garantit que les packages de base et recommandés ne sont pas compilés en octets. (Alternativement, la variable (marque ou environnement) R_NO_BASE_COMPILE peut être définie sur une valeur non vide pour la durée de la construction.)

L'option --with-internal-tzcode utilise le propre code de R et une copie de la base de données IANA pour gérer les fuseaux horaires. Ceci sera préféré lorsqu'il y a des problèmes avec la mise en œuvre du système, impliquant généralement des périodes après 2037 ou avant 1916. Un répertoire de fuseau horaire alternatif10 peut être utilisé, pointé par la variable d'environnement TZDIR : il doit contenir des fichiers tels que Europe/Londres. Sur tous les OS testés, le fuseau horaire du système a été déduit correctement, mais si nécessaire, il peut être défini comme valeur de la variable d'environnement TZ.

Les options --with-internal-iswxxxxx, --with-internal-towlower et --with-internal-wcwidth ont été introduites dans R 4.1.0. Celles-ci contrôlent le remplacement des fonctions de classification des caractères à l'échelle du système (telles que iswprint), de changement de casse (wctrans) et de largeur (wcwidth et wcswidth) par celles contenues dans les sources R. Le remplacement des fonctions de classification est effectué depuis de nombreuses années sur macOS et AIX (et Windows) : l'option --with-internal-iswxxxxx permet de supprimer cela sur ces plateformes ou de l'utiliser sur d'autres. Le remplacement des fonctions de changement de casse était nouveau dans R 4.1.0 et était la valeur par défaut sur macOS (et sur Windows depuis R 4.2.0).

Le remplacement des fonctions de largeur est également effectué depuis de nombreuses années et reste la valeur par défaut.
Ces options n'auront d'importance que pour ceux qui travaillent avec des données de caractères non-ASCII, en particulier dans les langues écrites dans une écriture non occidentale11 (qui inclut des « symboles » tels que les emoji). Notez que l'un de ces iswxxxxx est iswprint qui est utilisé pour décider s'il faut afficher un caractère sous forme de glyphe ou sous forme d'échappement '\U{xxxxxx}' - par exemple, essayez '"\U1f600"', un emoji. Les fonctions de largeur sont de la plus grande importance dans les paramètres régionaux d'Asie de l'Est : leurs valeurs diffèrent entre ces paramètres régionaux.
(Le remplacement des fonctions du système offre un certain degré d'indépendance par rapport à la plate-forme (y compris aux mises à jour du système d'exploitation), mais le remplace par une dépendance à l'égard de la version R.)

2.7.1 Symboles de débogage Par défaut, configure

ajoute un indicateur (généralement -g) aux indicateurs de compilation pour les sources C, Fortran et CXX. Cela ralentira la compilation et augmentera la taille des objets de R et des packages, il peut donc être judicieux de modifier ces indicateurs (définir 'CFLAGS', etc. dans config.site avant de configurer, ou de modifier les fichiers Makeconf et etc/Makeconf entre l'exécution de configure et fais).

Avoir des symboles de débogage disponibles est utile à la fois lors de l'exécution de R sous un débogueur (par exemple, R -d gdb) et lors de l'utilisation de désinfectants et de valgrind, tout est destiné aux experts.

Les symboles de débogage (et quelques autres) peuvent être « supprimés » lors de l'installation en utilisant

créer une bande d'installation

La qualité de cette prise en charge dépend de la plate-forme : cela fonctionne mieux sur ceux qui utilisent les binutils GNU.
Sous Linux 'x86_64', une réduction typique de la taille globale était de 92 Mo à 66 Mo. Sur macOS, les symboles de débogage ne sont pas inclus par défaut dans les fichiers .dylib et .so, il y a donc un risque négligeable différence.

2.7.2 Prise en charge d'OpenMP Par défaut,

configurez les recherches d'indicateurs appropriés12 pour la prise en charge d'OpenMP pour les compilateurs C, C++ (standard par défaut) et Fortran.

---

dix Comment préparer un tel répertoire est décrit dans le fichier src/extra/tzone/Notes dans les sources R.

11 Mais sous Windows, des problèmes ont été constatés avec les fonctions de changement de casse sur les caractères Latin-1 accentués.

12 par exemple, -fopenmp, -xopenmp ou -qopenmp. Cela inclut Clang et les compilateurs Intel et Oracle.

Seul le résultat C est actuellement utilisé pour R lui-même, et seulement si MAIN_LD/DYLIB_LD ne l'étaient pas.
spécifié. Ceci peut être annulé en spécifiant

R_OPENMP_CFLAGS

L'utilisation pour les packages a des restrictions similaires (impliquant SHLIB_LD et similaires : notez que comme le code Fortran est lié par défaut par le compilateur C (ou C++), les deux doivent prendre en charge OpenMP) et peuvent être remplacés en spécifiant certains des

SHLIB_OPENMP_CFLAGS
SHLIB_OPENMP_CXXFLAGS
SHLIB_OPENMP_FFLAGS

Les définir sur une valeur vide désactivera OpenMP pour ce compilateur (et la configuration avec --disable-openmp désactivera toute détection13 d'OpenMP). Le test de détection de configuration consiste à compiler et à lier un programme OpenMP autonome, ce qui n'est pas la même chose que compiler un objet partagé et le charger dans le programme C de l'exécutable de R. Notez que les valeurs remplacées ne sont pas testées.

2.7.3 Prise en charge de C++ C++

n'est pas utilisé par R lui-même, mais une prise en charge est fournie pour l'installation de packages avec du code C++ via les macros make définies dans le fichier etc/Makeconf (et avec des explications dans le fichier config.site) :

CXX
CXXFLAGS
CXXPICFLAGS
CXXSTD

CXX11
CXX11STD
CXX11Drapeaux
CXX11PICFLAGS

CXX14
CXX14STD
CXX14Drapeaux
CXX14PICFLAGS

CXX17
CXX17STD
CXX17Drapeaux
CXX17PICFLAGS

CXX20
CXX20STD
CXX20Drapeaux
CXX20PICFLAGS

CXX23
CXX23STD
CXX23Drapeaux
CXX23PICFLAGS

[13] Cela ne désactive pas nécessairement l'utilisation d'OpenMP – le code de configuration autorise les plates-formes sur lesquelles OpenMP est utilisé sans indicateur. Pour le compilateur flang fin 2017, le runtime Fortran utilisait toujours OpenMP.

Les macros CXX etc sont celles utilisées par défaut pour le code C++. configure tentera de définir le reste de manière appropriée, en choisissant pour CXXSTD et CXX11STD un indicateur approprié tel que -std=c++11 pour la prise en charge de C++11 (qui est requis si C++ doit être pris en charge). les valeurs déduites peuvent être remplacées dans le fichier config.site ou sur la ligne de commande configure : les valeurs fournies par l'utilisateur seront testées en compilant du code C++ 11/14/17/20/23.

Il se peut qu'il n'y ait pas d'indicateur approprié pour la prise en charge de C++14/17/20/23 avec le compilateur par défaut, auquel cas un compilateur différent pourrait être sélectionné pour CXX14/CXX17/CXX20/CXX23 avec ses indicateurs correspondants.

L'indicateur -std est pris en charge par les compilateurs GCC, clang++ et Intel. Actuellement accepté
les valeurs sont (plus quelques synonymes)

g++ : c++11 gnu+11 c++14 gnu++14 c++17 gnu++17 c++2a gnu++2a (à partir de 8) c++20 gnu++20 (à partir de 10) c++ 23 gnu++23 c++2b gnu++2b (à partir de 11)
Intel : c++11 c++14 (à partir de 16.0) c++17 (à partir de 17.0) c++20 (à partir de 2021.1) c++23 (à partir de 2022.2)

(Ceux pour LLVM clang++ sont documentés sur https://clang.llvm.org/cxx_status.html et suivent g++ : -std=c++20 est pris en charge à partir de Clang 10, -std=c++2b à partir de Clang 13. Apple Clang prend en charge -std=c++2b à partir de 13.1.6.)

Les « standards » pour g++ commençant par « gnu » activent les « extensions GNU » : il est difficile de les retrouver.

Pour l'utilisation de C++11 et versions ultérieures dans les packages R, consultez le manuel « Écriture d'extensions R ». Avant R 3.6.0, la norme C++ par défaut était celle du compilateur utilisé : actuellement, il s'agit de C++17 (si disponible) : cela peut être remplacé en définissant 'CXXSTD' lorsque R est configuré.

https://en.cppreference.com/w/cpp/compiler_support indique quelles versions des compilateurs courants prennent en charge (en partie) quelles normes C++. GCC 5 était la version minimale avec une prise en charge suffisante de C++14. GCC a introduit progressivement le support de C++17, mais la version 7 devrait suffire.

2.7.4 Normes C

La compilation de R nécessite C99 ou version ultérieure : C11 et C17 sont des mises à jour mineures, mais la mise à jour substantielle prévue pour « C23 » (maintenant attendue en 2024) sera également prise en charge.

À partir de R 4.3.0, les packages sont pris en charge pour indiquer leur version C préférée. Les macros CC17, C17FLAGS, CC23 et C23FLAGS peuvent être définies dans config.site (il y a des exemples là-bas).
Ceux pour C17 doivent prendre en charge C17 ou version antérieure et ne pas autoriser les ajouts C23, donc par exemple bool, true et false peuvent être utilisés comme identifiants. Ceux pour C23 devraient prendre en charge de nouveaux types tels que bool.

Certains compilateurs mettent en garde avec enthousiasme contre les prototypes. Pour la plupart, omettre les prototypes -Wstrict dans C17FLAGS suffit. Cependant, les versions 15 et ultérieures de LLVM clang et 14.0.3 et ultérieures d'Apple clang avertissent par défaut dans tous les modes si -Wall ou -pedantic est utilisé et peuvent nécessiter -Wno-strict-prototypes.

2.7.5 Optimisation du temps de liaison L'utilisation de

l'optimisation du temps de liaison (LTO) est prise en charge si la chaîne d'outils la prend en charge : configurez avec l'indicateur --enable-lto. Lorsque LTO est activé, il est utilisé pour le code compilé dans les packages complémentaires, sauf si l'indicateur --enable-lto=R est utilisé[14] .

Le principal avantage constaté jusqu'à présent avec LTO a été la détection de bogues de longue date dans la manière dont les packages transmettent les arguments au code compilé et entre les unités de compilation. L'analyse comparative réalisée en 2020 avec gcc/gfortran 10 a montré des gains de quelques pour cent en termes d'augmentation des performances et de réduction

[14] Ensuite, les packages recommandés installés dans le cadre de l'installation de R utilisent LTO, mais pas les packages installés ultérieurement.

en taille installée pour les builds sans symboles de débogage, mais des réductions de taille importantes pour certains packages15 avec symboles de débogage. (On dit que les gains de performances et de taille sont le plus souvent observés dans les versions C++ complexes.)

Il est souvent difficile de savoir si les chaînes d'outils prennent en charge LTO : tous les compilateurs C, le compilateur Fortran16 et l'éditeur de liens doivent le prendre en charge, et le prendre en charge par le même mécanisme (donc le mélange des familles de compilateurs peut ne pas fonctionner et un éditeur de liens autre que celui par défaut peut être nécessaire).). Il est soutenu par les projets GCC et LLVM depuis quelques années avec des implémentations divergentes.

La prise en charge de LTO a été ajoutée en 2011 pour GCC 4.5 sous Linux mais était peu utilisée avant 2019 : la prise en charge du compilateur s'est régulièrement améliorée au fil de ces années et --enable-lto=R est aujourd'hui utilisé pour certaines vérifications CRAN de routine .

Malheureusement, --enable-lto peut être accepté mais ne fait rien d'utile en silence si une partie de la chaîne d'outils ne prend pas en charge LTO : c'est moins courant qu'avant.

Diverses macros peuvent être définies dans le fichier config.site pour personnaliser la façon dont LTO est utilisé. Si le compilateur Fortran n'est pas de la même famille que les compilateurs C/C++, définissez la macro 'LTO_FC' (probablement vide). La macro 'LTO_LD' peut être utilisée pour sélectionner un autre éditeur de liens si cela s'avère nécessaire.

2.7.5.1 LTO avec GCC

Cela a été testé sous Linux avec gcc/gfortran 8 et versions ultérieures : cela nécessitait un réglage (par exemple dans config.site)

```
AR=gcc-ar
RANLIB=gcc-ranlib
```

Pour les compilateurs non système ou si ces wrappers n'ont pas été installés, vous aurez peut-être besoin de quelque chose comme

```
AR="ar --plugin=/path/to/liblto_plugin.so"
RANLIB="ranlib --plugin=/path/to/liblto_plugin.so"
```

amd NM devra peut-être être réglé de manière analogue. (Si vous utilisez une version compatible LTO pour vérifier les packages, définissez la variable d'environnement UserNM17 sur « gcc-nm ».)

Avec GCC 5 et versions ultérieures, il est possible de paralléliser des parties du processus de liaison LTO : définissez la macro make 'LTO' sur quelque chose comme 'LTO=-flto=8' (pour utiliser 8 threads), par exemple dans le fichier config.site.

Dans certaines circonstances et pour quelques packages, les options PIC ont dû être remplacées sous Linux avec GCC 9 et versions ultérieures : par exemple, utilisez dans config.site :

```
CPICFLAGS=-fPIC
CXXPICFLAGS=-fPIC
CXX11PICFLAGS=-fPIC
CXX14PICFLAGS=-fPIC
CXX17PICFLAGS=-fPIC
CXX20PICFLAGS=-fPIC
FPICFLAGS=-fPIC
```

Nous suggérons de ne les utiliser que si le problème est rencontré (il n'a pas été vu sur CRAN avec GCC 10 au moment de la rédaction).

Notez que R peut devoir être recompilé après même une mise à jour mineure du compilateur (par exemple de 10.1 à 10.2), mais cela peut ne pas être clair à partir de messages confus du compilateur.

---

15 Une installation CRAN complète réduite de 50 à 35Go. bien

16 qu'il existe la possibilité d'exclure Fortran, cela ne présente pas certains avantages. pas NM

17 comme nous l'avons constaté, il faut outrepasser cela.

2.7.5.2 LTO avec LLVM

LLVM prend en charge un autre type de LTO appelé « Thin LTO », ainsi qu'une implémentation similaire à GCC, parfois appelée « Full LTO ». (Voir https://clang.llvm.org/docs/ThinLTO.html.)
Actuellement, le seul compilateur LLVM pertinent pour R est clang pour lequel cela peut être sélectionné en définissant la macro 'LTO=-flto=thin'. LLVM a

```
AR=llvm-année
RANLIB=llvm-ranlib
```

(mais ce n'est pas le cas de macOS, et ceux-ci ne sont pas nécessaires là-bas). Lorsque l'éditeur de liens prend en charge un backend parallèle pour Thin LTO, cela peut être spécifié via la macro « LTO_LD » : voir l'URL ci-dessus pour les paramètres par éditeur de liens et d'autres optimisations de liaison.)

Par exemple, sur macOS, on peut utiliser

```
LTO=-flto=mince
LTO_FC=
LTO_LD=-Wl,-mllvm,-threads=4
```

pour utiliser Thin LTO avec 4 threads pour le code C/C++, mais ignorez LTO pour le code Fortran compilé avec gfortran.

On dit qu'il est particulièrement avantageux d'utiliser -O3 pour le clang en conjonction avec LTO.

2.7.5.3 LTO pour la vérification des paquets

LTO compile efficacement tout le code source d'un package en tant qu'unité de compilation unique et permet ainsi au compilateur (avec suffisamment d'indicateurs de diagnostic tels que -Wall) de vérifier la cohérence entre des unités de compilation normalement distinctes.

Avec gcc/gfortran 9.x et versions ultérieures18, LTO signalera les incohérences dans les appels aux sous-routines/fonctions Fortran, à la fois entre les fichiers sources Fortran et entre Fortran et C/C++. gfortran 8.4, 9.2 et versions ultérieures peuvent aider à les comprendre en extrayant les prototypes C des fichiers sources Fortran avec l'option -fc-prototypes-external, par exemple que (au moment de la rédaction) Fortran LOGICAL correspond à int_least32_t * en C.

Sur certains systèmes, il est possible de créer le support BLAS, LINPACK et LAPACK en tant que bibliothèques statiques contenant du code de niveau intermédiaire que LTO compilera pour tous les objets liés à ces bibliothèques, en configurant R avec --enable-lto=check. Cela vérifie la cohérence des appels à BLAS/LINPACK/LAPACK dans tous les packages installés à l'aide de la build. NB : comme son nom l'indique, cette option est destinée uniquement à vérifier l'installation de R et des packages : elle inclut ces routines de la bibliothèque (celles appelées directement et toutes dont elles dépendent) dans chaque package. Il est peu probable que cela fonctionne conjointement avec des options autres que celles par défaut pour BLAS et LAPACK, et la « liaison » avec ces bibliothèques peut être très lente.

2.8 Tester une installation

Les tests complets post-installation ne sont possibles que si les fichiers de test ont été installés avec

```
faire des tests d'installation
```

qui remplit un répertoire de tests dans l'installation.

Si cela a été fait, deux itinéraires de test sont disponibles. La première est de déménager à la maison répertoire de l'installation de R (tel que donné par R RHOME ou depuis R comme R.home()) et exécutez

```
tests de CD
## suivi de l'un des ../bin/R CMD
 make check ../bin/R CMD make
 check-devel
```

18 probablement aussi 8.4 et versions ultérieures.

../bin/R CMD fait tout vérifier

et d'autres cibles utiles sont test-BasePackages et test-Recommended pour exécuter respectivement des tests des packages standard et recommandés (s'ils sont installés).

Cela réexécute tous les tests pertinents pour le R installé (y compris par exemple le code dans les vignettes du package), mais pas par exemple ceux vérifiant l'exemple de code dans les manuels ni créant la bibliothèque Rmath autonome. Cela peut occasionnellement être utile lorsque l'environnement d'exploitation a été modifié, par exemple par des mises à jour du système d'exploitation ou en remplaçant le BLAS (voir Section A.3.1.4 [Shared BLAS], page 48).

Une vérification parallèle des packages peut être possible : définissez la variable d'environnement TEST_MC_CORES sur le nombre maximum de processus à exécuter en parallèle. Cela affecte à la fois la vérification des exemples de packages (partie de make check) et des sources du package (partie de make check-devel et make check-recommended). Cela nécessite une commande make qui prend en charge l'option make -jn : la plupart le font.

Alternativement, le R installé peut être exécuté, de préférence avec --vanilla. Alors

```
pdf("tests.pdf") ## facultatif, mais empêche les fenêtres graphiques de clignoter Sys.setenv(LC_COLLATE
= "C", LC_TIME = "C", LANGUAGE = "en") tools::testInstalledBasic("both")
outils : :testInstalledPackages(scope = "base")
outils::testInstalledPackages(scope = "recommended")
```

exécute les tests de base puis tous les tests sur les packages standards et recommandés. Ces tests peuvent être exécutés de n'importe où : les tests de base écrivent leurs résultats dans le dossier tests du répertoire personnel de R et exécutent moins de tests que la première approche : en particulier, ils ne testent pas les éléments qui nécessitent un accès à Internet – qui peuvent être testés par des

```
outils ::testInstalledBasic("internet")
```

Il est possible de tester les packages installés (mais pas leurs tests spécifiques au package) par testInstalledPackages même si make install-tests n'a pas été exécuté. Les sorties sont écrites dans le répertoire courant sauf si un autre est spécifié par outDir.

Notez que les résultats peuvent dépendre de la langue définie pour les heures et les messages : pour un maximum similarité avec les résultats de référence que vous voudrez peut-être essayer de définir (avant de démarrer la session R)

```
LANGUE=fr
```

et utilisez une locale UTF-8 ou Latin-1.

3 Installer R sous Windows

[Le reste de ce paragraphe n'est pertinent qu'après la publication.] Le répertoire bin/windows d'un site CRAN contient des binaires pour une distribution de base et un grand nombre de packages complémentaires de CRAN pour fonctionner sur Windows 64 bits.

R est mieux testé sur les versions actuelles de Windows 10 et Windows Server 2022 avec UTF-8 comme codage de jeu de caractères. Il fonctionne également sur Windows 11. Il fonctionne sur les anciennes versions de Windows, mais normalement avec un autre codage de jeu de caractères et peut nécessiter l'installation manuelle de Universal C Runtime (UCRT).

Votre système de fichiers doit autoriser les noms de fichiers longs (comme c'est probablement le cas, sauf peut-être pour certains systèmes montés en réseau). S'il ne prend pas également en charge la conversion en équivalents de noms courts (c'est-à-dire les noms DOS 8.3), alors R doit être installé dans un chemin qui ne contient pas d'espaces.

L'installation s'effectue via le programme d'installation R-4.3.1-win.exe. Double-cliquez simplement sur l'icône et suivez les instructions. Vous pouvez désinstaller R à partir du Panneau de configuration.

Il vous sera demandé de choisir une langue pour l'installation : ce choix s'applique à la fois à l'installation et la désinstallation mais pas l'exécution de R lui-même.

Consultez la FAQ R Windows (https://CRAN.R-project.org/bin/windows/base/rw-FAQ.html) pour plus de détails sur le programme d'installation binaire et pour des informations sur l'utilisation sur les anciens systèmes Windows.

3.1 Construction à partir des sources Il est

possible d'utiliser d'autres chaînes d'outils 64 bits (y compris « MSYS2 ») avec le support UCRT pour construire R, mais ce manuel ne documente que celles utilisées pour les distributions binaires de R 4.2.x et versions ultérieures. Lors de l'utilisation d'autres chaînes d'outils, les makefiles de R et les packages devront peut-être être adaptés.

3.1.1 Le jeu d'outils Windows

La distribution binaire de R est actuellement construite avec les outils de Rtools43 pour Windows (https://CRAN.R-project.org/bin/windows/Rtools/rtools43/rtools.html). Voir Building R et les packages (https://CRAN.R-project.org/bin/windows/base/howto-R-devel.html) pour en savoir plus
des détails sur la façon de l'utiliser.

L'ensemble d'outils comprend des compilateurs (GCC version 12.2.0 avec des correctifs supplémentaires sélectionnés) et des bibliothèques d'exécution du projet « MinGW-w64 » (http://mingw-w64.org/) ainsi qu'un certain nombre de bibliothèques statiques précompilées et en-têtes utilisés par les packages R et R, compilés par 'MXE' (https:// mxe.cc/) (environnement cross M, avec mises à jour maintenues par Tomas Kalibera). L'ensemble d'outils comprend également les outils de construction du projet « MSYS2 » (https://www.msys2.org/). Des outils de construction supplémentaires fournis par « MSYS2 » peuvent être installés via un gestionnaire de packages (« pacman »).

Les ensembles d'outils utilisés pour Windows 64 bits de 2008 à 2022 étaient basés sur MinGW-w64. L'aide de Yu Gong à une étape cruciale du portage de R sur MinGW-w64 est grandement appréciée, ainsi que l'aide de Kai Tietz, le développeur principal du projet MinGW-w64 et de Martin Storsjo.

3.1.2 LATEX Les

paquets de construction R et de vérification nécessitent une distribution de LATEX installée, avec le répertoire contenant pdflatex sur le chemin.

La distribution 'MiKTeX' (https://miktex.org/) de LATEX est celle utilisée sur CRAN. Cela peut être configuré pour installer des packages supplémentaires « à la volée » (sans demander), ce qui constitue la manière la plus simple de l'utiliser. La version « de base » de « MiKTeX » devra ajouter quelques packages.1 Dans tous les cas, assurez-vous que le package inconsolata est installé — vous pouvez vérifier auprès du gestionnaire de packages « MiKTeX ».

[1] Il y a des rapports de défauts de segmentation lorsque 'MiKTeX' installe des packages supplémentaires lors de la création de NEWS.pdf : la réexécution de make semble résoudre ce problème.

Il est également possible d'utiliser la distribution TeX Live depuis https://www.tug.org/texlive/ . (Le package CRAN tinytex (https://CRAN.R-project.org/package=tinytex) peut installer et gérer un sous-ensemble de TeX Live.)

3.2 Vérification de la construction

Vous pouvez tester une build en exécutant

```
faire un chèque
```

Les forfaits recommandés peuvent être vérifiés par

```
faire un chèque-recommandé
```

D'autres niveaux de contrôle sont

```
faire un contrôle-développement
```

pour une vérification plus approfondie de la fonctionnalité R, et

```
faire tout vérifier
```

pour check-devel et check-recommended.

Si un test échoue, il y aura presque toujours un fichier .Rout.fail dans le répertoire en cours de vérification (souvent des tests/Exemples ou tests) : examinez le fichier pour identifier le problème.

La vérification parallèle des sources des packages (qui fait partie de make check-devel et make check-recommended) est possible : voir la variable d'environnement TEST_MC_CORES pour le nombre maximum de processus à exécuter en parallèle.

3.3 Tester une installation

Le programme d'installation de Windows contient un ensemble de fichiers de test utilisés lors de la création de R.

Cet ensemble d'outils n'est pas nécessaire pour exécuter ces tests, mais une analyse plus complète des erreurs sera nécessaire. être donné si diff est dans le chemin.

Lancez Rgui ou Rterm (de préférence), de préférence avec --vanilla. Ensuite, exécutez Sys.setenv(LC_COLLATE

```
= "C", LC_TIME="C", LANGUAGE = "en") tools::testInstalledBasic("both")
tools::testInstalledPackages(scope = "base")
tools::testInstalledPackages(scope = "recommandé")
```

exécute les tests de base puis tous les tests sur les packages standards et recommandés. Ces tests peuvent être exécutés de n'importe où : testInstalledBasic écrit les résultats dans le dossier tests du répertoire personnel R (tel que donné par R.home()) et testInstalledPackages sous le répertoire courant, sauf si un autre est spécifié par outDir.

Pour que le dossier tests soit accessible en écriture, il faut normalement installer R dans un répertoire autre que le répertoire par défaut C:\Program Files. Le programme d'installation permet également d'installer R sans privilèges d'administrateur, voir la FAQ R Windows (https://CRAN.R-project.org/bin/windows/base/rw-FAQ.html) pour plus de détails.

Les résultats de l'exemple (md5sums) lors du test des outils peuvent différer de la sortie de référence car certains fichiers sont installés avec les fins de ligne CRLF de Windows. Attendez-vous également à des différences dans reg-plot-latin1.pdf.

On peut également exécuter des tests à partir du shell du jeu d'outils (par exemple bash) de la même manière qu'une installation de type Unix. Accédez au répertoire personnel de l'installation de R (tel qu'indiqué par R RHOME ou depuis R en tant que R.home()) et exécutez

```
tests de CD
## suivi de l'un des ../bin/R CMD
 make check ../bin/R CMD make check-
 devel
```

../bin/R CMD fait tout vérifier

N'oubliez pas que LATEX doit être sur le chemin.

4 Installer R sous macOS

[Le reste de ce paragraphe n'est pertinent qu'après la publication.] La première page d'un site CRAN contient un lien « Télécharger R pour (Mac) OS X » qui vous amène à une nouvelle page. Deux fichiers sont proposés au téléchargement, R-4.3.1-arm64.pkg et R-4.3.1.pkg. Les deux sont pour macOS 11 ou version ultérieure (Big Sur, .).
Monterey, Ventura, . .

La première exécution est destinée aux Mac « Apple Silicon » (alias « M1 » ou « M2 »), la seconde aux Mac plus anciens dotés d'un processeur « x86_64 » (Intel).

Le package R-4.3.1.pkg doit également être installé sur les processeurs « Apple Silicon » utilisant l'émulation « Rosetta »1, , mais la version native est préférée. Il est un peu plus rapide (et considérablement pour certaines tâches) mais peut donner des résultats numériques différents de ceux des plates-formes « x86_64 » les plus courantes (sous Windows, Linux, . . ainsi que macOS) · car le matériel ARM manque de flottement à précision étendue. opérations ponctuelles.

Il est important que si vous utilisez un package d'installation binaire, votre système d'exploitation soit entièrement mis à jour : regardez dans « Mise à jour du logiciel » dans « Préférences Système » pour être sûr.

Pour l'installer, double-cliquez simplement sur l'icône du fichier que vous avez téléchargé. À l'étape « Type d'installation », notez l'option « Personnaliser ». Cela montre actuellement quatre composants : tout le monde aura besoin du composant « R Framework » : les composants restants sont facultatifs. (Le composant 'Tcl/Tk' est nécessaire pour utiliser le package tcltk. Le composant 'Texinfo' n'est nécessaire que pour ceux qui installent les packages sources ou R à partir de ses sources.)

Remarque pour les utilisateurs de Ventura : l'installation à partir du dossier Téléchargements peut ne pas être autorisée ou nécessiter une autorisation supplémentaire. Nous vous suggérons donc de télécharger ailleurs, par exemple sur votre bureau ou votre dossier personnel.

Ce sont des packages Apple Installer. Si vous rencontrez un problème lors de l'installation, veuillez vérifier le journal de l'installateur en cliquant sur le menu « Fenêtre » et l'élément « Journal de l'installateur ». La sortie complète (sélectionnez « Afficher tout le journal ») est utile pour détecter les problèmes. Notez que le programme d'installation est suffisamment intelligent pour essayer de mettre à niveau la dernière version installée de l'application sur laquelle vous l'avez installée (qui n'est peut-être pas celle que vous souhaitez cette fois-ci...).

Diverses parties de la version nécessitent l'installation de XQuartz : voir https://www.xquartz.org/releases/ .
2 Ceux-ci incluent le package tcltk et le périphérique graphique X11 : tenter de les utiliser sans XQuartz vous le rappellera. Ceci est également nécessaire pour certaines versions d'appareils basés sur la cairographie (qui ne sont pas souvent utilisés sur macOS) tels que png(type = "cairo") et svg() et certains packages tiers (par exemple rgl (https:// CRAN.R-project.org/ package=rgl)).

Si vous mettez à jour votre version macOS, vous devez réinstaller R (et peut-être XQuartz) : le
Le programme d'installation peut adapter l'installation à la version actuelle du système d'exploitation.

Les installateurs pour R-patched et R-devel sont généralement disponibles sur https://mac.R-project. org. (Certains de ces packages peuvent être non signés/non notariés : pour les installer Contrôle/clic droit/deux doigts, sélectionnez « Ouvrir avec » et « Installateur ».)

Pour construire R à partir des sources, voir Section C.3 [macOS], page 60.

4.1 Exécuter R sous macOS Il existe deux manières

d'exécuter R sur macOS à partir d'une distribution binaire CRAN .

Il existe une console GUI normalement installée avec l'icône R dans /Applications que vous pouvez exécuter en double-cliquant (par exemple depuis Launchpad ou Finder). (Si vous ne le trouvez pas, il a peut-être été installé ailleurs, alors essayez de le rechercher dans Spotlight.) Ceci est généralement appelé R.app pour le distinguer de la ligne de commande R : son manuel d'utilisation fait actuellement partie du

---

1 Il peut vous être demandé d'installer Rosetta lors de la première utilisation – https://support.apple.com/en-us/HT211861 – ce qui peut nécessiter des privilèges d'administrateur.

2 Au moment de la rédaction, version 2.8.5 ou ultérieure.

FAQ macOS sur https://cran.r-project.org/bin/macosx/RMacOSX-FAQ.html et peut être consultée à partir du menu « Aide » de R.app.

Vous pouvez exécuter R et Rscript en ligne de commande à partir d'un Terminal[3] afin qu'ils puissent être saisis sous forme de commandes comme sur n'importe quel autre système Unix : voir le chapitre suivant de ce manuel. Il existe quelques petites différences qui peuvent surprendre les utilisateurs de R sur d'autres plateformes, notamment l'emplacement par défaut du répertoire de la bibliothèque personnelle (sous ~/Library/R, par exemple ~/Library/R/x86_64/4.2/library), et les avertissements, les messages et autres sorties vers stderr sont mis en évidence en gras.

Ceux qui utilisent le shell zsh (la valeur par défaut pour les nouveaux comptes d'utilisateurs à partir de Catalina) pourraient trouver la commande R masquée par le r intégré de zsh (qui rappelle les commandes). On peut utiliser un chemin complet vers R dans un alias, ou ajouter désactiver r à ~/.zshrc.

Si vous avez installé les deux packages d'installation sur un Mac arm64, le dernier installé sera utilisé.

Il a été signalé que l'exécution de R.app peut échouer si aucune préférence n'est stockée, donc si elle échoue lors du tout premier lancement, réessayez (la première tentative stockera certaines préférences).

Les utilisateurs de R.app doivent connaître la fonctionnalité « App Nap » (https://developer.apple.com/library/archive/releasenotes/MacOSX/WhatsNewInOSX/Articles/MacOSX10_9. html), ce qui peut faire en sorte que les tâches R semblent s'exécuter très lentement lorsqu'elles ne produisent pas de sortie dans la console. Voici des moyens de l'éviter :

- Assurez-vous que la console est complètement visible (ou au moins l'indicateur d'activité en haut le coin droit est visible).
- Dans un terminal, exécutez

  les valeurs par défaut écrivent org.R-project.R NSAppSleepDisabled -bool YES (voir https://developer.apple.com/library/archive/releasenotes/MacOSX/WhatsNewInOSX/Articles/MacOSX10_9.html).

L'utilisation du périphérique graphique X11 ou des versions basées sur X11 de View() et edit() pour les trames de données et les matrices (ces dernières sont la valeur par défaut pour la ligne de commande R mais pas pour R.app) nécessite XQuartz (https://www. xquartz.org/) à installer.

Dans certaines circonstances plutôt nébuleuses, des messages ont été reçus de fontconfig concernant des fichiers de configuration manquants/illisibles lors de l'utilisation de périphériques basés sur le Caire, en particulier X11(type = "cairo"). Avec XQuartz installé, il existe deux zones fontconfig de différentes versions et cela peut aider à définir

```
setenv FONTCONFIG_PATH /opt/X11/lib/X11/fontconfig
```

Un autre symptôme est que les polices italiques/obliques sont remplacées par des polices verticales.

4.2 La désinstallation sous macOS R pour macOS se

compose de deux parties : l'interface graphique (R.app) et le framework R. La désinstallation est aussi simple que de supprimer ces dossiers (par exemple en les faisant glisser vers la Corbeille, également appelée Corbeille). L'installation typique installera l'interface graphique dans le dossier /Applications/R.app et le framework R dans le dossier /Library/Frameworks/R.framework. Les liens vers R et Rscript dans /usr/local/bin doivent également être supprimés.

Si vous souhaitez vous débarrasser plus complètement de R à l'aide d'un terminal, exécutez simplement :

```
sudo rm -Rf /Bibliothèque/Frameworks/R.framework /Applications/R.app \ /usr/local/bin/R /usr/local/bin/
    Rscript
```

[3] Le programme d'installation place des liens vers R et Rscript dans /usr/local/bin. Si celles-ci sont manquantes ou si elles ne figurent pas sur votre chemin, vous pouvez exécuter directement les copies dans /Library/Frameworks/R.framework/Resources/bin ou les lier vous-même quelque part sur votre chemin.

L'installation comprend jusqu'à quatre packages Apple :4 pour la version Intel, org.R-project.x86_64.R.fw.pkg, org.R-project.x86_64.R.GUI.pkg, org.r-project. x86_64.tcltk et org.r-project.x86_64.texinfo. Vous pouvez utiliser sudo pkgutil --forget si vous souhaitez que le programme d'installation Apple oublie le package sans supprimer ses fichiers (utile pour le framework R lors de l'installation de plusieurs versions R en parallèle), ou après avoir supprimé les fichiers.

NB : les noms des packages sont sensibles à la casse et le domaine R est nommé de manière incohérente.

La désinstallation des composants Tcl/Tk et Texinfo (qui sont installés sous /opt/R/x86_64 sur une build 'x86_64' et /opt/R/arm64 pour une build 'arm64') n'est pas aussi simple. Vous pouvez lister les fichiers qu'ils ont installés dans un terminal, par exemple

```
pkgutil --files org.r-project.x86_64.tcltk pkgutil --files org.r-
project.x86_64.texinfo
```

(Pour la build 'Apple Silicon', remplacez x86_64 par arm64.) Ce sont des chemins relatifs à /, la racine du système de fichiers.

Si vous ne compilez pas R ni n'installez de packages à partir des sources, vous pouvez supprimer tout /opt/R/x86_64 ou /opt/R/arm64.

4.3 Versions multiples Le programme

d'installation supprimera toute version précédente5 du framework R qu'il trouve installée.
Cela peut être évité en utilisant pkgutil --forget (voir la section précédente). Cependant, notez que différentes versions sont installées sous /Library/Frameworks/R.framework/Versions comme 4.4 (ou 4.4-arm64), 4.3 et ainsi de suite, il n'est donc pas possible d'installer différentes versions '4.x.y' pour le même «x» et type de processeur.

R.app exécutera toujours la version « actuelle », c'est-à-dire la dernière version installée.

[4] Au moment de la rédaction : utilisez pkgutil --pkgs | grep -i org.r-project à vérifier.

[5] Plus précisément, du package Apple du même nom : cela signifie que les versions Intel et ARM peuvent être installées ensemble.

5 Courir R

Comment démarrer R et quelles options de ligne de commande sont disponibles sont abordés dans la section « Invoquer R » dans Une introduction à R.

Vous devez vous assurer que le shell a défini des limites de ressources adéquates : R s'attend à une taille de pile d'au moins 8 Mo et à pouvoir ouvrir au moins 256 descripteurs de fichiers. (Tout système d'exploitation moderne devrait avoir des limites par défaut au moins aussi grandes que celles-ci, mais apparemment NetBSD ne peut pas le faire. Utilisez la commande shell ulimit (sh/bash) ou limit (csh/tcsh) pour vérifier.) Pour certains compilateurs1 et packages, une pile plus grande la taille était nécessaire : 20-25 Mo ont suffi à ce jour.

R utilise un certain nombre de variables d'environnement, dont les valeurs par défaut sont définies dans le fichier R_HOME/etc/Renviron (il n'y en a aucune définie par défaut sous Windows et donc aucun fichier de ce type). Ceux-ci sont définis au moment de la configuration et vous ne souhaitez normalement pas les modifier – une exception possible est R_PAPERSIZE (voir Section B.3.1 [Définition du format de papier], page 52). Le format du papier sera déduit de la catégorie locale 'LC_PAPER' si elle existe et si R_PAPERSIZE n'est pas défini, et cela produira normalement le bon choix entre 'a4' et 'lettre' sur les Unix-alikes modernes (mais peut toujours être remplacé en définissant R_PAPERSIZE).

Diverses variables d'environnement peuvent être définies pour déterminer où R crée son répertoire temporaire par session. Les variables d'environnement TMPDIR, TMP et TEMP sont recherchées tour à tour et la première qui est définie et pointe vers une zone inscriptible est utilisée. Si aucun ne le fait, la valeur par défaut finale est /tmp sur les systèmes Unix-alikes et la valeur de R_USER sous Windows. Le chemin doit être un chemin absolu ne contenant pas d'espaces2 (et il est préférable d'éviter les caractères non alphanumériques tels que + ou les guillemets).

Certains systèmes de type Unix sont configurés pour supprimer périodiquement des fichiers et des répertoires de /tmp, par exemple par une tâche cron exécutant tmpwatch. Définissez TMPDIR sur un autre répertoire avant de démarrer des tâches de longue durée sur un tel système.

Notez que TMPDIR sera utilisé pour exécuter les scripts de configuration lors de l'installation des packages, donc si /tmp a été monté en tant que « noexec », TMPDIR doit être défini sur un répertoire à partir duquel l'exécution est autorisée.

---

[1] Y compris GCC 9 sur Linux.

[2] Sous Windows, un chemin contenant des espaces sera remplacé par la version « chemin court » si celle-ci ne contient pas d'espaces.

6 forfaits complémentaires

Il est utile d'utiliser la terminologie correcte. Un package est chargé depuis une bibliothèque par la fonction library(). Ainsi, une bibliothèque est un répertoire contenant les packages installés ; la bibliothèque principale est R_HOME/library, mais d'autres peuvent être utilisées, par exemple en définissant la variable d'environnement R_ LIBS ou en utilisant la fonction R .libPaths(). Pour éviter toute confusion, vous verrez souvent un répertoire de bibliothèques appelé « arborescence de bibliothèques ».

6.1 Packages par défaut L'ensemble des

packages chargés au démarrage est par défaut

```
> getOption("defaultPackages") [1] "ensembles de
 données" "utils"                "grDevices" "graphiques" "statistiques"                "méthodes"
```

(plus, bien sûr, la base) et cela peut être modifié en définissant l'option dans le code de démarrage (par exemple dans ~/.Rprofile). Il est initialement défini sur la valeur de la variable d'environnement R_DEFAULT_PACKAGES si elle est définie (sous forme de liste séparée par des virgules). Le paramètre R_DEFAULT_PACKAGES=NULL garantit que seul le package
la base est chargée.

Changer l'ensemble des packages par défaut est normalement utilisé pour réduire l'ensemble de vitesse lors de l'écriture de scripts : en particulier, ne pas utiliser de méthodes réduira le temps de démarrage d'un facteur allant jusqu'à deux.
Mais il peut également être utilisé pour personnaliser R, par exemple pour une utilisation en classe. Rscript vérifie également la variable d'environnement R_SCRIPT_DEFAULT_PACKAGES ; si défini, cela a priorité sur R_DEFAULT_PACKAGES.

6.2 Gestion des bibliothèques

Les packages R sont installés dans des bibliothèques, qui sont des répertoires du système de fichiers contenant un sous-répertoire pour chaque package installé.

R est livré avec une seule bibliothèque, R_HOME/library qui est la valeur de l'objet R '.Library' contenant les packages standard et recommandés1. Les sites et les utilisateurs peuvent en créer d'autres et les utiliser (ou non) dans une session R. Au niveau le plus bas, « .libPaths() » peut être utilisé pour ajouter des chemins à la collection de bibliothèques ou pour signaler la collection actuelle.

R utilisera automatiquement une bibliothèque spécifique au site R_HOME/site-library si elle existe (ce n'est pas le cas dans une installation Vanilla R). Cet emplacement peut être remplacé en définissant 2 '.Library.site' dans R_HOME/etc/Rprofile.site, ou (non recommandé) en définissant la variable d'environnement R_ LIBS_SITE.

Les utilisateurs peuvent disposer d'une ou plusieurs bibliothèques, normalement spécifiées par la variable d'environnement R_ LIBS_USER. Celui-ci a une valeur par défaut (pour le voir, utilisez 'Sys.getenv("R_LIBS_USER")' dans une session R), mais qui n'est utilisée que si le répertoire correspondant existe réellement (ce qui ne sera pas le cas par défaut).

R_LIBS_USER et R_LIBS_SITE peuvent spécifier plusieurs chemins de bibliothèque, séparés par des deux-points (points-virgules sous Windows).

6.3 Installation des packages Les packages

peuvent être distribués sous forme source ou sous forme binaire compilée. L'installation de packages sources contenant du code C/C++/ Fortran nécessite l'installation des compilateurs et des outils associés.
Les packages binaires sont spécifiques à la plate-forme et ne nécessitent généralement aucun outil spécial pour être installés, mais consultez la documentation de votre plate-forme pour plus de détails.

---

1 à moins qu'ils n'aient été exclus dans la

2 construction. sa liaison est verrouillée une fois les fichiers de démarrage lus, les utilisateurs ne peuvent donc pas la modifier facilement. Voir ?.libPaths pour savoir comment utiliser la nouvelle valeur.

Notez que vous devrez peut-être spécifier implicitement ou explicitement la bibliothèque à laquelle le package est à installer. Ce n'est bien sûr un problème que si vous possédez plusieurs bibliothèques.

Assurez-vous que la variable d'environnement TMPDIR n'est pas définie (et que /tmp existe et peut être écrit et exécuté à partir de) ou qu'il s'agit du chemin absolu vers un répertoire temporaire valide, ne contenant pas les espaces.

Pour la plupart des utilisateurs, il suffit d'appeler « install.packages(pkgname) » ou son équivalent GUI si l'intention est d'installer un package CRAN et qu'un accès Internet est disponible.3 Sur la plupart des systèmes, « install.packages() » permettra d'installer des packages. sélectionné dans une zone de liste (généralement avec des milliers d'éléments).

Pour installer des packages à partir des sources sur un système Unix, utilisez dans un terminal R CMD INSTALL -l /path/to/library pkg1 pkg2 ...

La partie '-l /path/to/library' peut être omise, auquel cas la première bibliothèque d'une session R normale est utilisée (celle affichée par .libPaths()[1]).

Un certain nombre d'options sont disponibles : utilisez R CMD INSTALL --help pour voir la liste actuelle.

Alternativement, les packages peuvent être téléchargés et installés à partir de R. Choisissez d'abord votre miroir CRAN le plus proche à l'aide de ChooseCRANmirror(). Ensuite, téléchargez et installez les packages pkg1 et pkg2 en > install.packages(c("pkg1", "pkg2"))

Les dépendances essentielles des packages spécifiés seront également récupérées. Sauf si la bibliothèque est spécifiée (argument lib), la première bibliothèque dans le chemin de recherche de la bibliothèque est utilisée : si elle n'est pas accessible en écriture, R demandera à l'utilisateur (dans une session interactive) si la bibliothèque personnelle par défaut doit être créée, et si elle est autorisée à la créer. y installera les packages.

Si vous souhaitez récupérer un package et tous ceux dont il dépend (de quelque manière que ce soit) qui ne sont pas déjà installés, utilisez par exemple

> install.packages("Rcmdr", dependencies = TRUE) install.packages peut

installer un package source à partir d'un fichier .tar.gz local (ou d'une URL vers un tel fichier) en définissant l'argument repos sur NULL : celui-ci sera sélectionné automatiquement si le nom donné est un seul fichier .tar.gz.

install.packages peut rechercher dans plusieurs référentiels, spécifiés comme vecteur de caractères par l'argument repos : ceux-ci peuvent inclure un miroir CRAN , Bioconductor, R-forge, rforge.net, des archives locales, des fichiers locaux, . .). La fonction setRepositories() peut sélectionner parmi les référentiels dont l'installation R est consciente.

Ce qui laisse parfois perplexe les utilisateurs, c'est que install.packages() peut signaler qu'un Le package qui, selon eux, devrait être disponible, n'est pas trouvé. Quelques raisons possibles :

- Le package, tel que grid ou tcltk, fait partie de R lui-même et n'est pas disponible autrement.
- Le package n'est pas dans les dépôts disponibles, vérifiez donc lesquels ont été sélectionnés par getOption("repos")
- Le package est disponible, mais pas pour la version actuelle de R ni pour le type d'OS (Unix/Windows). Pour récupérer les informations sur les versions disponibles du package pkg, utilisez

  av <- available.packages(filters=list()) av[av[, "Package"] == pkg,]

  dans votre session R, et regardez les champs 'Depends' et 'OS_type' (il peut y avoir plus de une entrée correspondante). Si le package dépend d'une version de R ultérieure à celle utilisée, il est possible qu'une version antérieure soit disponible et fonctionnera avec votre version de R : pour CRAN, recherchez « Anciennes sources » sur la page d'accueil CRAN du package et manuellement récupérez une version appropriée (d'âge comparable à votre version de R).

[3] Si un proxy doit être défini, consultez ?download.file.

Les utilisateurs naïfs oublient parfois qu'en plus d'installer un package, ils doivent utiliser une bibliothèque pour rendre ses fonctionnalités disponibles.

6.3.1 Fenêtres

Ce que fait install.packages par défaut est différent sur les systèmes Unix (sauf macOS) et Windows. Sur les Unix-alikes, il consulte la liste des packages sources disponibles sur CRAN (ou autre(s) référentiel(s), télécharge la dernière version des sources des packages et les installe (via R CMD INSTALL). Sous Windows, il examine (par défaut) d'abord la liste des versions binaires des packages disponibles pour votre version de R et télécharge les dernières versions (le cas échéant). Si aucune version binaire n'est disponible ou si la version source est plus récente, il installera les versions sources des packages sans code C/C++/Fortran compilé, et proposera de le faire pour ceux qui en ont, si make est disponible (et cela peut être réglé par option "install.packages.compile.from.source").

Sous Windows, install.packages peut également installer un package binaire à partir d'un fichier zip local (ou de l'URL d'un tel fichier) en définissant l'argument repos sur NULL. Rgui.exe a un menu Packages avec une interface GUI pour installer.packages, update.packages et la bibliothèque.

Les packages binaires Windows pour R sont distribués sous la forme d'un seul binaire contenant l'une ou les deux architectures (32 et 64 bits). A partir de R 4.2.0, ils contiennent toujours uniquement l'architecture 64 bits.

R CMD INSTALL fonctionne sous Windows pour installer les packages sources. Aucun outil supplémentaire n'est nécessaire si le package ne contient pas de code compilé, et install.packages(type="source") fonctionnera pour de tels packages. Ceux qui ont du code compilé ont besoin des outils (voir Section 3.1.1 [Le jeu d'outils Windows], page 16). Les outils sont trouvés automatiquement par R lorsqu'ils sont installés par le programme d'installation du jeu d'outils. Voir Building R et les packages (https://cran.r-project.org/bin/windows/base/howto-R-devel.html) pour plus de détails.

Des problèmes d'autorisation occasionnels après le déballage des paquets sources ont été observés sur certains systèmes : ils ont été contournés en définissant la variable d'environnement R_INSTALL_TAR sur "tar.exe".

Si vous disposez uniquement d'un package source connu pour fonctionner avec R actuel et que vous souhaitez simplement une version binaire de Windows, vous pouvez utiliser le service de création proposé sur https://win-builder.r-project.org/ .

Pour presque tous les packages, R CMD INSTALL tentera d'installer les versions 32 et 64 bits d'un package s'il est exécuté à partir d'une installation 32/64 bits de R (seules les versions et installations 64 bits sont prises en charge depuis R 4.2.0) . Il signalera le succès si l'installation de l'architecture du R en cours d'exécution a réussi, que l'autre architecture ait été installée avec succès ou non. Les exceptions sont les packages avec un script configure.win non vide ou qui utilisent src/Makefile.win. Si configure.win fait quelque chose d'approprié aux deux architectures, utilisez l'option 4 --force-biarch : sinon R CMD INSTALL --merge-multiarch peut être appliqué à une archive tar source pour fusionner des installations séparées de 32 et 64 bits. (Cela ne peut être appliqué qu'à une archive tar et ne réussira que si les deux installations réussissent.)

Si vous disposez d'un package sans code compilé et sans aide spécifique à Windows, vous pouvez compresser une installation sur un autre système d'exploitation et l'installer à partir de ce fichier zip sous Windows. Cependant, un tel package peut être installé à partir des sources sous Windows sans aucun outil supplémentaire.

6.3.2 macOS

Sur macOS, install.packages fonctionne comme sur d'autres systèmes de type Unix, mais il existe un type supplémentaire mac.binary (disponible pour la distribution CRAN mais pas lors de la compilation de R à partir des sources) qui peut être transmis à install.packages afin de téléchargez et installez les packages binaires à partir d'un référentiel approprié. Ces fichiers de packages binaires pour macOS portent l'extension

[4] pour un petit nombre de packages CRAN pour lesquels cela est connu pour être sûr et nécessaire au constructeur automatique, c'est la valeur par défaut. Regardez la source de tools:::.install_packages pour la liste. Il peut également être spécifié dans le fichier DESCRIPTION du package.

'.tgz'. L'interface graphique R.app fournit des menus pour l'installation de packages binaires ou sources, à partir de CRAN, d'autres référentiels ou de fichiers locaux.

Sur les builds R utilisant des packages binaires, la valeur par défaut est de taper les deux : cela examine d'abord la liste des packages binaires disponibles pour votre version de R et installe les dernières versions (le cas échéant). Si aucune version binaire n'est disponible ou si la version source est plus récente, il installera les versions sources des packages sans code C/C++/Fortran compilé et proposera de le faire pour ceux qui en ont, si make est disponible.

Notez que la plupart des packages binaires qui incluent du code compilé sont liés à une série particulière (par exemple R 4.2.x ou 4.3.x) de R.

L'installation de packages sources qui ne contiennent pas de code compilé ne devrait fonctionner sans outils supplémentaires. Pour les autres, vous aurez besoin des « Outils de ligne de commande » pour Xcode et des compilateurs qui correspondent à ceux utilisés pour construire R : voir Section C.3 [macOS], page 60.

Le package rJava (https://CRAN.R-project.org/package=rJava) et ceux qui en dépendent nécessitent un runtime Java installé et plusieurs packages nécessitent l'installation de X11, y compris ceux utilisant Tk. Voir Section C.3 [macOS], page 60, et Section C.3.7 [Java (macOS)], page 66. Le package rjags (https://CRAN.R-project.org/package=rjags) nécessite une version de JAGS installé sous /usr/local, comme ceux sur https://sourceforge.net/projects/mcmc-jags/files/JAGS/ 4.x/ Mac%20OS%20X/.

L'extension Tcl/Tk BWidget était autrefois distribuée avec R mais ne l'est plus ; Tktable a été distribué avec la plupart des versions de R (mais pas les versions 4.0.0 et non 'arm64' de 4.1.0 ou 4.1.1).

Les compilateurs par défaut spécifiés sont affichés dans le fichier /Library/Frameworks/R.framework/Resources/etc/ Makeconf. Au moment de la rédaction, ces paramètres supposaient que les compilateurs C, For-tran et C++ étaient sur le chemin (voir Section C.3 [macOS], page 60). Les paramètres peuvent être modifiés, soit en éditant ce fichier, soit dans un fichier tel que ~/.R/Makevars (voir la section suivante).
Les entrées qui peuvent devoir être modifiées incluent 'CC', 'CXX', 'FC', 'FLIBS' et les indicateurs correspondants, et peut-être 'CXXCPP', 'DYLIB_LD', 'MAIN_LD', 'SHLIB_CXXLD' et 'SHLIB_LD', ainsi que leurs variantes « CXX11 », « CXX14 », « CXX17 » et « CXX20 ».

Ainsi, par exemple, vous pouvez sélectionner un clang LLVM spécifique pour C et C++ avec de nombreux vérification en ayant dans ~/.R/Makevars CC = /

```
usr/local/clang/bin/clang -isysroot /Library/Developer/
    CommandLineTools/SDKs/MacOSX.sdk
CXX = /usr/local/clang/bin/clang++ -isysroot
    /Bibliothèque/Développeur/CommandLineTools/SDK/MacOSX.sdk
CXX11 = $CXX
CXX14 = $CXX
CXX17 = $CXX
CXX20 = $CXX
CFLAGS = -g -O2 -Mur -pédant -Wconversion -Wno-sign-conversion
CXXFLAGS = -g -O2 -Mur -pédantique -Wconversion -Wno-sign-conversion
CXX11FLAGS = $CXXFLAGS
CXX14FLAGS = $CXXFLAGS
CXX17FLAGS = $CXXFLAGS
CXX20FLAGS = $CXXFLAGS
```

(longues lignes divisées pour le manuel uniquement) et pour la distribution macOS actuelle de gfortran sur https://mac.r-project.org/tools/

```
FC = /opt/gfortran/bin/gfortran (arm64)

FLIBS = -L/opt/gfortran/lib/gcc/aarch64-apple-darwin20.0/12.2.0 -L/opt/gfortran/lib -lgfortran -lemutls_w
    -lquadmath (Intel)
```

FLIBS = -L/opt/gfortran/lib/gcc/x86_64-apple-darwin20.0/12.2.0 -L/opt/gfortran/lib -lgfortran -lquadmath

(ligne interrompue ici pour le manuel uniquement).

Si cette version de Clang prend en charge OpenMP, vous pouvez ajouter

SHLIB_OPENMP_CFLAGS = -fopenmp
SHLIB_OPENMP_CXXFLAGS = -fopenmp

pour compiler des packages utilisant OpenMP. Il sera également nécessaire de faire en sorte que la bibliothèque libomp.dylib soit trouvée à la fois au moment de l'installation et au moment de l'exécution, par exemple en la copiant/en la liant à un endroit recherché, comme /usr/local/lib.

Apple inclut de nombreuses bibliothèques Open Source dans macOS mais de plus en plus sans les en-têtes correspondants (pas même dans Xcode ni dans les outils de ligne de commande) : il s'agit souvent de versions assez anciennes. Si vous installez des packages à partir des sources en les utilisant, il est généralement plus simple d'installer une copie à jour et liée statiquement du package Open Source à partir de ses sources ou de https://mac.r-project.org/bin/ . Mais parfois, il est souhaitable/nécessaire d'utiliser la bibliothèque liée dynamiquement d'Apple, auquel cas les en-têtes appropriés peuvent être extraits des sources5 disponibles via https://opensource.apple.com – cela a été utilisé pour iodbc.

Ceux qui utilisent Command Line Tools / Xcode 12 ou version ultérieure (tel que publié pour macOS 11 'Big Sur')
je veux probablement faire en sorte que le drapeau

-Wno-implicit-function-declaration

fait partie de CFLAGS. Apple a modifié la valeur par défaut pour faire des déclarations implicites une erreur de compilation et les auteurs de packages et de logiciels externes ne savaient pas que cela pouvait être fait - la plupart des problèmes rencontrés concernaient les scripts de configuration.

Une certaine prudence peut être nécessaire lors de la sélection des compilateurs lors de l'installation de logiciels externes à utiliser avec des packages. Les compilateurs « système » utilisés lors de la construction de R sont clang et clang++, mais la chaîne d'outils Apple fournit également des compilateurs appelés gcc et g++ qui, malgré leurs noms, sont basés sur LLVM et libc++ comme ceux du système et qui se comportent presque de la même manière que les compilateurs « système » utilisés lors de la construction de R. ceux du système. La plupart des logiciels Open Source ont un script de configuration développé à l'aide de GNU autoconf et sélectionnent donc gcc et g++ comme compilateurs par défaut : cela fonctionne généralement bien.
Par souci de cohérence, on peut utiliser

./configure CC=clang CFLAGS=-O2 CXX=clang++ CXXFLAGS=-O2

(en évitant le -g par défaut d'autoconf). Soyez prudent si vous placez le répertoire /usr/local/gfortran/bin sur votre chemin car il contient les (réels) gcc et g++ qui peuvent être trouvés à la place des commandes fournies par Apple, et risquent de ne pas pouvoir trouver les en-têtes et les bibliothèques6 de le SDK. À partir de R 4.3.0, R CMD INSTALL et install.packages() tentent d'invoquer configure avec les mêmes compilateurs et indicateurs que ceux utilisés pour construire R.

Pour 'arm64', tous les scripts de configuration n'ont pas été mis à jour pour reconnaître la plate-forme et peuvent donc nécessiter l'indicateur --build=aarch64-apple-darwin20.1.0. Sachez également que l'exécution des compilateurs à partir d'une application 'x86_64' les fait passer à la génération de code pour ce processeur : cela s'applique à un terminal, un shell, un ancien cmake ou une marque (non système), et à partir de R CMD INSTALL ou install. paquets(). On peut utiliser

./configure CC="clang -arch arm64" CFLAGS=-O2 CXX="clang++ -arch arm64" CXXFLAGS=-O2

pour forcer le code 'arm64'.

5 Notez que la capitalisation et la gestion des versions peuvent différer du projet Open Source.

6 Depuis Big Sur, ces bibliothèques ne sont pas visibles publiquement : les compilateurs du système renvoient plutôt à des fichiers de « définition basée sur le texte » (.tbd).

6.3.3 Personnalisation de la compilation des packages Les

indicateurs de compilation spécifiques au système R et au package peuvent être remplacés ou ajoutés en définissant les variables Make appropriées dans le fichier personnel HOME/.R/Makevars-R_PLATFORM (mais HOME/.R/Makevars.win ou HOME/.R/Makevars.win64 sous Windows), ou si cela n'existe pas, HOME/.R/Makevars, où 'R_PLATFORM' est la plateforme pour laquelle R a été construit, telle que disponible dans le composant plateforme de R variable R.version. Le chemin complet vers un fichier personnel alternatif[7] peut être spécifié via la variable d'environnement R_MAKEVARS_USER.

Les développeurs de packages sont encouragés à utiliser ce mécanisme pour activer une quantité raisonnable de messages de diagnostic (« avertissements ») lors de la compilation, comme par exemple -Wall -pedantic pour les outils de GCC, la collection de compilateurs GNU ou pour clang.

A noter que ce mécanisme peut également être utilisé lorsqu'il est nécessaire de modifier l'optimisation niveau lors de l'installation d'un package particulier. Par exemple

```
## pour le code C
CFLAGS = -g -O -mtune=native ## pour le
code C++
CXXFLAGS = -g -O -mtune=native ## pour le
code C++11
CXX11FLAGS = -g -O -mtune=native ## pour le
code Fortran de forme fixe
FFLAGS = -g -O -mtune=native ## pour le
code C17
C17FLAGS = -g -O -mtune=native -Wno-strict-prototypes
```

Notez que si vous avez spécifié une norme C++ ou C autre que celle par défaut, vous devez définir le ou les indicateurs appropriés à cette norme.

Une autre utilisation consiste à remplacer les paramètres dans une installation binaire de R. Par exemple, pour le distribution actuelle de gfortran sur https://mac.r-project.org/tools/

```
FC = /opt/gfortran/bin/gfortran (arm64)

FLIBS = -L/opt/gfortran/lib/gcc/aarch64-apple-darwin20.0/12.2.0 -L/opt/gfortran/lib -lgfortran
    -lemutls_w -lquadmath (Intel)

FLIBS = -L/opt/gfortran/lib/gcc/x86_64-apple-darwin20.0/12.2.0 -L/opt/gfortran/lib -lgfortran
    -lquadmath (ligne interrompue ici pour le manuel uniquement).
```

Il est également prévu un fichier Makevars.site à l'échelle du site sous R_HOME/etc (dans un répertoire spécifique à la sous-architecture, le cas échéant). Ceci est lu immédiatement après Makeconf, et le chemin vers un fichier alternatif peut être spécifié par la variable d'environnement R_MAKEVARS_SITE.

Notez que ces mécanismes ne fonctionnent pas avec les packages qui ne parviennent pas à transmettre les paramètres aux sous-marques, peut-être en lisant etc/Makeconf dans les makefiles des sous-répertoires. Heureusement, de tels forfaits sont inhabituels.

6.3.4 Sous-architectures multiples Lors de l'installation

de packages à partir de leurs sources, il y a quelques considérations supplémentaires sur les installations qui utilisent des sous-architectures. Ceux-ci sont couramment utilisés sous Windows mais peuvent en principe être utilisés sur d'autres plateformes.

Lorsqu'un package source est installé par une version de R qui prend en charge plusieurs sous-architectures, le processus d'installation normal installe les packages pour toutes les sous-architectures. Les exceptions sont

[7] utiliser un chemin contenant des espaces est susceptible de poser des problèmes

Similaires à Unix

où se trouve un script de configuration ou un fichier src/Makefile.

les fenêtres

où il y a un script configure.win non vide, ou un fichier src/Makefile.win (avec quelques exceptions où le package est connu pour avoir un configure.win indépendant de l'architecture, ou si --force-biarch ou le champ 'Biarch ' dans le fichier DESCRIPTION est utilisé pour l'affirmer).

Dans ces cas, seule l'architecture actuelle est installée. D'autres sous-architectures peuvent être installées par

R CMD INSTALL --libs-only pkg en utilisant le

chemin vers R ou R --arch pour sélectionner la sous-architecture supplémentaire. Il existe également R CMD INSTALL --merge-multiarch pour construire et fusionner les deux architectures, en commençant par une source
archive tar.

6.3.5 Compilation d'octets Depuis R 3.6.0,

tous les packages sont par défaut compilés par octets.

La compilation d'octets peut être contrôlée paquet par paquet par le champ 'ByteCompile' dans le
Fichier DESCRIPTION.

6.3.6 Logiciel externe

Certains packages R contiennent du code compilé qui renvoie à des bibliothèques de logiciels externes. À moins que la bibliothèque externe ne soit liée statiquement (ce qui est fait autant que possible pour les packages binaires sous Windows et macOS), les bibliothèques doivent être trouvées lorsque le package est chargé et pas seulement lors de son installation. La manière dont cela doit être effectué dépend du système d'exploitation (et dans certains cas de la version).

Pour les systèmes Unix à l'exception de macOS, le mécanisme principal est le cache ld.so contrôlé par ldconfig : les bibliothèques dynamiques externes enregistrées dans ce cache seront trouvées. Les emplacements de bibliothèque standard seront couverts par le cache, et un logiciel bien conçu ajoutera ses emplacements (comme le fait par exemple openmpi sur Fedora). Le mécanisme secondaire consiste à consulter la variable d'environnement LD_LIBRARY_PATH. Le script R contrôle cette variable et la définit sur la concaténation de R_LD_LIBRARY_PATH, R_JAVA_LD_LIBRARY_PATH et la valeur d'environnement de LD_LIBRARY_PATH. Les deux premiers ont des valeurs par défaut qui sont normalement définies lors de l'installation de R (mais peuvent être remplacées dans l'environnement), donc LD_LIBRARY_PATH est le meilleur choix à définir pour un utilisateur.

Sur macOS, le mécanisme principal consiste à intégrer le chemin absolu vers les bibliothèques dynamiques dépendantes dans un objet lors de sa compilation. Peu de packages R permettent de le faire, mais cela peut être édité[8] via install_name_tool — qui ne traite que des dépendances directes et celles-ci devraient également être compilées pour inclure les chemins absolus de leurs dépendances. Si le choix du chemin absolu doit être différé au temps de chargement, la manière dont ils sont résolus est décrit dans man dyld : le rôle de LD_LIBRARY_PATH est remplacé sur macOS par DYLD_LIBRARY_PATH et DYLD_FALLBACK_LIBRARY_PATH. L'exécution de R CMD otool -L sur l'objet partagé du package indiquera où (le cas échéant) ses dépendances sont résolues. DYLD_FALLBACK_LIBRARY_PATH est préféré (et c'est celui qui est manipulé par le script R), mais depuis la version 10.11 (« El Capitan »), le comportement par défaut a été modifié pour des raisons de sécurité afin d'ignorer ces variables d'environnement lors de l'appel d'un script shell (et R est un script shell). Cela constitue la seule option portable pour définir R_LD_LIBRARY_PATH dans l'environnement, quelque chose comme export R_LD_LIBRARY_PATH="'R RHOME'/lib:/opt/local/lib"

Les règles précises permettant à Windows de rechercher les DLL sont complexes et dépendent de la version de Windows. Mais pour les besoins actuels, la solution principale consiste à placer les répertoires contenant les DLL vers lesquels le package est lié (et toutes ces DLL vers lesquelles sont liés) sur le PATH.

[8] Ils doivent avoir été créés en utilisant -headerpad_max_install_names, qui est la valeur par défaut pour un package R.

Le danger de toute méthode impliquant la définition de variables d'environnement est de masquer par inadvertance une bibliothèque système. C'est moins pour DYLD_FALLBACK_LIBRARY_PATH et pour en ajoutant à PATH sous Windows (car il devrait déjà contenir les chemins de la bibliothèque système).

6.4 Mise à jour des packages

La commande update.packages() est le moyen le plus simple de garantir que tous les packages de votre le système est à jour. Il télécharge la liste des packages disponibles et leurs versions actuelles, le compare avec ceux installés et propose de récupérer et d'installer ceux qui ont des versions ultérieures les référentiels.

Une interface alternative pour maintenir les packages à jour est fournie par la commande packageStatus(), qui renvoie un objet avec des informations sur tous les packages installés et disponibles sur plusieurs référentiels. Les méthodes d'impression et de synthèse donnent un aperçu de packages installés et disponibles, la méthode de mise à niveau propose de récupérer et d'installer les dernières versions des packages obsolètes.

Une information supplémentaire parfois utile renvoyée par packageStatus() est le statut d'un paquet, comme "ok", "mise à jour" ou "indisponible" (dans les dépôts actuellement sélectionnés). Par exemple

```
> inst <- packageStatus()$inst
> inst[inst$Status != "ok", c("Package", "Version", "Statut")]
                                Version du package Statut
Biobase                  Biobase 2.8.0 indisponible
RCurl                        RCurl 1.4-2 mise à niveau
Rgraphviz          Rgraphviz 1.26.0 indisponible
rgdal                    mise à jour rgdal 0.6-27
```

6.5 Suppression de paquets

Les packages peuvent être supprimés de plusieurs manières. À partir d'une invite de commande, ils peuvent être supprimés par

```
R CMD REMOVE -l /chemin/vers/bibliothèque pkg1 pkg2 ...
```

À partir d'un processus R en cours d'exécution, ils peuvent être supprimés en

```
> supprimer.packages(c("pkg1", "pkg2"),
                                lib = fichier.path("chemin", "vers", "bibliothèque"))
```

Enfin, on peut simplement supprimer le répertoire du package de la bibliothèque.

6.6 Configuration d'un référentiel de packages

Des utilitaires tels que install.packages peuvent être pointés vers n'importe quel référentiel de style CRAN et les utilisateurs R voudront peut-être créer le leur. La « base » d'un référentiel est une URL telle que https://www.stats.ox.ac.uk/pub/RWin/ : il doit s'agir d'un schéma d'URL pris en charge par download.packages (qui inclut également « https:// », « ftp:// » et « file:// »). Sous cette URL de base, il devrait y avoir être des arborescences de répertoires pour un ou plusieurs des types de distributions de packages suivants :

- "source" : situé dans src/contrib et contenant les fichiers .tar.gz. D'autres formes de compression peuvent être utilisées, par exemple les fichiers .tar.bz2 ou .tar.xz. Les référentiels complets contiennent les sources correspondant à n'importe quel paquet binaire, et dans tous les cas il est sage d'avoir un src/contrib zone avec un fichier PACKAGES éventuellement vide.
- "win.binary" : situé dans bin/windows/contrib/xy pour les versions R xyz et contenant Fichiers .zip pour Windows.
- "mac.binary" : situé dans bin/macosx/contrib/4.y pour les builds CRAN pour macOS pour R versions 4.yz, contenant des fichiers .tgz.

- "mac.binary.el-capitan" : situé dans bin/macosx/el-capitan/contrib/3.y pour le
  CRAN construit pour les versions R 3.yz, contenant des fichiers .tgz.

Chaque répertoire de terminal doit également contenir un fichier PACKAGES. Il peut s'agir d'une concaténation des fichiers DESCRIPTION des packages séparés par des lignes vides, mais seuls quelques champs sont nécessaires. Le moyen le plus simple de configurer un tel fichier est d'utiliser la fonction write_PACKAGES dans le package d'outils, et son aide explique quels champs sont nécessaires. En option, il peut également y avoir des fichiers PACKAGES.rds et PACKAGES.gz, téléchargés de préférence dans PACKAGES. (Si vous disposez d'un serveur mal configuré qui ne signale pas correctement les fichiers inexistants, vous pourriez avoir besoin de ces fichiers.)

Pour ajouter votre référentiel à la liste proposée par setRepositories(), consultez le fichier d'aide de cette fonction.

Les référentiels incomplets sont mieux spécifiés via un argument de contribution plutôt que via leur définition en tant que référentiel.

Un référentiel peut contenir des sous-répertoires, lorsque les descriptions dans le fichier PACKAGES des packages dans les sous-répertoires doivent inclure une ligne de la forme

Chemin : chemin/vers/sous-répertoire

— encore une fois, write_PACKAGES est le moyen le plus simple de configurer cela.

6.7 Vérification des packages sources installés

Il peut être pratique d'exécuter la vérification R CMD sur un package installé, en particulier sur une plateforme qui utilise des sous-architectures. Voici comment procéder, avec le paquet source dans le répertoire pkg (ou un nom de fichier tarball) : R CMD INSTALL -l libdir pkg > pkg.log 2>&1 R

CMD check -l libdir --install=check:pkg paquet .log

Lorsque des sous-architectures sont utilisées, la ligne de contrôle R CMD peut être répétée avec des architectures supplémentaires en

R --arch arch CMD check -l libdir --extra-arch --install=check:pkg.log pkg où --extra-arch sélectionne uniquement les vérifications qui dépendent du code installé et non celles qui analysent les sources. (Si plusieurs sous-architectures échouent uniquement parce qu'elles nécessitent des paramètres différents, par exemple des variables d'environnement, il peut être nécessaire d'ajouter --no-multiarch aux lignes INSTALL.) Sur les systèmes Unix, l'architecture à exécuter est sélectionnée par --arch : ceci peut également être utilisé sous Windows avec R_HOME/bin/R.exe, mais il est plus habituel de sélectionner le chemin d'accès au Rcmd.exe de l'architecture souhaitée.

Donc sous Windows pour installer, vérifier et empaqueter pour distribution un paquet source à partir d'une archive tar qui a été testé sur une autre plateforme que l'on pourrait utiliser

.../bin/x64/Rcmd INSTALL -l libdir tarball --build > pkg.log 2>&1

7 Internationalisation et localisation

L'internationalisation fait référence au processus permettant la prise en charge de nombreuses langues humaines, et la localisation à l'adaptation à un pays et une langue spécifiques.

Les versions actuelles de R prennent en charge tous les jeux de caractères que le système d'exploitation sous-jacent peut gérer. Ceux-ci sont interprétés en fonction des paramètres régionaux actuels, un sujet suffisamment compliqué pour mériter une section distincte. Notez cependant que R n'a pas de prise en charge intégrée des langues de droite à gauche et de la sortie bidirectionnelle, s'appuyant sur les services du système d'exploitation. Par exemple, la façon dont les vecteurs de caractères en UTF-8 contenant à la fois des chiffres anglais et des caractères hébreux sont imprimés dépend du système d'exploitation (et peut-être des paramètres régionaux).

L'autre aspect de l'internationalisation est la prise en charge de la traduction des messages. Ceci est activé dans presque toutes les versions de R.

7.1 Locaux

Une locale est une description de l'environnement local de l'utilisateur, y compris la langue préférée, le codage des caractères, la devise utilisée et ses conventions, etc. Les aspects des paramètres régionaux sont accessibles par les fonctions R Sys.getlocale et Sys.localeconv.

Le système de dénomination des paramètres régionaux est spécifique au système d'exploitation. Il existe un accord assez large sur les programmes, mais pas sur les détails de leur mise en œuvre. Une locale doit spécifier

- Un langage humain. Ceux-ci sont généralement spécifiés par une abréviation minuscule à deux caractères. suivant la norme ISO 639 (voir par exemple https://en.wikipedia.org/wiki/ISO_639-1).
- Un « territoire », utilisé principalement pour préciser la monnaie. Ceux-ci sont généralement spécifiés par une abréviation majuscule à deux caractères conformément à la norme ISO 3166 (voir par exemple https://en.wikipedia.org/wiki/ISO_3166).
- Un codage de jeu de caractères, qui détermine à la fois comment un flux d'octets doit être divisé en caractères et quels caractères représentent les sous-séquences d'octets. Parfois, la combinaison de la langue et du territoire est utilisée pour spécifier le codage, par exemple pour distinguer le chinois traditionnel du chinois simplifié.
- Facultativement, un modificateur, par exemple pour indiquer que l'Autriche doit être considérée comme pré- ou post-Euro. Le modificateur est également utilisé pour indiquer l'écriture (@latin, @cyrillic pour le serbe, @iqtelif) ou le dialecte de la langue (par exemple @saaho, un dialecte de l'afar, et @bokmal et @nynorsk, dialectes du norvégien considérés par certains systèmes d'exploitation comme distincts). langues, no et nn).

R concerne principalement le premier (pour les traductions) et le troisième. Notez que le jeu de caractères peut être déductible de la langue, car certains systèmes d'exploitation ne proposent qu'un seul jeu de caractères par langue.

7.1.1 Paramètres régionaux sous Unix-alikes

Linux moderne utilise les spécifications locales XPG1 qui ont la forme 'en_GB', 'en_GB.UTF-8', 'aa_ER.UTF-8@saaho', 'de_AT.iso885915@euro', les composants étant dans l'ordre indiqué ci-dessus. (Voir man locale et locale -a pour plus de détails.) Des schémas similaires sont utilisés par la plupart des systèmes Unix : certains (y compris certaines distributions de Linux) utilisent « .utf8 » plutôt que « .UTF-8 ».

Notez que même si les paramètres régionaux UTF-8 sont aujourd'hui presque universellement utilisés, des paramètres régionaux tels que 'en_GB' utilise des encodages 8 bits pour une compatibilité ascendante.

7.1.2 Paramètres régionaux sous Windows

Windows utilise également des paramètres régionaux, mais spécifiés de manière un peu moins concise. La plupart des utilisateurs rencontreront les paramètres régionaux uniquement via les menus déroulants, mais plus d'informations et de listes peuvent être trouvées en recherchant « Chaînes de pays de langue Windows ».

[1] 'X/Open Portability Guide', qui a connu plusieurs versions.

Il ne propose qu'un seul encodage par langue.

Une certaine prudence est nécessaire avec les noms de paramètres régionaux de Windows. Par exemple, le chinois est traditionnel Chinois et non chinois simplifié tel qu'il est utilisé dans la plupart des pays de langue chinoise.

7.1.3 Paramètres régionaux sous macOS

macOS prend en charge les paramètres régionaux à sa manière, mais l'interface graphique R tente de rendre cela plus facile pour les utilisateurs. Voir https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPInternational/ pour savoir comment les utilisateurs peuvent définir leurs paramètres régionaux. Comme avec Windows, les utilisateurs finaux ne verront généralement que des listes de langues/territoires. Les utilisateurs de R dans un terminal devront peut-être définir les paramètres régionaux sur quelque chose comme « en_GB.UTF-8 » s'ils sont par défaut « C » (comme c'est parfois le cas lors d'une connexion à distance et pour des tâches par lots : notez si le terminal définit l'environnement LANG. variable est une préférence (avancée), mais elle le fait par défaut).

En interne, macOS utilise une forme similaire à Linux : la principale différence par rapport aux autres Unix-alikes est que lorsqu'un jeu de caractères n'est pas spécifié, il est supposé être UTF-8.

7.2 Localisation des messages

La langue préférée pour les messages est par défaut issue des paramètres régionaux. Ceci peut être remplacé d'abord par la définition de la variable d'environnement LANGUAGE, puis[2] par les variables d'environnement LC_ALL, LC_MESSAGES et LANG. (Les trois derniers sont normalement utilisés pour définir les paramètres régionaux et ne devraient donc pas être nécessaires, mais le premier est uniquement utilisé pour sélectionner la langue des messages.) Le code s'efforce de mapper les paramètres régionaux aux langues, mais sur certains systèmes (notamment Windows) les noms de paramètres régionaux requis pour la variable d'environnement LC_ALL ne correspondent pas tous aux noms de langues XPG et il peut donc être nécessaire de définir LANGUAGE. (Un exemple est « LC_ALL=es » sous Windows, qui définit les paramètres régionaux sur l'estonien et la langue sur l'espagnol.)

Il est généralement possible de changer la langue une fois que R est exécuté via (pas Windows) Sys.setlocale("LC_MESSAGES", "new_locale"), ou en définissant une variable d'environnement telle que LANGUAGE, à condition que la langue vers laquelle vous changez[3] puisse être affichée dans le jeu de caractères actuel. Mais cela est spécifique au système d'exploitation et il est connu que cela cesse de fonctionner lors d'une mise à niveau du système d'exploitation. Notez que les messages traduits peuvent être mis en cache, donc tenter de changer la langue d'une erreur qui a déjà été générée dans une autre langue peut ne pas fonctionner.

Les messages sont divisés en domaines et des traductions peuvent être disponibles pour tout ou partie des messages d'un domaine. R utilise les domaines suivants.

- Domaine R pour les messages d'erreur et d'avertissement de niveau C provenant de l'interpréteur R.
- Domaine R-pkg pour les messages d'arrêt, d'avertissement et de message R dans chaque package, y compris R-base pour le package de base.
- Pack de domaine pour les messages de niveau C dans chaque package.
- Domaine RGui pour les menus, etc. du front-end de l'interface graphique R pour Windows.

Diviser les messages de cette manière permet à R d'être extensible : à mesure que les packages sont chargés, leurs catalogues de traduction de messages peuvent également être chargés.

R peut être construit sans prise en charge des traductions, mais il est activé par défaut.

Les domaines de niveau R et C sont subtilement différents, par exemple dans la manière dont les chaînes sont canonisées. calisé avant d'être transmis pour traduction.

Les traductions sont recherchées par domaine en fonction de la langue actuellement spécifiée, aussi spécifiquement que possible, ainsi par exemple un catalogue de traduction autrichien (« de_AT ») sera utilisé de préférence à un catalogue générique allemand (« de ») pour un utilisateur autrichien. Cependant, si une traduction spécifique

[2] Sur certains systèmes, définir LC_ALL ou LC_MESSAGES sur « C » désactive la LANGUE.

[3] Si vous essayez de passer du français au russe, sauf dans les paramètres régionaux UTF-8, vous constaterez peut-être que les messages passent en anglais.

Le catalogue existe mais ne contient pas de traduction, moins les catalogues spécifiques sont consultés.
Par exemple, R a des catalogues pour « en_GB » qui traduisent les américanismes (par exemple, « gray ») dans les messages standards en anglais.4 Deux autres exemples : il existe des catalogues pour « es », qui est l'espagnol tel qu'écrit en Espagne et ces catalogues sera également utilisé par défaut dans les pays hispanophones d'Amérique latine, ainsi que pour 'pt_BR', qui sont utilisés pour les paramètres régionaux brésiliens mais pas pour les paramètres régionaux spécifiant le Portugal.

Les traductions dans la bonne langue mais avec le mauvais jeu de caractères sont utilisées par un réencodage à la volée. La variable LANGUAGE (uniquement) peut être une liste séparée par deux points, par exemple « se:de », donnant un ensemble de langues par ordre décroissant de préférence. Une valeur spéciale est 'en@quot', qui peut être utilisée dans une locale UTF-8 pour avoir des messages d'erreur américains avec des paires de guillemets simples traduits en guillemets directionnels Unicode.

Si aucun catalogue de traduction approprié n'est trouvé ou si un message particulier n'est pas traduit dans aucun catalogue approprié, « anglais »5 est utilisé.

Voir https://developer.r-project.org/Translations30.html pour savoir comment préparer et installer des catalogues de traduction.

[4] la langue écrite en Angleterre : certaines personnes vivant aux USA s'approprient ce nom pour leur langue.
[5] avec les américanismes.

8 Choisir entre les versions 32 et 64 bits

Presque tous les processeurs actuels disposent de jeux d'instructions de 32 et 64 bits. Certains systèmes d'exploitation fonctionnant sur de tels processeurs offrent le choix de créer une version 32 bits ou 64 bits de R (et les détails sont donnés ci-dessous sous les systèmes d'exploitation spécifiques). Pour la plupart, une version 64 bits est la version par défaut et, de plus en plus, seule celle-ci est prise en charge : par exemple, macOS est uniquement en 64 bits depuis Catalina (sorti en 2019) et R ne prend en charge que Windows 64 bits depuis R 4.2.0.

Toutes les versions actuelles de R utilisent des entiers 32 bits (ceci est appliqué dans la version) et des réels double précision ISO/IEC 605591 , et calculent ainsi avec la même précision2 et avec les mêmes limites sur les tailles des quantités numériques. La principale différence réside dans la taille des pointeurs.

Les versions 64 bits présentent à la fois des avantages et des inconvénients :

- L'espace mémoire virtuel total mis à disposition d'un processus 32 bits est limité par la taille du pointeur à 4 Go, et sur la plupart des systèmes d'exploitation à 3 Go (voire 2 Go). Les limites pour les processus 64 bits sont beaucoup plus grandes (par exemple 8 à 128 To).

  R alloue de la mémoire pour les objets volumineux selon les besoins et supprime tous ceux inutilisés lors du garbage collection. Lorsque la taille des objets représente une fraction appréciable de la limite d'adresse, la fragmentation de l'espace d'adressage devient un problème et il se peut qu'il n'y ait aucun trou disponible correspondant à la taille demandée. Cela peut entraîner une récupération de place plus fréquente ou l'impossibilité d'allouer des objets volumineux. À titre indicatif, cela deviendra un problème pour les versions 32 bits avec des objets représentant plus de 10 % de la taille de l'espace d'adressage (environ 300 Mo) ou lorsque la taille totale des objets utilisés est d'environ un tiers (environ 1 Go).

- Seules les versions 64 bits prennent en charge les « vecteurs longs », ceux comportant 231 éléments ou plus (ce qui nécessite au moins au moins 16 Go de stockage pour chaque vecteur numérique).
- La plupart des systèmes d'exploitation 32 bits limitent par défaut la taille des fichiers à 2 Go (et cela peut également s'appliquer aux versions 32 bits sur des systèmes d'exploitation 64 bits). Cela peut souvent être contourné : configure sélectionne les définitions appropriées si cela est possible. Les versions 64 bits ont des limites beaucoup plus grandes.
- Comme les pointeurs sont plus grands, les structures de base de R sont plus grandes. Cela signifie que les objets R prennent plus de place et (généralement) plus de temps à manipuler. Ainsi, les versions 64 bits de R fonctionneront, toutes choses égales par ailleurs, plus lentement que les versions 32 bits.
- Cependant, les « autres choses » peuvent ne pas être égales. Dans le cas spécifique de « x86_64 » par rapport à « ix86 », le processeur 64 bits possède des fonctionnalités (telles que les instructions SSE2) dont la présence est garantie mais qui sont facultatives sur le processeur 32 bits, et dispose également de registres plus généraux. . Cela signifie que sur des puces comme un ordinateur de bureau Intel i7, la version vanille 64 bits de R a été environ 10 % plus rapide sous Linux et macOS. (Les processeurs des ordinateurs portables sont généralement relativement plus lents en mode 64 bits.)

Ainsi, pour plus de vitesse, vous souhaiterez peut-être utiliser une version 32 bits (en particulier sur un ordinateur portable), mais pour gérer de gros ensembles de données (et peut-être des fichiers volumineux), une version 64 bits. Vous pouvez souvent construire les deux et les installer au même endroit : Voir Section 2.6 [Sous-architectures], page 8.

Même sur les versions 64 bits de R, il existe des limites sur la taille des objets R (voir help("Memory-limits")), dont certaines proviennent de l'utilisation d'entiers 32 bits (en particulier dans le code Fortran).

Par exemple, chaque dimension d'un tableau est limitée à $2^{31} - 1$.

1 également connu sous le nom d' IEEE 754

2 au moins lors du stockage des quantités : la précision sur FPU peut varier

9 La bibliothèque Rmath autonome

Les routines prenant en charge les fonctions de distribution et spéciales1 dans R et quelques autres sont déclarées dans le fichier d'en-tête C Rmath.h. Ceux-ci peuvent être compilés dans une bibliothèque autonome pour être reliés à d'autres applications. (Notez qu'il ne s'agit pas d'une bibliothèque distincte lorsque R est construit et que la version autonome diffère de plusieurs manières.)

Les makefiles et autres sources nécessaires se trouvent dans le répertoire src/nmath/standalone, donc les instructions suivantes supposent qu'il s'agit du répertoire de travail actuel (dans l'arborescence du répertoire de construction sous Unix s'il est séparé des sources).

Rmath.h contient 'R_VERSION_STRING', qui est une chaîne de caractères contenant le R actuel version, par exemple "4.0.0".

Il existe un accès complet à la gestion par R de NaN, Inf et -Inf via des versions spéciales des macros et des fonctions

ISNAN, R_FINITE, R_log, R_pow et R_pow_di

et les constantes (externes) R_PosInf, R_NegInf et NA_REAL.

Il n'y a aucun support pour la notion de valeurs manquantes de R, en particulier ni pour NA_INTEGER ni la distinction entre NA et NaN pour les doubles.

Un peu de prudence est nécessaire pour utiliser les routines de nombres aléatoires. Vous devrez fournir le générateur de nombres aléatoires uniformes

double unif_rand (vide)

ou utilisez celui fourni (et avec une bibliothèque partagée ou une DLL, vous devrez peut-être utiliser celui fourni, qui est le Marsaglia-multicarry avec un point d'entrée set_seed(unsigned int,

unsigned int) pour définir ses graines).

Les fonctionnalités permettant de modifier le générateur de nombres aléatoires normal sont disponibles via la constante N01_kind. Cela prend les valeurs du type d'énumération

```
typedef énumération {
    BUGGY_KINDERMAN_RAMAGE,
    AHRENS_DIETER,
    BOÎTE_MULLER,
    USER_NORM,
    INVERSION,
    KINDERMAN_RAMAGE
}Type N01 ; (et
```

'USER_NORM' n'est pas disponible).

9.1 Similaires à Unix

Si R n'a pas déjà été créé dans l'arborescence des répertoires, configure doit être exécuté comme décrit dans les principales instructions de construction.

Puis (dans src/nmath/standalone)

faire

créera des bibliothèques autonomes libRmath.a et libRmath.so (libRmath.dylib sur macOS) : 'make static' et 'make shared' n'en créeront qu'une.

Pour utiliser les routines dans vos propres programmes C ou C++, incluez

#définir MATHLIB_STANDALONE

[1] par exemple les fonctions Bessel, bêta et gamma

#include <Rmath.h>

et créez un lien vers '-lRmath' (et '-lm' si nécessaire sur votre système d'exploitation). Le fichier exemple test.c ne fait rien d'utile, mais est fourni pour tester le processus (via make test). Notez que vous ne pourrez probablement pas l'exécuter à moins d'ajouter le répertoire contenant libRmath.so à la variable d'environnement LD_LIBRARY_ PATH (libRmath.dylib, DYLD_FALLBACK_LIBRARY_PATH sur macOS).

Les cibles

faire installer

faire la désinstallation

va (dé)installer l'en-tête Rmath.h et les bibliothèques partagées et statiques (si construites). prefix= et DESTDIR sont pris en charge, ainsi qu'un contrôle plus précis comme décrit pour la version principale.

'make install' installe un fichier que pkg-config pourra utiliser par exemple

$(CC) 'pkg-config --cflags libRmath' -c test.c $(CC) 'pkg-config --libs libRmath' test.o -o test

Sur certains systèmes, « make install-strip » installera une bibliothèque partagée supprimée.

9.2 Fenêtres

Vous devez configurer presque tous les outils pour créer R, puis les exécuter (dans un shell de type Unix)

(cd ../../gnuwin32; make MkRules) (cd ../../include; make -f Makefile.win config.h Rconfig.h Rmath.h) make -f Makefile.win

Alternativement, dans un shell cmd.exe, utilisez

cd ../../include make -f Makefile.win config.h Rconfig.h Rmath.h cd ../nmath/standalone make -f Makefile.win

Cela crée une bibliothèque statique libRmath.a et une DLL Rmath.dll. Si vous voulez une bibliothèque d'importation libRmath.dll.a (vous n'en avez pas besoin), utilisez

make -f Makefile.win implib partagé

Pour utiliser les routines dans vos propres programmes C ou C++ à l'aide de MinGW-w64, incluez

#définir MATHLIB_STANDALONE
#include <Rmath.h>

et un lien vers '-lRmath'. Cela utilisera le premier trouvé de libRmath.dll.a, libRmath.a et Rmath.dll dans cet ordre, donc le résultat dépend des fichiers présents. Vous devriez pouvoir forcer une liaison statique ou dynamique via

-Wl,-Bstatic -lRmath -Wl,Bdynamic -Wl,-Bdynamic -lRmath ou en créant un lien

vers des fichiers explicites (comme dans la cible 'test' de Makefile.win : cela crée deux exécutables, test.exe qui est dynamiquement lié et test-static.exe, qui est lié statiquement).

Il est possible de créer un lien vers Rmath.dll en utilisant d'autres compilateurs, soit directement, soit via une bibliothèque d'import : si vous faites une bibliothèque d'import MinGW-w64 comme ci-dessus, vous créerez un fichier Rmath.def qui pourra être utilisé (éventuellement après édition) pour créer une bibliothèque d'importation pour d'autres systèmes tels que Visual C++.

Si vous utilisez des liens dynamiques, vous devez utiliser

#définir MATHLIB_STANDALONE
#définir RMATH_DLL

[2] y compris la copie de MkRules.dist vers MkRule.local et la sélection de l'architecture.

```
#include <Rmath.h>
```

pour garantir que les constantes comme NA_REAL sont correctement liées. (L'importation automatique fonctionnera probablement avec MinGW-w64, mais il vaut mieux en être sûr. Cela fonctionnera probablement également avec VC++, Borland et des compilateurs similaires.)

Annexe A Autres programmes essentiels et utiles sous un système Unix

Cette annexe donne des détails sur les programmes dont vous aurez besoin pour construire R sur des plates-formes de type Unix, ou qui seront utilisés par R s'ils sont trouvés par configure.

N'oubliez pas que certains systèmes de gestion de paquets (tels que RPM et Debian/Ubuntu) font une distinction entre la version utilisateur d'un paquet et la version de développement. Ce dernier porte généralement le même nom mais avec l'extension '-devel' ou '-dev' : vous devez installer les deux versions.

A.1 Programmes et bibliothèques essentiels

Vous avez besoin d'un moyen de compiler C et Fortran 90 (voir Section B.6 [Utiliser Fortran], page 53).
Votre compilateur C doit être conforme à la norme ISO/IEC 600591 POSIX 1003.1 et C99.2 R essaie de choisir les indicateurs3 appropriés pour les compilateurs C qu'il connaît, mais vous devrez peut-être définir CC ou CFLAGS de manière appropriée. (Notez que les options essentielles pour exécuter le compilateur même pour la liaison, telles que celles permettant de définir l'architecture, doivent être spécifiées dans le cadre de CC plutôt que dans CFLAGS.)

Sauf si vous ne souhaitez pas afficher de graphiques à l'écran (ou utiliser macOS), vous devez installer « X11 », y compris ses en-têtes et ses bibliothèques clientes. Pour les distributions Fedora/RedHat récentes, cela signifie (au moins) les RPM 'libX11', 'libX11-devel', 'libXt' et 'libXt-devel'. Sur Debian/Ubuntu, nous recommandons le méta-paquet 'xorg-dev'. Si vous ne les souhaitez vraiment pas, vous devrez configurer explicitement R sans X11, en utilisant --with-x=no.

L'édition de la ligne de commande (et l'achèvement des commandes) dépend de la bibliothèque GNU readline (y compris ses en-têtes) : la version 6.0 ou ultérieure est nécessaire pour que toutes les fonctionnalités soient activées. Sinon, vous devrez configurer avec --with-readline=no (ou équivalent).

Une fonction iconv suffisamment complète est essentielle. L'utilisation de R nécessite qu'iconv soit capable de traduire entre "latin1" et "UTF-8", de reconnaître "" (comme encodage actuel) et "ASCII", et de traduire vers et depuis les formats de caractères larges Unicode "UCS -[24][BL]E" — ceci est vrai par défaut pour la glibc4 mais pas pour la plupart des Unix commerciaux. Cependant, vous pouvez utiliser GNU libiconv (tel qu'utilisé sur macOS : voir https://www.gnu.org/software/libiconv/).

Le système d'exploitation doit disposer d'une prise en charge suffisante5 pour les types de caractères larges : ceci est vérifié lors de la configuration. Certaines fonctions C996 sont requises et vérifiées lors de la configuration. Un petit nombre de fonctions POSIX7 sont essentielles, et d'autres8 seront utilisées si disponibles.

Installations de zlib (version 1.2.5 ou ultérieure), libbz2 (version 1.0.6 ou ultérieure : appelée bzip2-libs/bzip2-devel ou libbz2-1.0/libbz2-dev par certaines distributions Linux) et liblzma9 version 5.0.3 ou ultérieure sont requis.

---

1 également connu sous le nom d' IEEE 754

2 Notez que les compilateurs C11 n'ont pas besoin d'être conformes à C99 : R nécessite la prise en charge des tableaux doubles complexes et de longueur variable qui sont facultatifs en C11 mais obligatoires en C99. C17 (également connu sous le nom de C18 car publié en 2018) est une « version de correction de bugs » de C11, clarifiant la norme. Cependant, tous les compilateurs récents connus en mode C11 ou C17 sont compatibles C99 et la plupart utilisent par défaut C17.

3 Les exemples sont -std=gnu99, -std=c99 et -c99.

4 Cependant, il est possible de rompre le comportement par défaut de la glibc en respécifiant les modules gconv à charger.

5 plus précisément, la fonctionnalité C99 des en-têtes wchar.h et wctype.h, les types wctans_t et mbstate_t et les fonctions mbrtowc, mbstowcs, wcrtomb, wcscoll, wcstombs, wctrans, wctype et iswctype. y compris expm1, hypot, log1p, closeint et

6 va_copy. y compris les appels système opendir, readdir, closeir,

7 popen, stat, glob, access, getcwd et chdir, sélectionnez sur un système Unix et putenv ou setenv. tel que realpath, lien symbolique. le plus souvent distribué dans le cadre de xz : les noms

8 possibles dans les distributions

9 Linux incluent xz-devel/xz-libs et liblzma-dev.

PCRE1 (version 8.32 ou ultérieure, anciennement connu sous le nom de PCRE) ou PCRE2 est requis : PCRE2 est préféré et l'utilisation de PCRE1 nécessite l'option de configuration --with-pcre1. Seules la bibliothèque 8 bits et les en-têtes sont nécessaires s'ils sont emballés séparément. La prise en charge JIT (facultatif) est souhaitable pour obtenir les meilleures performances. Pour PCRE2 >= 10.30 (ce qui est souhaitable car la correspondance a été réécrite pour ne pas utiliser la récursion et les tables Unicode ont été mises à jour vers la version 10)

./configure --enable-jit suffit. Si vous

construisez PCRE1 pour une utilisation avec R, une commande de configuration appropriée peut être

./configure --enable-utf --enable-unicode-properties --enable-jit --disable-cpp

L'indicateur --enable-jit est pris en charge pour la plupart des processeurs courants mais ne fonctionne pas (bien ou pas du tout) pour macOS « arm64 ».

Certains packages nécessitent les « propriétés Unicode » qui sont facultatives pour PCRE1 : prise en charge de ceci et JIT peuvent être vérifiés au moment de l'exécution en appelant pcre_config().

La bibliothèque libcurl (version 7.28.0 ou ultérieure) est requise. Les informations sur libcurl se trouvent dans le script curl-config : si cela est manquant ou doit être remplacé10, il existe des macros pour le faire décrites dans le fichier config.site.

Un programme tar est nécessaire pour décompresser les sources et les packages (y compris les packages recommandés). Une version 11 capable de détecter automatiquement les archives compressées est préférable pour une utilisation avec untar() : le script de configuration recherche gtar et gnutar avant tar – utilisez la variable d'environnement TAR pour remplacer cela. (Sur les systèmes NetBSD/OpenBSD, définissez-le sur bsdtar s'il est installé.)

Il doit y avoir des versions appropriées des outils grep et sed : les problèmes proviennent généralement des anciennes variantes d'AT&T et de BSD. configure essaiera de trouver des versions appropriées (y compris en recherchant dans /usr/xpg4/bin qui est utilisé sur certains Unix commerciaux).

Vous ne pourrez pas créer la plupart des manuels à moins d'avoir installé texi2any version 5.1 ou ultérieure (ce qui nécessite Perl), et sinon la plupart des manuels HTML seront liés à une version sur CRAN. Pour créer des versions PDF des manuels, vous aurez également besoin d'installer le fichier texinfo.tex (qui fait partie de la distribution GNU texinfo mais fait souvent partie du package TEX dans les redistributions) ainsi que texi2dvi.
[12] De plus, les versions de texi2dvi et texinfo.tex doivent être compatibles : nous avons constaté des problèmes avec les anciennes distributions TEX.

Si vous souhaitez construire à partir du référentiel R Subversion, alors texi2any est fortement recommandé car il est utilisé pour créer des fichiers qui sont dans l'archive mais non stockés dans le référentiel Subversion.

La documentation PDF (y compris doc/NEWS.pdf) et les vignettes de construction nécessitent pdftex et pdflatex. Nous avons besoin de LATEX version 2005/12/01 ou ultérieure (pour le support UTF-8). La création de manuels de packages PDF (y compris le manuel de référence R) et de vignettes est sensible à la version de l' hyperref du package LATEX et nous recommandons que la distribution TEX utilisée soit maintenue à jour. Un certain nombre de packages LATEX standard sont requis pour les manuels PDF (y compris url et certains packages de polices tels que times et helvetic et également amsfonts) et d'autres tels que hyperref et inconsolata sont souhaitables (et sans eux, vous devrez peut-être modifier les valeurs par défaut de R. : voir Section 2.3 [Réalisation des manuels], page 5). Notez que le package hyperref (actuellement) nécessite les packages kvoptions, ltxcmds et refcount, et qu'inconsolata nécessite xkey-val. La construction des vignettes de base nécessite fancyvrb, natbib, parskip (qui nécessite actuellement etoolbox) et des listings. Pour les distributions basées sur TeX Live, l'approche la plus simple peut être d'installer les collections collection-latex, collection-fontsrecommended, collection-latexrecommended,

---

[dix] par exemple pour spécifier une liaison statique avec une construction qui possède à la fois des bibliothèques partagées et statiques.

[11] Comme GNU tar 1.15 ou version ultérieure, bsdtar (de https://github.com/libarchive/libarchive/, utilisé comme tar par FreeBSD et macOS 10.6 et versions ultérieures) ou tar de Heirloom Toolchest (https://heirloom.sourceforge. net/tools.html), bien que ce dernier ne prenne pas en charge la compression xz. texi2dvi est normalement un script shell. Certains des problèmes observés avec les versions

[12] défectueuses de texi2dvi peuvent être contournés en définissant la variable d'environnement R_TEXI2DVICMD sur l'émulation de valeur.

collection-fontsextra et collection-latexextra (en supposant qu'ils ne soient pas installés par défaut) : Fe-dora utilise des noms comme texlive-collection-fontsextra et Debian/Ubuntu comme texlive-fonts-extra.

Les programmes qpdf et ghostscript (gs) sont souhaitables car ils seront utilisés pour compacter le vignettes PDF installées et tous les manuels PDF.

Les programmes essentiels doivent être dans votre PATH au moment de l'exécution de configure : cela capturera les chemins complets.

Pour que les dates et heures fonctionnent correctement, il est essentiel que les tables définissant les fuseaux horaires soient installées : celles-ci se trouvent généralement dans un composant du système d'exploitation nommé quelque chose comme tzdata. Sur la plupart des systèmes d'exploitation, ils sont requis, mais des installations d'Alpine Linux ont été vues sans eux. À partir de R 4.3.0, il existe une vérification de configuration pour que les dates et heures récentes fonctionnent correctement dans différents fuseaux horaires, ce qui détecte cela lors de l'installation à partir des sources (mais pas pour les distributions binaires).

Ceux qui distribuent des versions binaires de R devront peut-être connaître les licences des bibliothèques externes auxquelles il est lié (y compris les bibliothèques « utiles » de la section suivante). La bibliothèque liblzma est dans le domaine public et X11, libbzip2, libcurl et zlib ont des licences de style MIT. PCRE et PCRE2 ont une licence de style BSD qui nécessite la distribution de la licence (incluse dans le fichier COPYRIGHTS de R) dans des distributions binaires. GNU readline est sous licence GPL (la ou les versions de GPL dépendent de la version de readline).

A.2 Bibliothèques et programmes utiles La possibilité d'utiliser

des messages traduits utilise gettext et nécessite très probablement GNU gettext : vous en avez besoin pour travailler avec de nouvelles traductions, mais sinon la version du runtime gettext contenue dans les sources R sera utilisée si aucun gettext externe approprié n'est trouvé.

La version « moderne » des périphériques graphiques X11(), jpeg(), png() et tiff() utilise les bibliothèques Cairo et Pango. La version Cairo 1.2.0 ou ultérieure et la version 1.10 ou ultérieure de Pango sont requises (mais des versions beaucoup plus récentes sont actuelles). R vérifie pkg-config et l'utilise pour vérifier d'abord que le package 'pangocairo' est installé (et sinon, 'cairo'), puis si le code approprié peut être compilé. Ces tests échoueront si pkg-config n'est pas installé[13], et pourraient échouer si cairo a été construit de manière statique à moins que l'option de configuration --with-static-cairo ne soit utilisée. La plupart des systèmes sur lesquels Gtk+ 2.8 ou version ultérieure est installé auront des bibliothèques adaptées : pour les utilisateurs de Fedora, le RPM pango-devel et ses dépendances suffisent. Il est possible (mais très inhabituel sur une plateforme avec X11) de construire Cairo sans son module cairo-xlib auquel cas X11(type = "cairo") ne sera pas disponible.
Pango est facultatif mais hautement souhaitable car il est susceptible d'offrir un bien meilleur rendu du texte, y compris le crénage.

Pour la meilleure expérience de police avec ces appareils, vous devez installer des polices appropriées : les utilisateurs Linux voudront le package urw-fonts. Sur les plateformes qui le disposent, le package msttcorefonts[14] fournit des versions TrueType des polices Monotype telles qu'Arial et Times New Roman.
Un autre ensemble de polices utiles sont les polices TrueType « libération » disponibles sur https://pagure.io/ qui couvrent libération. Celles-ci [15] les alphabets latin, grec et cyrillique ainsi qu'une bonne gamme de polices et de signes de partagent des métriques avec Arial, Times New Roman et Courier New, et contiennent des polices plutôt similaires aux deux premières (https://en.wikipedia.org/wiki/Liberation_fonts). Ensuite, il y a le projet « Free UCS Outline Fonts » (https://www.gnu.org/software/freefont/) qui sont des polices OpenType/TrueType basées sur les polices URW mais avec une couverture Unicode étendue. Consultez l'aide R sur X11 pour sélectionner de telles polices.

---

[13] Si nécessaire, le chemin vers pkg-config peut être spécifié en définissant PKG_CONFIG dans config.site, sur la ligne de commande configure ou dans l'environnement. Il existe une réimplémentation compatible de pkg-config appelée pkgconf qui peut être utilisée dans le cas peu probable où elle serait installée mais non liée à

[14] pkg-config. également connu sous le nom de ttf-mscorefonts-installer dans le monde Debian/Ubuntu : voir aussi https://en.wikipedia. org/wiki/

[15] Core_fonts_for_the_Web. libération ttf dans Debian/Ubuntu.

Les périphériques graphiques bitmap jpeg(), png() et tiff() nécessitent que les en-têtes et bibliothèques appropriés soient installés : jpeg (version 6b ou ultérieure, ou libjpeg-turbo) ou libpng (version 1.2.7 ou ultérieure) et zlib ou libtiff (les versions 4.0.[5-10] et 4.[123].0 ont été testées) respectivement. pkg-config est utilisé s'il est disponible et nécessite donc le fichier .pc approprié (qui nécessite la version 4.x de libtiff et n'est pas disponible sur toutes les plateformes pour jpeg avant la version 9c). Ils ont également besoin de la prise en charge de X11 ou du Caire (voir ci-dessus). Si la prise en charge de ces périphériques n'est pas requise ou si des bibliothèques système cassées doivent être évitées, il existe des options de configuration --without-libpng, --without-jpeglib et --without-libtiff. La bibliothèque TIFF possède de nombreuses fonctionnalités optionnelles telles que jpeg, libz, zstd, lzma, webp, jbig et jpeg12, dont aucune n'est requise pour les périphériques tiff() mais peuvent devoir être présentes pour lier la bibliothèque (généralement un problème uniquement pour liaison statique). pkg-config peut vous indiquer quelles autres bibliothèques sont requises pour la liaison, par exemple par pkg-config libtiff-4 --static --libs.

L'option --with-system-tre est également disponible : elle nécessite une version récente de TRE. (Les dernières sources se trouvent dans le référentiel git à l' adresse https://github.com/laurikari/tre/, mais au moment de la rédaction de cet article, la version résultante n'a pas terminé ses vérifications, et R n'a pas non plus été construit avec la version fournie par Fedora.)

Une implémentation de XDR est requise, et les sources R en contiennent une qui suffira probablement (bien qu'une version système puisse avoir des performances plus élevées). XDR fait partie de RPC et a historiquement fait partie de la libc sous Unix. (En principe, man xdr_string devrait vous indiquer quelle bibliothèque est nécessaire, mais ce n'est souvent pas le cas : sur certains systèmes d'exploitation, elle est fournie par libnsl.) Cependant, certaines versions[16] de la glibc l'omettent ou la cachent dans l'intention d' utiliser la bibliothèque TI-RPC , auquel cas libtirpc (et sa version de développement) doit être installé, et ses en-têtes[17] doivent être sur le chemin d'inclusion C ou sous /usr/include/tirpc.

L'utilisation de la sélection du presse-papiers X11 nécessite les en-têtes et bibliothèques Xmu. Ceux-ci font normalement partie d'une installation X11 (par exemple le méta-paquet Debian 'xorg-dev'), mais certaines distributions l'ont divisé en parties plus petites, ainsi par exemple les versions récentes de Fedora nécessitent 'libXmu' et 'libXmu-devel'. RPM.

Certains systèmes (notamment macOS et au moins certains systèmes FreeBSD) n'ont pas une prise en charge inadéquate du classement dans les paramètres régionaux multi-octets. Il est possible de remplacer la prise en charge du classement du système d'exploitation par celle d'ICU (International Components for Unicode, https://icu.unicode.org/), ce qui permet un contrôle beaucoup plus précis du classement sur tous les systèmes. ICU est disponible sous forme de sources et de distributions binaires pour (au moins) la plupart des distributions Linux, FreeBSD, macOS et AIX, généralement sous le nom de libicu ou icu4c. Il sera utilisé par défaut lorsqu'il est disponible : si une version très ancienne ou cassée d'ICU est trouvée, elle peut être supprimée par --without-ICU.

Les périphériques bitmap et dev2bitmap et la fonction embedFonts() utilisent ghostscript (https://www.ghostscript.com/). Cela doit être soit dans votre chemin lorsque la commande est exécutée, soit dans son chemin complet spécifié par la variable d'environnement R_GSCMD à ce moment-là.

Au moment de la rédaction d'une installation complète sur Fedora Linux, les packages suivants étaient utilisés et leurs versions de développement, et cela peut fournir une liste de contrôle utile pour d'autres systèmes :

bzip2 caire fontconfig freetype fribidi gcc gcc-gfortran gcc-c++ glib2 glibc harfbuzz libX11 libXext libXt libcurl libicu libjpeg libpng libtiff libtirpc libxcrypt ncurses pango pkgconf-pkg-config pcre2 readline tcl tk xz zlib

plus, de préférence une installation TeX, Java et LAPACK >= 3.10.0.

A.2.1 Tcl/Tk Le

package tcltk nécessite que Tcl/Tk ≥ 8.4 soit installé : les sources sont disponibles sur https://www.tcl. merci/. Pour spécifier les emplacements des fichiers Tcl/Tk, vous aurez peut-être besoin des options de configuration

[16] Y compris celui utilisé par Fedora.

[17] R utilise rpc/xdr.h mais cela inclut netconfig.h du répertoire tirpc supérieur.

--avec-tcltk

utilisez Tcl/Tk, ou spécifiez son répertoire de bibliothèque

--with-tcl-config=TCL_CONFIG

spécifier l'emplacement de tclConfig.sh

--with-tk-config=TK_CONFIG

spécifier l'emplacement de tkConfig.sh

ou utilisez les variables de configuration TCLTK_LIBS et TCLTK_CPPFLAGS pour spécifier les indicateurs nécessaires à la liaison avec les bibliothèques Tcl et Tk et pour trouver les en-têtes tcl.h et tk.h, respectivement.
Si vous avez installé les versions 32 et 64 bits de Tcl/Tk, il peut être nécessaire de spécifier les chemins d'accès aux fichiers de configuration corrects pour éviter toute confusion entre eux.

Les versions de Tcl/Tk jusqu'à 8.5.19 et 8.6.12 ont été testées (y compris la plupart des versions de 8.4.x, mais pas récemment).

Notez que l'en-tête tk.h comprend 18 en-têtes X11, vous aurez donc besoin de X11 et de ses fichiers de développement installés.

A.2.2 Prise en charge de Java

Le processus de construction recherche la prise en charge de Java sur le système hôte et, s'il le trouve, définit certains paramètres utiles pour les packages utilisant Java (tels que rJava (https://CRAN.R-project.org/ package=rJava) et JavaGD (https://CRAN.R-project.org/package=JavaGD) : ceux-ci nécessitent un JDK complet). Cette vérification peut être supprimée par l'option de configuration --disable-java. La variable de configuration JAVA_HOME peut être définie pour pointer vers un JRE/JDK spécifique, sur la ligne de commande de configuration ou dans l'environnement.

Parmi ces paramètres, les principaux sont certains chemins vers les bibliothèques Java et JVM, qui sont stockés dans la variable d'environnement R_JAVA_LD_LIBRARY_PATH dans le fichier R_HOME/etc/ldpaths (ou une version spécifique à une sous-architecture). Un paramètre typique pour Linux 'x86_64' est

JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.322.b06-6.fc34.x86_64/jre R_JAVA_LD_LIBRARY_PATH=${JAVA_HOME}/lib/amd64/ serveur

Malheureusement, cela dépend de la version exacte du JRE/JDK installé et peut donc nécessiter une mise à jour si l'installation Java est mise à jour. Cela peut être fait en exécutant R CMD javareconf qui met à jour les paramètres dans R_HOME/etc/Makeconf et R_HOME/etc/ldpaths. Voir R CMD javareconf --help pour plus de détails : notez que cela doit être fait par le compte propriétaire de l'installation R.

Une autre façon de remplacer ces paramètres consiste à définir la variable d'environnement R_JAVA_LD_LIBRARY_PATH (avant le démarrage de R, donc pas dans ~/.Renviron), ce qui suffit pour exécuter des packages utilisant Java déjà installés. Par exemple

R_JAVA_LD_LIBRARY_PATH=/usr/lib/jvm/java-1.8.0/jre/lib/amd64/server Il peut être possible d'éviter cela en spécifiant un lien invariant comme chemin lors de la configuration.
Par exemple, sur ce système, l'un des

JAVA_HOME=/usr/lib/jvm/java
JAVA_HOME=/usr/lib/jvm/java-1.8.0 JAVA_HOME=/usr/lib/jvm/java-1.8.0/jre JAVA_HOME=/usr/lib/jvm/jre-1.8.0

a fonctionné (puisque le paramètre 'auto' de /etc/alternatives a choisi Java 8 alias 1.8.0).

Les distributions Oracle 'non-serveur' de Java à partir de la version 11 sont d'un JDK complet. Cependant, Linux les distributions peuvent prêter à confusion : par exemple Fedora 34 avait

java-1.8.0-openjdk

[18] Cela est vrai même pour la version « Aqua » de Tk sur macOS, mais les distributions de celle-ci incluent une copie des fichiers X11 nécessaires.

java-1.8.0-openjdk-devel java-
openjdk java-
openjdk-devel java-11-
openjdk java-11-
openjdk-devel java-17-openjdk
java-17-openjdk-devel
java-latest-openjdk java-latest-
openjdk-développement

où les RPM -devel sont nécessaires pour compléter le JDK. Debian/Ubuntu utilise '-jre' et '-jdk',

par exemple

sudo apt install default-jdk

A.2.3 Autres langages compilés Certains packages

complémentaires nécessitent un compilateur C++. Ceci est spécifié par les variables de configuration CXX, CXXFLAGS et similaires. configure trouvera normalement un compilateur approprié. Il est possible de spécifier un compilateur C++17 alternatif par les variables de configuration CXX17, CXX17STD, CXX17FLAGS et similaires (voir Section 2.7.3 [Support C++], page 11). Encore une fois, configure trouvera normalement une valeur appropriée pour CXX17STD si le compilateur fourni par CXX est capable de compiler du code C++17, mais il est possible qu'un compilateur complètement différent soit nécessaire. (Des macros similaires sont fournies pour C++20.)

Pour les fichiers sources d'extension .f90 ou .f95 contenant du Fortran de forme libre, le compilateur défini par la macro FC est utilisé par R CMD INSTALL. Notez qu'il est détecté par le nom de la commande sans test qu'elle peut réellement compiler du code Fortran 90. Définissez la variable de configuration FC pour la remplacer si nécessaire : les variables FCFLAGS et FCLIBS_XTRA peuvent également devoir être définies.

Voir le fichier config.site dans la source R pour plus de détails sur ces variables.

A.3 Algèbre linéaire Les routines

d'algèbre linéaire dans R utilisent des routines BLAS (Basic Linear Algebra Subprograms, https://netlib.org/blas/faq.html) , et la plupart utilisent des routines de LAPACK (Linear Algebra PACKage, https://netlib.org/lapack/). Les sources R contiennent des implémentations de référence (Fortran), mais elles peuvent être remplacées par des bibliothèques externes, généralement celles optimisées pour la vitesse sur des processeurs spécifiques. Ces bibliothèques contiennent normalement toutes les routines BLAS et certaines routines LAPACK optimisées et peut-être le reste de LAPACK de l'implémentation de référence. En raison du fonctionnement des liens, l'utilisation d'une bibliothèque BLAS externe peut nécessiter l'utilisation de la version de LAPACK qu'elle contient.

Notez que les implémentations alternatives ne donneront pas des résultats numériques identiques. Certaines différences peuvent être bénignes (comme les signes de SVD et de vecteurs propres), mais les routines optimisées peuvent être moins précises et (en particulier pour LAPACK) peuvent provenir d'anciennes versions avec moins de corrections. Cependant, R s'appuie sur la conformité ISO/IEC 60559. Cela peut être rompu si, par exemple, le code suppose que les termes avec un facteur zéro sont toujours nuls et n'ont pas besoin d'être calculés, alors que x*0 peut être NaN. Le BLAS interne a été largement corrigé pour éviter cela alors que la documentation de MKL a prévenu

> Les routines LAPACK supposent que les matrices d'entrée ne contiennent pas de valeurs spéciales IEEE 754 telles que les valeurs INF ou NaN. L'utilisation de ces valeurs spéciales peut amener LAPACK à renvoyer des résultats inattendus ou à devenir instable.

Certaines bibliothèques externes sont multithread. Un problème est que le profilage R (qui utilise le signal SIGPROF) peut causer des problèmes et vous souhaiterez peut-être désactiver le profilage si vous utilisez un BLAS multithread . Notez que l'utilisation d'un BLAS multithread peut nécessiter plus de CPU

temps et encore plus de temps écoulé (parfois de façon spectaculaire) que l'utilisation d'un BLAS similaire à un seul thread. Sur une machine exécutant d'autres tâches, il peut y avoir des conflits pour les caches CPU qui réduisent l'efficacité de l'optimisation de l'utilisation du cache par une implémentation BLAS : certaines personnes préviennent que cela est particulièrement problématique pour les CPU hyperthreadés.

Les routines BLAS et LAPACK peuvent être utilisées dans du code threadé, par exemple dans les sections OpenMP de packages tels que mgcv (https://CRAN.R-project.org/package=mgcv). Les implémentations de référence sont thread-safe, mais les implémentations externes peuvent ne pas l'être (même celles à thread unique) : cela peut conduire à des résultats incorrects ou à des erreurs de segmentation difficiles à retrouver.

Il existe une tendance chez les redistributeurs de R à utiliser des bibliothèques d'algèbre linéaire « améliorées » sans expliquer leurs inconvénients.

A.3.1 BLAS

Une bibliothèque BLAS externe doit être explicitement demandée au moment de la configuration.

Vous pouvez spécifier une bibliothèque BLAS particulière via une valeur pour l'option de configuration --with-blas. Si celui-ci est donné sans =, sa valeur est extraite de la variable d'environnement BLAS_LIBS, définie par exemple dans config.site. Si ni l'option ni la variable d'environnement ne fournissent de valeur, une recherche est effectuée pour un BLAS[19] approprié. Si la valeur n'est pas évidemment une commande de l'éditeur de liens (commençant par un tiret ou donnant le chemin d'une bibliothèque), elle est préfixée par '-l', donc

```
--with-blas="foo"
```

est une instruction pour établir un lien avec '-lfoo' pour trouver un BLAS externe (qui doit être trouvé à la fois au moment de la liaison et au moment de l'exécution).

Le code de configuration vérifie que le BLAS externe est complet (il doit inclure toutes les routines double précision et double complexe, ainsi que LSAME) et semble utilisable. Cependant, un BLAS externe doit être utilisable à partir d'un objet partagé (il doit donc contenir du code indépendant de la position), et cela n'est pas vérifié.

Certains BLAS améliorés sont spécifiques au système de compilateur (Accelerate sur macOS, sunperf sur Solaris20[20], libessl sur IBM). L'incantation correcte pour ceux-ci est souvent trouvée via --with-blas sans valeur sur les plateformes appropriées.

Notez que sous Unix (mais pas sous Windows), si R est compilé avec un BLAS autre que celui par défaut et que --enable-BLAS-shlib n'est pas utilisé (c'est la valeur par défaut sur toutes les plateformes sauf AIX), alors tous les packages utilisant BLAS doivent être aussi. Ainsi, si R est reconstruit pour utiliser un BLAS amélioré , des packages tels que quantreg (https://CRAN.R-project.org/package=quantreg) devront être réinstallés.

Les systèmes Debian/Ubuntu fournissent un moyen spécifique au système pour changer le BLAS utilisé : construisez R avec -with-blas pour sélectionner la version du système d'exploitation du BLAS de référence, puis utilisez des alternatives de mise à jour pour basculer entre les bibliothèques BLAS disponibles. Voir https://wiki. debian.org/DebianScience/LinearAlgebraLibraries.

Fedora 33 et versions ultérieures proposent « FlexiBLAS », un mécanisme similaire pour changer de BLAS utilisé (https://www.mpi-magdeburg.mpg.de/projects/flexiblas). Cependant, plutôt que de remplacer libblas, cela nécessite de configurer R avec l'option --with-blas=flexiblas. Des wrappers « Backend » sont disponibles pour les versions de référence BLAS, ATLAS et série, threadées et OpenMP d'OpenBLAS et BLIS. Cela peut être contrôlé à partir d'une session R en cours d'exécution par le package flexiblas (https://CRAN.R-project.org/package=flexiblas). Apparemment non documenté : FlexiBLAS sur Fedora fournit un LAPACK complet, mais pas les routines améliorées d'ATLAS ou d'OpenBLAS.

Les implémentations BLAS qui utilisent des calculs parallèles peuvent être non déterministes : c'est connu pour ATLAS.

[19] L'ordre de recherche est actuellement OpenBLAS, BLIS, ATLAS, des choix spécifiques à la plateforme (voir ci-dessous) et enfin une libblas générique.

[20] Utilisation des compilateurs Oracle Developer Studio cc et f95

A.3.1.1 ATLAS

ATLAS (https://math-atlas.sourceforge.net/) est un BLAS « optimisé » qui fonctionne sur une large gamme de plates-formes de type Unix. Malheureusement, il est construit par défaut en tant que bibliothèque statique qui, sur certaines plates-formes, peut ne pas pouvoir être utilisée avec des objets partagés tels que ceux utilisés dans les packages R.
Soyez prudent lorsque vous utilisez des versions prédéfinies des bibliothèques statiques ATLAS (elles semblent fonctionner sur les plates-formes « ix86 », mais pas toujours sur celles « x86_64 »).

ATLAS contient des remplacements pour un petit nombre de routines LAPACK, mais peut être construit pour fusionnez-les avec les sources LAPACK de référence pour inclure une bibliothèque LAPACK complète.

Les versions récentes d'ATLAS peuvent être construites comme une seule bibliothèque partagée, soit libsatlas, soit libtatlas (respectivement en série ou threadée) : celles-ci peuvent même contenir un LAPACK complet. De telles versions peuvent être utilisées par l'un des

--with-blas=satlas
--with-blas=tatlas

ou, comme sur Fedora 'x86_64' où un chemin doit être spécifié, --with-blas="-L/
usr/lib64/atlas -lsatlas" --with-blas="-L/usr/lib64/atlas - Atlas"

Les bibliothèques ATLAS distribuées ne peuvent pas être adaptées à votre machine et constituent donc un compromis : par exemple, les RPM Fedora tunes21 'x86_64' pour les processeurs avec extensions SSE3, et des RPM séparés peuvent être disponibles pour des familles de processeurs spécifiques.

Notez que la construction de R sur Linux avec des bibliothèques partagées distribuées peut nécessiter l'installation des packages « -devel » ou « -dev ».

La liaison avec plusieurs bibliothèques statiques nécessite l'un des

--with-blas="-lf77blas -latlas"
--with-blas="-lptf77blas -lpthread -latlas" --with-blas="-L/path/to/
ATLAS/libs -lf77blas -latlas" --with-blas="-L/path/to/ ATLAS/libs -lptf77blas -lpthread
-latlas"

Consultez son guide d'installation22 pour savoir comment construire ATLAS en tant que bibliothèque partagée ou en tant que bibliothèque statique.
bibliothèque avec du code indépendant de la position (sur les plateformes où cela compte).

Selon la FAQ ATLAS23, le nombre maximum de threads utilisés par ATLAS multithread est défini au moment de la compilation. En outre, l'auteur déconseille d'utiliser ATLAS multithread sur des processeurs hyperthread sans restreindre les affinités au moment de la compilation à un cœur virtuel par processeur physique. (Pour les bibliothèques Fedora, l'indicateur de compilation spécifie 4 threads.)

A.3.1.2 OpenBLAS et BLIS Le Dr Kazushige Goto a

écrit un BLAS optimisé pour plusieurs processeurs et systèmes d'exploitation, qui a été gelé en 2010. OpenBLAS (https://www.openblas.net/) est un projet descendant prenant en charge certains processeurs ultérieurs. .

Cela peut être utilisé en configurant R avec quelque chose comme

--with-blas="openblas"

Voir la Section A.3.1.4 [Shared BLAS], page 48, pour une manière alternative (et à bien des égards préférable) de les utiliser.

Certaines plates-formes fournissent plusieurs versions d'OpenBLAS : par exemple Fedora 34 a des RPM24

flamber

---

21 La seule façon de voir exactement pour quels processeurs les bibliothèques distribuées ont été optimisées est de lire le fichier atlas.spec.

22 https://math-atlas.sourceforge.net/atlas_install/ https://math-

23 atlas.sourceforge.net/faq.html#tnum (et plus, par exemple pour les

24 entiers 64 bits et les versions statiques).

openblas-threads
openblas-openmp

fournissant des bibliothèques partagées

libopenblas.so
libopenblasp.so
libopenblaso.so

respectivement, chacun pouvant être utilisé comme un BLAS partagé. Pour les deuxième et troisième, le nombre de threads est contrôlé respectivement par OPENBLAS_NUM_THREADS et OMP_NUM_THREADS (comme d'habitude pour OpenMP).

Ceux-ci et leurs équivalents Debian contiennent une implémentation complète de LAPACK.

Notez que la construction de R sous Linux avec des bibliothèques distribuées peut nécessiter « -devel » ou « -dev » paquets installés.

Pour les processeurs 'ix86' et 'x86_64', les bibliothèques les plus distribuées contiennent plusieurs alternatives pour différentes microarchitectures de processeur, le choix étant effectué au moment de l'exécution.

Un autre projet descendant est BLIS (https://github.com/flame/blis). Cela a (en Fedora) bibliothèques partagées

libblis.so
libbliso.so

(p pour 'threads', o pour OpenMP comme pour OpenBLAS) qui peut également être utilisé comme BLAS partagé. Les versions Fedora n'incluent pas LAPACK dans les bibliothèques BLIS.

A.3.1.3 Intel MKL

Pour les processeurs Intel (et peut-être d'autres) et certaines distributions de Linux, il existe la Math Kernel Library d'Intel[25]. Nous vous encourageons à lire la documentation installée avec la bibliothèque avant d'essayer de créer un lien vers MKL. Celui-ci inclut un « conseiller en ligne de lien » qui suggérera les incantations appropriées : son utilisation est recommandée. Ou consultez https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-link-line-advisor.html#gs.vpt6qp .

Il existe également des versions de MKL pour macOS[26] et Windows, mais une fois celles-ci essayées ils ne fonctionnaient pas avec les compilateurs par défaut utilisés pour R sur ces plateformes.

L'interface MKL a changé plusieurs fois mais est restée stable ces dernières années : les exemples suivants ont été utilisés avec les versions 10.3 à 2023.0.0, pour les compilateurs GCC sur CPU 'x86_64'.

Pour utiliser une version séquentielle de MKL, nous avons utilisé

```
MKL_LIB_PATH=/path/to/intel_mkl/mkl/lib/intel64 export
LD_LIBRARY_PATH=$MKL_LIB_PATH MKL="-L$
{MKL_LIB_PATH} -lmkl_gf_lp64 -lmkl_core -lmkl_sequential" ./configure --with-blas="$MKL" --
 avec-lapack
```

L'option --with-lapack est utilisée puisque MKL contient une copie optimisée de LAPACK (souvent plus ancienne que la version actuelle) ainsi que du BLAS (voir Section A.3.2 [LAPACK], page 49), bien que cela puisse être omis.

Le MKL fileté peut être utilisé en remplaçant la ligne définissant la variable MKL par

```
MKL="-L${MKL_LIB_PATH} -lmkl_gf_lp64 -lmkl_core \ -lmkl_gnu_thread
        -dl -fopenmp"
```

[25] Parfois connue sous le nom de « Bibliothèque Intel oneAPI Math Kernel » ou même « oneMKL ».

[26] Le problème pour macOS réside dans l'utilisation de routines doublement complexes.

R peut également être lié à une seule bibliothèque partagée, libmkl_rt.so, pour BLAS et LA-PACK, mais la bonne couche d'interface OpenMP et MKL doit ensuite être sélectionnée via des variables d'environnement. Avec les versions 64 bits et les compilateurs GCC, nous avons utilisé

exporter MKL_INTERFACE_LAYER=GNU,LP64
exporter MKL_THREADING_LAYER=GNU

Sur Debian/Ubuntu, MKL est fourni par le paquet intel-mkl-full et on peut définir libmkl_rt.so comme implémentation à l'échelle du système de BLAS et LAPACK lors de l'installation du paquet, de sorte que R soit également installé à partir du paquet Debian/Ubuntu. r-base l'utiliserait. Il est cependant toujours essentiel de définir MKL_INTERFACE_LAYER et MKL_THREADING_LAYER avant d'exécuter R, sinon les calculs MKL produiront des résultats incorrects. R n'a pas besoin d'être reconstruit pour utiliser MKL, mais configurez des tests d'inclusion qui peuvent découvrir certaines erreurs telles qu'un échec de définition de la bonne couche d'interface OpenMP et MKL.

Le nombre de threads par défaut sera choisi par le logiciel OpenMP, mais peut être contrôlé en définissant OMP_NUM_THREADS ou MKL_NUM_THREADS, et dans les versions récentes, il semble être par défaut une valeur raisonnable pour une utilisation exclusive de la machine. (Parallel MKL n'a pas toujours réussi make check-all, mais l'a fait avec MKL 2019.4 et versions ultérieures.)

MKL inclut une implémentation partielle de FFTW3, ce qui provoque des problèmes pour les applications nécessitant certaines fonctionnalités de FFTW3 non prises en charge dans MKL. Veuillez consulter les manuels MKL pour la description de ces limitations et pour obtenir des instructions sur la façon de créer une version personnalisée de MKL qui exclut les wrappers FFTW3.

A.3.1.4 BLAS partagé

La bibliothèque BLAS sera utilisée pour de nombreux packages complémentaires ainsi que pour R lui-même. Cela signifie qu'il est préférable d'utiliser une bibliothèque BLAS partagée/dynamique , car la plupart d'une bibliothèque statique sera compilée dans l'exécutable R et dans chaque package utilisant BLAS.

R offre la possibilité de compiler le BLAS dans une bibliothèque dynamique libRblas stockée dans R_HOME/lib et reliant R lui-même et tous les packages complémentaires à cette bibliothèque.

C'est la valeur par défaut sur toutes les plateformes sauf AIX à moins qu'un BLAS externe ne soit spécifié et trouvé : pour ce dernier, il peut être utilisé en spécifiant l'option --enable-BLAS-shlib, et il peut toujours être désactivé via --disable-BLAS-chlib.

Cela présente à la fois des avantages et des inconvénients.

- Il économise de l'espace en n'ayant qu'une seule copie des routines BLAS , ce qui est utile s'il y a est un BLAS statique externe (comme c'était le cas pour ATLAS).
- L'utilisation d'un BLAS partagé peut entraîner des inconvénients en termes de performances . Le plus probable est probablement lorsque le BLAS interne de R est utilisé et que R n'est pas construit en tant que bibliothèque partagée, lorsqu'il est possible de créer le BLAS dans R.bin (et libR.a) sans utiliser de code indépendant de la position. Cependant, des expériences ont montré que dans de nombreux cas, l'utilisation d'un BLAS partagé était aussi rapide, à condition d'utiliser des niveaux élevés d'optimisation du compilateur.
- Il est facile de changer le BLAS sans avoir besoin de réinstaller R et tous les packages complémentaires, puisque toutes les références au BLAS passent par libRblas, et cela peut être remplacé. Notez cependant que toutes les bibliothèques dynamiques vers lesquelles les liens de remplacement devront être trouvées par l'éditeur de liens : cela peut nécessiter que le chemin de la bibliothèque soit modifié dans R_HOME/etc/ldpaths.

Une autre option pour modifier le BLAS utilisé consiste à créer un lien symbolique entre une seule bibliothèque BLAS dynamique et R_HOME/lib/libRblas.so. Par exemple, juste

mv R_HOME/lib/libRblas.so R_HOME/lib/libRblas.so.keep ln -s /usr/lib64/
libopenblasp.so.0 R_HOME/lib/libRblas.so sur 'x86_64' Fedora changera le BLAS utilisé

en OpenBLAS multithread . Un lien similaire fonctionne pour la plupart des versions d'OpenBLAS (à condition que le répertoire lib approprié se trouve dans le

chemin de la bibliothèque d'exécution ou cache ld.so). Il peut également être utilisé pour un ATLAS à bibliothèque unique, donc sur Fedora 'x86_64', soit

ln -s /usr/lib64/atlas/libsatlas.so.3 R_HOME/lib/libRblas.so ln -s /usr/lib64/atlas/libtatlas.so.3 R_HOME/lib/libRblas.so peut être utilisé avec son Bibliothèques ATLAS. (Si vous avez

installé le RPMS '-devel', vous pouvez omettre le .0/.3.)

Notez que reconstruire ou créer un lien symbolique avec libRblas.so peut ne pas suffire si l'intention est d'utiliser un LAPACK modifié contenu dans un BLAS externe : ce dernier pourrait même provoquer des conflits.

Cependant, sur Fedora où la distribution OpenBLAS contient une copie de LAPACK, c'est ce dernier qui est utilisé.

A.3.2 LAPACK

Si lors de la configuration de R, une bibliothèque système LAPACK de version 3.10.0 ou ultérieure est trouvée (et ne contient pas de routines BLAS), elle sera utilisée au lieu de compiler le code LAPACK dans les sources du package. Cela peut être évité en configurant R avec --without-lapack. L'utilisation d'un liblapack.a statique n'est pas prise en charge.

On suppose que -llapack est la bibliothèque LAPACK de référence mais sur Debian/Ubuntu elle peut être changée, y compris après l'installation de R. Sur une telle plate-forme, il est préférable d'utiliser explicitement --without-lapack ou ---with-blas--with-lapack (voir ci-dessous). Les exemples connus27 d'une bibliothèque LAPACK sans référence trouvée lors de l'installation contiennent tous des routines BLAS et ne sont donc pas utilisés par une exécution de configuration par défaut.

Il est prévu de spécifier une bibliothèque LAPACK externe avec l'option --with-lapack, principalement pour gérer les bibliothèques BLAS qui contiennent une copie de LAPACK (telles que Accelerate sur macOS et certaines versions d'ATLAS, FlexiBLAS, MKL et OpenBLAS sur 'ix86' /'x86_64' Linux). Au moins LAPACK version 3.2 est requis. Cela ne peut être fait que si --with-blas a
déjà utilisé.

Cependant, les gains de performance probables sont considérés comme faibles (et pourraient être négatifs). La valeur par défaut n'est pas de rechercher une bibliothèque LAPACK appropriée, ce qui n'est absolument pas recommandé. Vous pouvez spécifier une bibliothèque LAPACK spécifique ou rechercher une bibliothèque générique par l'option de configuration --with-lapack sans valeur. La valeur par défaut pour --with-lapack est de vérifier la bibliothèque BLAS (pour la fonction DPSTRF), puis de rechercher une bibliothèque externe '-llapack'. Les sites recherchant l'algèbre linéaire la plus rapide possible voudront peut-être créer une bibliothèque LAPACK en utilisant le sous-ensemble de LAPACK optimisé pour ATLAS. De même, OpenBLAS peut être construit pour contenir un sous-ensemble optimisé de LAPACK ou un LAPACK complet (ce dernier semblant être la valeur par défaut).

Une valeur pour --with-lapack peut être définie via la variable d'environnement LAPACK_LIBS, mais elle ne sera utilisée que si --with-lapack est spécifié et que la bibliothèque BLAS ne contient pas LAPACK.

Veuillez garder à l'esprit que l'utilisation de --with-lapack est fournie uniquement parce qu'elle est nécessaire sur certaines plates-formes et parce que certains utilisateurs souhaitent expérimenter les améliorations de performances revendiquées. En pratique, ses principales utilisations sont sans valeur,

- avec un BLAS « amélioré » tel que ATLAS, FlexiBLAS, MKL ou OpenBLAS qui contient un LAPACK complet (pour éviter d'éventuels conflits), ou
- sur les systèmes Debian/Ubuntu pour sélectionner le liblapack système qui peut être commuté par le mécanisme des « alternatives ».

A.3.3 Mises en garde

Comme pour toutes les bibliothèques, vous devez vous assurer qu'elles et R ont été compilés avec des compilateurs et des indicateurs compatibles. Par exemple, cela signifie que sur Sun Sparc utilisant les compilateurs Oracle, l'indicateur -dalign est nécessaire si sunperf doit être utilisé.

---

[27] ATLAS, OpenBLAS et Accélération.

Sur certains systèmes, il a été nécessaire de construire un BLAS/LAPACK externe avec le même compilateur Fortran utilisé pour construire R.

Les bibliothèques BLAS et LAPACK construites avec les versions récentes de gfortran nécessitent des appels de C/C++ pour gérer les longueurs de caractères « cachées » — R lui-même le fait, mais de nombreux packages ne le faisaient pas et certains ont des erreurs de segmentation. Ceci a été largement contourné en utilisant l'indicateur Fortran -fno-optimize-sibling-calls (anciennement défini par configure s'il détectait gfortran 7 ou version ultérieure) : cependant, l'utilisation des en-têtes R qui incluent ces arguments de longueur de caractère n'est plus facultative dans les packages. .

LAPACK 3.9.0 (et probablement plus tôt) avait un bug dans lequel le sous-programme DCOMBSSQ pouvait faire interpréter NA comme zéro. Ceci est corrigé dans les sources R 3.6.3 et ultérieures, mais si vous utilisez un LAPACK externe, vous devrez peut-être le corriger ici. (Le bug a été corrigé dans la version 3.9.1 et la routine supprimée dans la version 3.10.1.)

Le code (dans dlapack.f) devrait se lire

```
*          ..
*          .. Déclarations exécutables ..
*
           SI( V1( 1 ).GE.V2( 1 ) ) ALORS
              SI( V1( 1 ).NE.ZERO ) ALORS
                 V1( 2 ) = V1( 2 ) + ( V2( 1 ) / V1( 1 ) )**2 * V2( 2 )
              AUTRE
                 V1(2) = V1(2) + V2(2)
              FIN SI
           AUTRE
              V1( 2 ) = V2( 2 ) + ( V1( 1 ) / V2( 1 ) )**2 * V1( 2 )
              V1(1) = V2(1)
           FIN SI
           RETOUR
```

(La clause ELSE interne manquait dans LAPACK 3.9.0.)

Si vous utilisez un LAPACK externe, soyez conscient des problèmes potentiels liés à d'autres bogues dans les sources LAPACK (ou dans les corrections publiées sur ces sources), observés à plusieurs reprises dans les distributions Linux au fil des ans. Nous avons même vu des distributions avec des routines LAPACK manquantes dans leur liblapack.

Nous comptons sur un support limité dans LAPACK pour les matrices de 231 éléments ou plus : il est possible qu'un LAPACK externe n'ait pas ce support.

Annexe B Configuration sous Unix

B.1 Options de configuration configure propose de

nombreuses options : en cours d'exécution

```
./configure --help
```

donnera une liste. Les plus importants non traités ailleurs sont probablement (valeurs par défaut entre parenthèses) --with-x

utilise le système X Window [oui]

--x-includes=DIR

Les fichiers d'inclusion X sont dans DIR

--x-bibliothèques=DIR

Les fichiers de la bibliothèque X sont dans DIR

--avec-readline

utiliser la bibliothèque readline (si disponible) [oui]

--enable-R-profiling tentative de

compilation du support pour Rprof() [oui]

--activer le profilage de la mémoire

tenter de compiler le support de Rprofmem() et tracemem() [non]

--enable-R-shlib

construire R en tant que bibliothèque partagée/dynamique [non]

--enable-BLAS-shlib

construire le BLAS en tant que bibliothèque partagée/dynamique [oui, sauf sur AIX]

Vous pouvez utiliser --without-foo ou --disable-foo pour les négatifs.

Vous souhaiterez utiliser --disable-R-profiling si vous créez un exécutable profilé de R (par exemple avec '-pg)'. La prise en charge du profilage R nécessite la prise en charge du système d'exploitation pour les threads POSIX (alias « pthreads »), qui sont disponibles sur toutes les plates-formes Unix grand public.

Flag --enable-R-shlib oblige le processus make à construire R en tant que bibliothèque dynamique (partagée), généralement appelée libR.so, et à lier l'exécutable principal de R, R.bin, à cette bibliothèque. Cela ne peut être fait que si tout le code (y compris les bibliothèques système) peut être compilé dans une bibliothèque dynamique, et il peut y avoir une pénalité en termes de performances1. Donc, vous ne le souhaitez probablement que si vous utilisez une application qui intègre R. Notez que le code C dans les packages installés sur un système R lié avec --enable-R-shlib est lié à la bibliothèque dynamique et que de tels packages ne peuvent donc pas être utilisés. à partir d'un système R construit de la manière par défaut. De plus, comme les packages sont liés à R, ils le sont sur certains systèmes d'exploitation également liés aux bibliothèques dynamiques auxquelles R lui-même est lié, ce qui peut entraîner des conflits de symboles.

Pour une utilisation maximale efficace de valgrind, R doit être compilé avec l'instrumentation valgrind.
L'option de configuration est --with-valgrind-instrumentation=level, où le niveau est 0, 1 ou 2.
(Le niveau 0 est le niveau par défaut et n'ajoute rien.) Les en-têtes système pour valgrind seront utilisés s'ils sont présents (sous Linux, ils peuvent se trouver dans un package séparé tel que valgrind-devel). Notez cependant qu'il n'y a aucune garantie que le code dans R sera compatible avec les très anciens en-têtes valgrind2 ou futurs.

Si vous devez reconfigurer R avec différentes options, vous devrez peut-être exécuter make clean ou même faites distclean avant de le faire.

Le script configure possède d'autres options génériques ajoutées par autoconf et qui ne sont pas prises en charge. porté pour R : en particulier, la construction d'une architecture sur un hôte différent n'est pas possible.

---

[1] Nous avons mesuré 15 à 20 % sur Linux « i686 » et environ 10 % sur Linux « x86_64 ».

[2] Nous pensons que les versions 3.4.0 à 3.19.0 sont compatibles.

B.2 Prise en charge de l'internationalisation

La traduction des messages est prise en charge via GNU gettext, sauf si elle est désactivée par l'option de configuration --disable-nls. Le rapport de configuration affichera NLS comme l'une des « fonctionnalités supplémentaires » si la prise en charge a été compilée, et l'exécution dans une langue anglaise (mais pas la langue C) inclura

Prise en charge du langage naturel mais fonctionnant dans une langue anglaise

dans le message d'accueil au démarrage de R.

B.3 Variables de configuration Si vous avez besoin ou

souhaitez définir certaines variables de configuration sur autre chose que leur valeur par défaut, vous pouvez le faire soit en éditant le fichier config.site (qui documente la plupart des variables que vous pourriez vouloir définir : d'autres peuvent être vu dans le fichier etc/Renviron.in) ou sur la ligne de commande comme

./configure VAR=valeur

Si vous construisez dans un répertoire différent des sources, il peut y avoir des copies de config.site dans les répertoires source et build, et les deux seront lus (dans cet ordre). De plus, s'il existe un fichier ~/.R/config, il est lu entre les fichiers config.site dans les répertoires source et build.

Il existe également un mécanisme général d'autoconf pour les fichiers config.site, qui sont lus avant chacun de ceux mentionnés dans le paragraphe précédent. Cela examine d'abord un fichier spécifié par la variable d'environnement CONFIG_SITE, et sinon, est défini sur des fichiers tels que /usr/local/share/config.site et /usr/local/etc/config.site dans la zone (exemple par /usr/local) où R serait

installée.

Ces variables sont précieuses, ce qui implique qu'elles ne doivent pas être exportées vers l'environnement, sont conservées dans le cache même si elles ne sont pas spécifiées sur la ligne de commande, sont vérifiées pour la cohérence entre deux exécutions de configuration (à condition que la mise en cache soit utilisée) et sont conservés lors de la reconfiguration automatique comme s'ils avaient été passés en arguments de ligne de commande, même si aucun cache n'est utilisé.

Consultez la section sur la sortie des variables de configure --help pour une liste de toutes ces variables.

Si vous constatez que vous devez modifier les variables de configuration, il convient de noter que certains paramètres peuvent être mis en cache dans le fichier config.cache, et c'est une bonne idée de supprimer ce fichier (s'il existe) avant de reconfigurer. Notez que la mise en cache est désactivée par défaut : utilisez l'option de ligne de commande --config-cache (ou -C) pour activer la mise en cache.

B.3.1 Définition du format de papier Une

variable courante à modifier est R_PAPERSIZE, qui est par défaut « a4 », et non « lettre ». (Les valeurs valides sont « a4 », « lettre », « légal » et « exécutif ».)

Ceci est utilisé à la fois lors de la configuration de R pour définir la valeur par défaut et lors de l'exécution de R pour remplacer la valeur par défaut. Il est également utilisé pour définir le format du papier lors de la création de manuels PDF.

La configuration par défaut sera le plus souvent « a4 » si R_PAPERSIZE n'est pas défini. (Si le programme paperconf est trouvé, présent dans de nombreuses distributions Linux, ou si la variable d'environnement PAPERSIZE est définie, celles-ci sont utilisées pour produire la valeur par défaut.)

B.3.2 Paramétrage des navigateurs Une autre

variable précieuse est R_BROWSER, le navigateur HTML par défaut , qui doit prendre la valeur d'un exécutable dans le chemin de l'utilisateur ou spécifier un chemin complet.

Son homologue pour les fichiers PDF est R_PDFVIEWER.

B.3.3 Indicateurs de compilation Si vous

avez des bibliothèques et des fichiers d'en-tête, par exemple pour GNU readline, dans des répertoires non-système, utilisez les variables LDFLAGS (pour les bibliothèques, en utilisant les indicateurs '-L' à transmettre à l'éditeur de liens) et CPPFLAGS (pour fichiers d'en-tête, utilisant respectivement les indicateurs '-I' à transmettre aux préprocesseurs C/C++), pour spécifier ces emplacements. Ceux-ci sont par défaut '-L/usr/local/lib' (LDFLAGS, '-L/usr/local/lib64' sur la plupart des systèmes d'exploitation Linux 64 bits) et '-I/usr/local/include' (CPPFLAGS, mais notez celui sur la plupart des systèmes /usr/local/include est considéré comme un répertoire d'inclusion système et donc les instances de cette macro seront ignorées) pour détecter les cas les plus courants. Si les bibliothèques ne sont toujours pas trouvées, alors peut-être que votre compilateur/éditeur de liens ne prend pas en charge la réorganisation des indicateurs -L et -l. Dans ce cas, utilisez un autre compilateur (ou un script shell frontal qui effectue la réorganisation).

Ces indicateurs peuvent également être utilisés pour créer une version plus rapide de R. Sur la plupart des plates-formes utilisant gcc, avoir « -O3 » dans CFLAGS et FFLAGS produit des gains de performances intéressants avec gcc et gfortran, mais peut entraîner une version moins fiable (les deux des erreurs de segmentation et des calculs numériques incorrects ont été constatés). Sur les systèmes utilisant l' éditeur de liens GNU (en particulier ceux utilisant R comme bibliothèque partagée), il est probable qu'inclure '-Wl,-O1' dans LDFLAGS en vaut la peine, et "-Bdirect,--hash-style=both,-Wl ,-O1" est recommandé sur https:// lwn.net/Articles/ 192624/. Optimiser la compilation sur une famille de CPU spécifique (par exemple « -mtune=native » pour gcc) peut donner des gains de performances intéressants, en particulier sur les architectures plus anciennes telles que « ix86 ».

B.3.4 Création de manuels Les paramètres

par défaut pour la création de manuels sont contrôlés par R_RD4PDF et R_PAPERSIZE.

B.4 Paramétrage du shell Par défaut les

scripts shell tels que R seront des scripts '#!/bin/sh' (ou utilisant le SHELL choisi par configure). Ceci est presque toujours satisfaisant, mais sur quelques systèmes, /bin/sh n'est pas un shell ou un clone Bourne, et le shell à utiliser peut être modifié en définissant la variable de configuration R_SHELL sur une valeur appropriée (un chemin complet vers un shell, par exemple /usr/local/bin/bash).

B.5 Utilisation de make Pour

construire dans un répertoire séparé, vous avez besoin d'un make qui supporte la variable VPATH, par exemple GNU make et dmake.

Si vous souhaitez utiliser une marque sous un autre nom, par exemple si votre marque GNU s'appelle 'gmake', vous devez définir la variable MAKE au moment de la configuration, par exemple ./configure

MAKE=gmake

B.6 Utilisation de Fortran

Pour compiler R, vous avez besoin d'un compilateur Fortran 90. La valeur par défaut actuelle est de rechercher gfortran, g95, xlf95 f95, fort, ifort, ifc, efc, pgfortran, pgf95 lf95, ftn, nagfor, xlf90, f90, pgf90, pghpf, epcf90. (Notez qu'ils sont recherchés par leur nom, sans vérifier le standard de Fortran qu'ils prennent en charge.) La commande et les indicateurs utilisés doivent prendre en charge le Fortran de forme fixe avec l'extension .f : dans le cas inhabituel où un indicateur spécifique est nécessaire pour la forme libre. Fortran avec l'extension .f90 ou .f95, cela peut être spécifié dans le cadre de FCFLAGS.

Le mécanisme de recherche peut être modifié à l'aide de la variable de configuration FC qui spécifie la commande qui exécute le compilateur Fortran. Si votre compilateur Fortran se trouve dans un emplacement non standard, vous devez définir la variable d'environnement PATH en conséquence avant d'exécuter configure, ou utiliser la variable configure FC pour spécifier son chemin complet.

Si vos bibliothèques Fortran se trouvent dans des endroits un peu particuliers, vous devriez également consulter LD_LIBRARY_ PATH (ou l'équivalent de votre système) pour vous assurer que toutes les bibliothèques se trouvent sur ce chemin.

Notez que seuls les compilateurs Fortran qui convertissent les identifiants en minuscules sont pris en charge.

Vous devez définir tous les indicateurs de compilation (le cas échéant) nécessaires pour garantir que l'entier Fortran est équivalent à un pointeur C int et que la double précision Fortran est équivalente à un double pointeur C. Ceci est vérifié pendant le processus de configuration.

Une partie du code Fortran utilise des variables DOUBLE COMPLEX et COMPLEX*16. Ceci est vérifié au moment de la configuration, ainsi que son équivalence avec la structure Rcomplex C définie dans R_ext/Complex.h.

gfortran 10 donne par défaut une erreur de compilation pour la pratique auparavant répandue consistant à transmettre un élément de tableau Fortran là où un tableau est attendu, ou un scalaire au lieu d'un tableau de longueur un. Voir https://gcc.gnu.org/gcc-10/porting_to.html. gfortran 12 erreurs dans plus de cas.

B.7 Compiler et charger les indicateurs Une large gamme

d'indicateurs peut être définie dans le fichier config.site ou en tant que variables de configuration sur la ligne de commande. Nous avons déjà mentionné

Répertoire de recherche du fichier d'en-tête CPPFLAGS (-I) et toutes autres options diverses pour les fichiers C et
Préprocesseurs et compilateurs C++

Chemin LDFLAGS (-L), suppression (-s) et toutes autres options diverses pour l'éditeur de liens

et d'autres incluent

FLAGS indicateurs de débogage et d'optimisation, C

MAIN_CFLAGS
idem, pour compiler le programme principal (par exemple lors du profilage)

SHLIB_CFLAGS
pour les objets partagés (aucun exemple connu)

FFLAGS indicateurs de débogage et d'optimisation, Fortran de forme fixe

Indicateurs de débogage et d'optimisation FCFLAGS, Fortran de forme libre

SAFE_FFLAGS
idem pour les fichiers source qui nécessitent un comportement exact en virgule flottante

MAIN_FFLAGS
idem, pour compiler le programme principal (par exemple lors du profilage)

SHLIB_FFLAGS
pour les objets partagés (aucun exemple connu)

MAIN_LDFLAGS
drapeaux supplémentaires pour le lien principal

SHLIB_LDFLAGS
drapeaux supplémentaires pour lier les objets partagés

LIBnn le répertoire de la bibliothèque principale, lib ou lib64

CPICFLAGS
drapeaux spéciaux pour compiler du code C à transformer en un objet partagé

DRAPEAU FPICF
indicateurs spéciaux pour compiler du code Fortran à transformer en un objet partagé

CXXPICFLAGS
indicateurs spéciaux pour compiler du code C++ à transformer en objet partagé

DEFS définit à utiliser lors de la compilation du code C dans R lui-même

Les chemins de bibliothèque spécifiés comme -L/lib/path dans LDFLAGS sont rassemblés et ajoutés au début de LD_LIBRARY_PATH (ou l'équivalent de votre système), il ne devrait donc pas être nécessaire d'utiliser les indicateurs -R ou -rpath.

Les variables telles que CPICFLAGS sont déterminées lorsque cela est possible par configure. Certains systèmes autorisent deux types d'indicateurs PIC, par exemple « -fpic » et « -fPIC », et s'ils diffèrent, le premier n'autorise qu'un nombre limité de symboles dans un objet partagé. Étant donné que R en tant que bibliothèque partagée contient environ 6 200 symboles, en cas de doute, utilisez la version plus grande.

Les autres variables souvent définies par configure incluent 'MAIN_LDFLAGS', 'SAFE_FFLAGS', 'SHLIB_LDFLAGS' et 'SHLIB_CXXLDFLAGS' : voir le fichier config.site dans les sources pour plus de documentation sur celles-ci et d'autres.

Pour compiler une version de profilage de R, on pourrait par exemple vouloir utiliser 'MAIN_CFLAGS=-pg', 'MAIN_FFLAGS=-pg', 'MAIN_LDFLAGS=-pg' sur les plateformes où '-pg' ne peut pas être utilisé avec du code indépendant de la position. .

Attention : il peut être nécessaire de paramétrer CFLAGS et FFLAGS de manière compatible avec les bibliothèques à utiliser : un problème possible est l'alignement des doubles, un autre est la manière dont les structures sont passées.

Sur certaines plates-formes, configure sélectionnera des indicateurs supplémentaires pour CFLAGS, CPPFLAGS et LIBS dans R_XTRA_CFLAGS (et ainsi de suite). Il s'agit d'options qui sont toujours nécessaires, par exemple pour forcer la conformité à la norme CEI 60559.

B.8 Mode Mainteneur

Plusieurs fichiers font partie des sources R mais peuvent être régénérés à partir de leurs propres sources en configurant avec l'option --enable-maintainer-mode puis en exécutant make dans le répertoire de construction. Cela nécessite l'installation d'autres outils, abordés dans le reste de cette section.

Le fichier configure est créé à partir de configure.ac et des fichiers sous m4 par autoconf et aclocal (qui font partie du package automake). Il existe une exigence formelle de version sur autoconf de 2.69 ou ultérieure, mais il est peu probable que des versions autres que les versions3 les plus récentes aient été minutieusement testées.

Le fichier src/include/config.h est créé par autoheader (qui fait partie d'autoconf).

Les fichiers de grammaire *.y sont convertis en sources C par une implémentation de yacc, généralement bison -y : ils se trouvent dans src/main et src/library/tools/src. Il est connu que les versions antérieures de bison génèrent du code qui lit (et dans certains cas écrit) en dehors des limites du tableau : bison 2.6.1 s'est avéré satisfaisant.

Les sources ultimes du compilateur de packages se trouvent dans son répertoire nowweb. Pour recréer les sources depuis src/library/compiler/noweb/compiler.nw, la commande notangle est requise. Certaines distributions Linux incluent cette commande dans le package nowweb. Il peut également être installé à partir des sources sur https://www.cs.tufts.edu/~nr/noweb/4 . Les sources du package ne sont recréées même en mode responsable que si src/library/compiler/noweb/compiler.nw a été mis à jour.

[3] au moment de la révision de ce paragraphe fin 2021, autoconf-2.71 et automake-1.16.5.

[4] Les liens se sont révélés difficiles d'accès, auquel cas récupérez la copie mise à disposition sur https://developer.r-project.org/noweb-2.11b.tgz .

Annexe C Notes sur la plateforme

Cette section fournit quelques notes sur la construction de R sur différentes plates-formes de type Unix. Ces notes sont basées sur des tests exécutés sur un ou deux systèmes dans chaque cas avec des ensembles particuliers de compilateurs et de bibliothèques de support. Le succès de la construction de R dépend de l'installation et du fonctionnement corrects du logiciel de support ; vos résultats peuvent différer si vous disposez d'autres versions de compilateurs et de bibliothèques de support.

Les anciennes versions de ce manuel contiennent des notes sur les plates-formes telles que HP-UX, IRIX, Alpha/ OSF1 (pour R < 2.10.0, et la prise en charge a depuis été supprimée pour toutes ces plates-formes) et AIX (pour R < = 3.5.x). pour lequel nous n'avons pas eu de rapports récents.

Les macros C permettant de sélectionner des plates-formes particulières peuvent être difficiles à localiser (il existe de nombreuses informations erronées sur le Web). Le wiki (actuellement) sur https://sourceforge.net/p/predef/wiki/Home/ peut être utile. Les sources R ont utilisé (souvent dans les logiciels inclus sous src/extra)

AIX : _AIX
Cygwin : __CYGWIN__
FreeBSD : __FreeBSD__ HP-
UX : __hpux__, __hpux IRIX : sgi,
__sgi Linux : __linux__
macOS : __APPLE__

NetBSD : __NetBSD__
OpenBSD : __OpenBSD__
Windows : _WIN32, _WIN64

L'identification des compilateurs peut être très délicate. GCC définit __GNUC__, tout comme d'autres compilateurs qui prétendent s'y conformer, notamment (LLVM et Apple) clang et les compilateurs Intel récents.
De plus, ils utilisent la valeur de __GNUC__ pour leur version, et non la version de GCC avec laquelle ils prétendent être compatibles. LLVM et Apple clang définissent __clang__ comme leur version majeure, mais par exemple, le 13.xy d'Apple est très différent du 13.xy de LLVM. Et les compilateurs basés sur LLVM clang, par exemple d'IBM, le définiront. Certains des logiciels inclus utilisent __APPLE_CC__ pour identifier un compilateur Apple (qui incluait autrefois les versions Apple de GCC), mais le clang Apple est mieux identifié par la macro __apple_build_version__.

C.1 Problèmes X11

Le périphérique graphique 'X11()' est celui démarré automatiquement sur les systèmes Unix (sauf la plupart des versions de macOS) lors du traçage. Comme son nom l'indique, il s'affiche sur un serveur X (local ou distant), et s'appuie sur les services fournis par le serveur X.

La version « moderne » du périphérique « X11() » est basée sur les graphiques « Cairo » et (dans la plupart des implémentations) utilise « fontconfig » pour sélectionner et restituer les polices. Cela se fait sur le serveur, et bien qu'il puisse y avoir des problèmes de sélection, ils sont plus faciles à résoudre que les problèmes avec 'X11()' évoqués dans le reste de cette section.

Lorsque le X11 a été conçu, la plupart des écrans tournaient autour de 75 dpi, alors qu'aujourd'hui ils sont de l'ordre de 100 dpi ou plus. Si vous constatez que X11() signale1 des tailles de police manquantes, en particulier les plus grandes, il est probable que vous n'utilisez pas de polices évolutives et que vous n'avez pas installé les versions 100 dpi des polices X11. Les noms et les détails diffèrent selon le système, mais ressembleront probablement à celui de Fedora.

xorg-x11-fonts-75dpi xorg-x11-
fonts-100dpi

[1] par exemple, la police X11 de taille 14 n'a pas pu être chargée.

xorg-x11-fonts-ISO8859-2-75dpi xorg-x11-fonts-
Type1 xorg-x11-fonts-cyrillique

et vous devez vous assurer que les versions '-100dpi' sont installées et sur le chemin de la police X11 (vérifiez via xset -q). Le périphérique 'X11()' essaie de définir une taille en points et non en pixels : les utilisateurs d'ordinateurs portables peuvent trouver le paramètre par défaut de 12 trop grand (bien que très souvent les écrans d'ordinateurs portables soient réglés sur une résolution fictive pour apparaître comme un ordinateur de bureau réduit. écran).

Des problèmes plus compliqués peuvent survenir dans les paramètres régionaux non européens, donc si vous en utilisez un, la première chose à vérifier est que les choses fonctionnent dans les paramètres régionaux C. Les problèmes probables sont l'incapacité de trouver des polices ou des glyphes mal rendus (souvent sous la forme d'une paire de caractères ASCII). X11 fonctionne en se voyant demander une spécification de police et en proposant son idée d'une correspondance étroite. Pour le texte (contrairement aux symboles utilisés par plotmath), la spécification est le premier élément de l'option "X11fonts" qui est par défaut

"-adobe-helvetica-%s-%s-*-*-%d-*-*-*-*-*-*-*"

Si vous utilisez un codage sur un octet, par exemple ISO 8859-2 en Europe de l'Est ou KOI8-R en russe, utilisez xlsfonts pour rechercher une famille de polices appropriée dans votre codage (le dernier champ de la liste). Si vous n'en trouvez pas, il est probable que vous deviez installer d'autres packages de polices, tels que « xorg-x11-fonts-ISO8859-2-75dpi » et « xorg-x11-fonts-cyrillic » indiqués dans la liste ci-dessus.

Les codages multi-octets (le plus souvent UTF-8) sont encore plus compliqués. Il existe peu de polices dans « iso10646-1 », le codage Unicode, et elles ne contiennent qu'un sous-ensemble des glyphes disponibles (et sont souvent à largeur fixe conçues pour être utilisées dans les terminaux). Dans de tels paramètres régionaux, des jeux de polices sont utilisés, constitués de polices codées dans d'autres encodages. Si les paramètres régionaux que vous utilisez ont une entrée dans le répertoire 'XLC_LOCALE' (généralement /usr/share/X11/locale), il est probable que tout ce que vous ayez à faire soit de choisir une spécification de police appropriée contenant des polices dans les encodages spécifiés. là. Sinon, vous devrez peut-être vous procurer une entrée de paramètres régionaux appropriée pour X11. Cela peut signifier que, par exemple, le texte japonais peut être affiché lors de l'exécution dans « ja_JP.UTF-8 », mais pas lors de l'exécution dans « en_GB.UTF-8 » sur la même machine (bien que sur certains systèmes, de nombreux paramètres régionaux UTF-8 X11 soient affichés). sont alias 'en_US.UTF-8' qui couvre plusieurs jeux de caractères, par exemple ISO 8859-1 (Europe occidentale), JISX0208 (Kanji), KSC5601 (Coréen), GB2312 (Han chinois) et JISX0201 (Kana)).

Sur certains systèmes, des polices évolutives sont disponibles couvrant une large gamme de glyphes. Les polices TrueType/OpenType constituent une source d'information et peuvent offrir une couverture élevée. Les polices Type 1 en sont une autre : l'ensemble URW de polices Type 1 fournit des polices standard telles que Helvetica avec une plus grande couverture de glyphes Unicode que les bitmaps X11 standard, y compris le cyrillique. Ceux-ci ne font généralement pas partie de l'installation par défaut et le serveur X devra peut-être être configuré pour les utiliser. Elles peuvent se trouver dans le répertoire des polices X11 ou ailleurs, par exemple, /usr/share/fonts/default/

Type1 /usr/share/fonts/ja/TrueType

C.2 Linux

Linux est la principale plate-forme de développement pour R, donc la compilation à partir des sources est normalement simple avec les compilateurs et bibliothèques les plus courants.2 Cette section concerne

les compilateurs GCC : gcc/gfortran/g++.

Rappelons que certains systèmes de gestion de packages (tels que RPM et deb) font une distinction entre la version utilisateur d'un package et la version développeur. Ce dernier porte généralement le même nom mais avec l'extension '-devel' ou '-dev' : vous devez installer les deux versions. Veuillez donc vérifier la sortie de configuration pour voir si les fonctionnalités attendues sont détectées : si par exemple

[2] Par exemple, glibc : d'autres bibliothèques C telles que musl (telle qu'utilisée par Alpine Linux) ont été utilisées mais ne sont pas systématiquement testées.

'readline' est manquant, ajoutez le package développeur. (Sur la plupart des systèmes, vous aurez également besoin de « ncurses » et de son package de développement, bien que ceux-ci doivent être des dépendances du ou des packages « readline ».)
Vous devriez vous attendre à voir dans le résumé de configuration

```
Interfaces prises en charge : X11, tcltk pcre2,
Bibliothèques externes : readline, curl
Capacités supplémentaires : PNG, JPEG, TIFF, NLS, le Caire, ICU Lorsque R a été installé à
```

partir d'une distribution binaire, il y a parfois des problèmes avec des composants manquants tels que le compilateur Fortran. La recherche dans les archives « R-help » révélera normalement ce qui est nécessaire.

Il semble que Linux 'ix86' accepte le code non-PIC dans les bibliothèques partagées, mais ce n'est pas nécessairement le cas sur d'autres plateformes, en particulier sur les processeurs 64 bits tels que 'x86_64'. Il peut donc être nécessaire de faire preuve de prudence avec les bibliothèques BLAS et lors de la construction de R en tant que bibliothèque partagée pour garantir que du code indépendant de la position est utilisé dans toutes les bibliothèques statiques (telles que les bibliothèques Tcl/Tk, libpng, libjpeg et zlib) qui pourraient être liées. Heureusement, celles-ci sont normalement construites sous forme de bibliothèques partagées, à l'exception des bibliothèques ATLAS BLAS .

Les paramètres d'optimisation par défaut choisis pour CFLAGS, etc. sont conservateurs. Il est probable que l'utilisation de -mtune entraînera des améliorations significatives des performances sur les processeurs récents : une possibilité consiste à ajouter -mtune=native pour obtenir les meilleures performances possibles sur la machine sur laquelle R est installé. Il est également possible d'augmenter les niveaux d'optimisation jusqu'à -O3 : cependant pour de nombreuses versions des compilateurs cela a posé des problèmes dans au moins un package CRAN .

N'utilisez pas -O3 avec gcc 11.0 ou 11.1 : cela compile mal le code, ce qui donne des résultats plausibles mais incorrects. (Cela a été vu dans le package MASS (https://CRAN.R-project.org/package=MASS) mais a été contourné à partir de la version 3.1-57.)

Pour les plates-formes prenant en charge à la fois 64 et 32 bits, il est probable que

```
LDFLAGS="-L/usr/local/lib64 -L/usr/local/lib"
```

est approprié puisque la plupart (mais pas tous) les logiciels installent leurs bibliothèques 64 bits dans /usr/local/lib64.
Pour créer une version 32 bits de R sur 'x86_64' avec Fedora 34, nous avons utilisé

```
CC="gcc-m32"
CXX="g++-m32"
FC = "gfortran -m32"
OBJC=${CC}
LDFLAGS="-L/usr/local/lib"
LIBnn=lib
```

Notez l'utilisation de 'LIBnn' : 'x86_64' Fedora installe son logiciel 64 bits dans /usr/lib64 et son logiciel 32 bits dans /usr/lib. La liaison ignorera les binaires inappropriés, mais par exemple, les scripts de configuration Tcl/Tk 32 bits se trouvent dans /usr/lib. Il peut également être nécessaire de définir le chemin pkg-config, par exemple en export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig:/

```
usr/lib/pkgconfig
```

La libcurl système 32 bits ne fonctionnait pas avec les certificats CA système : ce problème est résolu dans la suite de tests de R.

Les versions 64 bits sous Linux sont construites avec la prise en charge des fichiers > 2 Go, et les versions 32 bits le seront si possible, à moins que --disable-largefile ne soit spécifié.

Notez que certaines distributions de glibc 32 bits utilisent un type time_t 32 bits, donc pour passer tous les les vérifications date-heure nécessitent que R soit construit avec l'indicateur --with-internal-tzcode.

Les utilisateurs de processeurs « ix86 » prenant en charge SSE23 peuvent préférer utiliser les options C/C++/Fortran.

```
-mfpmath=sse -msse2
```

[3] Probablement depuis 2005, y compris le Pentium 4 et tous les processeurs « x86_64 » dotés de compilateurs 32 bits.

pour forcer la virgule flottante à utiliser les mêmes instructions que les builds 'x86_64' et donc à ne pas utiliser de résultats intermédiaires de 'précision étendue' de 80 bits. (NB : cela affecte plus que les opérations à virgule flottante. Pour certains systèmes d'exploitation et versions de gcc, il peut être nécessaire d'ajouter -mstackrealign.)

Pour créer une version 64 bits de R sur 'ppc64' (également connu sous le nom de 'powerpc64') avec gcc 4.1.1, Ei-ji Nakama a utilisé

```
CC="gcc-m64"
CXX="gxx-m64"
FC = "gfortran -m64"
CFLAGS="-mminimal-toc -fno-optimize-sibling-calls -g -O2"
FFLAGS="-mminimal-toc -fno-optimize-sibling-calls -g -O2"
```

les indicateurs supplémentaires étant nécessaires pour résoudre les problèmes de liaison avec libnmath.a et lors de la liaison de R en tant que bibliothèque partagée.

Le paramétrage de la macro 'SAFE_FFLAGS' peut nécessiter un peu d'aide. Il ne devrait pas avoir besoin d'indicateurs supplémentaires sur les plates-formes autres que « 68000 » (peu probable) et « ix86 ». Pour ce dernier, si le compilateur Fortran est GNU (gfortran ou éventuellement g77) les drapeaux

```
-msse2 -mfpmath=sse
```

sont ajoutés : les versions antérieures de R ont ajouté -ffloat-store et cela peut toujours être nécessaire si un processeur 'ix86' est rencontré sans prise en charge SSE2. Notez qu'il s'agit d'un remplacement de 'FFLAGS', il devrait donc inclure tous les indicateurs dans cette macro (sauf peut-être le niveau d'optimisation).

Des indicateurs de compilation supplémentaires peuvent être spécifiés pour des contrôles de sûreté/sécurité supplémentaires. Par exemple Fedora ajoute

```
-Werror=format-security -Wp,-D_FORTIFY_SOURCE=3 -Wp,-D_GLIBCXX_ASSERTIONS
-Fexceptions -fstack-protector-strong -fasynchronous-unwind-tables -fstack-clash-protection -fcf-protection
```

à tous les indicateurs du compilateur C, C++ et Fortran (même si _GLIBCXX_ASSERTIONS est uniquement pour le C++ dans GCC et glibc actuels et qu'aucun d'entre eux n'est documenté pour gfortran). L'utilisation de _GLIBCXX_ ASSERTIONS liera abort et printf dans presque tout le code C++, et R CMD check --as-cran avertira.

C.2.1 Clang R a été

construit avec les compilateurs Linux 'ix86' et 'x86_64' C et C++ (https://clang.llvm.org) basés sur les frontaux Clang, invoqués ensemble par CC=clang CXX=clang++ avec gfortran.
Ceux-ci prennent des options très similaires à celles des compilateurs GCC correspondants.

Celui-ci doit être utilisé conjointement avec un compilateur Fortran : le code de configuration supprimera -lgcc de FLIBS, ce qui est nécessaire pour certaines versions de gfortran.

La valeur par défaut actuelle pour clang++ consiste à utiliser le runtime C++ du g++ installé. L'utilisation du runtime du projet libc++ (https://libcxx.llvm.org/) (Fedora RPM libcxx-devel) via -stdlib=libc++ a également été testée.

Les versions récentes ont (facultatif une fois construites) le support OpenMP.4 Il y

a des problèmes pour mélanger clang 15.0.0 construit par défaut sur Linux pour produire du code PIE et gfortran 11 ou 12, qui ne le font pas. Un symptôme est que configure ne détecte pas FC_LEN_T, ce qui peut être surmonté en définissant

```
FPIEFLAGS=-fPIE
```

dans config.site. (À partir de R 4.2.2, configure essaie cette valeur si elle n'est pas définie.)

[4] Cela nécessite également le runtime OpenMP qui a parfois été distribué séparément.

C.2.2 Compilateurs Intel Il n'y a pas

eu de rapports récents sur l'utilisation des compilateurs Intel : cette section est laissée au cas où elle fournirait des conseils utiles.

Les compilateurs Intel ont été utilisés sous Linux « ix86 » et « x86_64 ». Brian Ripley version utilisée 9.0 des compilateurs pour 'x86_64' sur Fedora Core 5 avec

```
CC=icc
CFLAGS="-g -O3 -wd188 -ip -mp"
FC=iffort
FLAGS="-g -O3 -mp"
CXX=icpc
CXXFLAGS="-g -O3 -mp"
ICC_LIBS=/opt/compilers/intel/cce/9.1.039/lib IFC_LIBS=/opt/
compilers/intel/fce/9.1.033/lib LDFLAGS="-L$ICC_LIBS -L$IFC_LIBS
-L/usr/local/ lib64"
SHLIB_CXXLD=icpc
```

Il peut être nécessaire d'utiliser CC="icc -std=c99" ou CC="icc -c99" pour la conformité C99. L'indicateur -wd188 supprime un grand nombre d'avertissements concernant le type d'énumération 'Rboolean'. Étant donné que le compilateur Intel C définit '__GNUC__' sans émulation complète de gcc, nous suggérons d'ajouter CPPFLAGS=-no-gcc.

Pour maintenir une arithmétique CEI 60559 correcte, vous devrez probablement ajouter des indicateurs à CFLAGS, FFLAGS et CXXFLAGS tels que -mp (illustré ci-dessus) ou -fp-model précis -fp-model source, selon la version du compilateur.

D'autres ont signalé du succès avec les versions 10.x et 11.x. Bjørn-Helge Mevik rapporte un succès avec la version 2015.3 des compilateurs, en utilisant (pour un CPU SandyBridge sur Centos 6.x)

```
fast="-fp-model précis -ip -O3 -opt-mem-layout-trans=3 -xHost -mavx"
CC=icc
CFLAGS="$rapide -wd188"
FC=iffort
FFLAGS="$rapide"
CXX=icpc
CXXFLAGS="$rapide"
```

Il est possible que les builds 32 doivent forcer l'utilisation des instructions SSE2 dans SAFE_FFLAGS, par exemple en

```
SAFE_FFLAGS=-axsse2
```

C.3 macOS

Les instructions ici concernent les versions Intel 64 bits (« x86_64 ») ou « Apple Silicon » (« arm64 ») sur macOS 11 (Big Sur), 12 (Monterey) et 13 (Ventura). (Ils peuvent très bien fonctionner sur Intel macOS 10.14 ou 10.15, mais n'y sont pas testés.)

C.3.1 Prérequis Les composants

Intel s'installent dans /opt/R/x86_64, ceux 'arm64' dans /opt/R/arm64 qui peuvent ne pas exister, il est donc plus simple de créer d'abord ce répertoire et d'ajuster sa propriété si vous le souhaitez :

```
sudo mkdir -p /opt/R/arm64 sudo chown
-R $USER /opt/R et ajoutez /opt/R/
```

arm64/bin à votre chemin.

Définissez une variable appropriée dans votre Terminal : set

```
LOCAL=//opt/R/x86_64 # Intel
```

définissez LOCAL=/opt/R/arm64 # Apple Silicon pour utiliser

les extraits de code ici.

Les éléments suivants sont essentiels pour

construire R : • Les « outils de ligne de commande » d'Apple : ils peuvent être (ré)installés en exécutant xcode-select --installer dans un terminal.

Si vous disposez d'une nouvelle installation du système d'exploitation, exécuter par exemple make dans un terminal proposera l'installation des outils de ligne de commande. Si vous avez installé Xcode, cela fournit les outils de ligne de commande. Les outils devront peut-être être réinstallés lors de la mise à niveau de macOS, car la mise à niveau peut les supprimer partiellement ou complètement.

Les outils de ligne de commande fournissent des compilateurs C et C++ dérivés du clang de LLVM mais aujourd'hui connus sous le nom de « Apple clang » avec une version différente (donc Apple clang 14 n'a aucun rapport avec LLVM clang 14).

• Un compilateur Fortran.

Voir Section C.3.2 [Compilateurs Fortran], page 63. • Composants binaires pcre25 et xz (pour

liblzma) de https://mac.r-project.org/bin/ . Il existe un script R pour vous aider à installer tous les composants nécessaires. (Au moment de la rédaction de cet article, install.libs("r-base-dev") n'installait ni readline5 ni ceux nécessaires à la prise en charge de Pango.)

Les utilisateurs d'Intel veulent les composants darwin20 : ceux darwin17 sont pour macOS 10.13-10.15.

Ou cela peut être fait manuellement, par exemple

curl -OL https://mac.r-project.org/bin/darwin20/x86_64/pcre2-10.42-darwin.20-x86_6 sudo tar -xvzf pcre2-10.42-darwin.20-x86_64.tar.gz -C / curl -OL https://mac.r-project.org/bin/darwin20/x86_64/xz-5.4.2-darwin.20-x86_64.t sudo tar -xvzf xz-5.4.2-darwin.20-x86_64. tar.xz -C /

(Sudo n'est pas nécessaire si votre compte possède /opt/R/x86_64 ou /opt/R/arm64 selon le cas.)

Les messages tels que « opt/R/ : Impossible de restaurer l'heure » doivent être ignorés.

et souhaitable

libedit (alias editline) sera utilisé : si 6 Si readline n'est pas présent, l'émulation dans la version Apple du • composant readline5. vous souhaitez éviter cela, configurez avec --without-lire la ligne.

• Composants jpeg, libpng, pkgconfig, tiff et zlib-system-stub de https://mac. r-project.org/bin// pour la gamme complète de périphériques graphiques bitmap. (Certaines versions de tiff peuvent nécessiter libwebp et/ou openjpeg.)

• Un sous-système X sauf configuration utilisant --without-x : voir https://www.xquartz.org/.
Le script de configuration de R peut être invité à rechercher X11 dans l'emplacement principal de XQuartz /opt/X11, par exemple

en utilisant --x-includes=/opt/X11/include --x-libraries=/opt/X11/lib Méfiez-vous des pré

-versions release de XQuartz, qui peuvent être proposées sous forme de mise à jour.

• Un compilateur Objective-C, tel que fourni par clang dans les outils de ligne de commande : il est nécessaire pour le périphérique graphique quartz().

Utilisez --without-aqua si vous voulez une version standard de type Unix : outre la désactivation de quartz() et la possibilité d'utiliser la version avec R.app, cela modifie également l'emplacement par défaut de la bibliothèque personnelle (voir ?. libPaths).

• Une installation Tcl/Tk, Voir non défini [En-têtes et bibliothèques Tcl/Tk], page non défini.

---

[5] Si vous le compilez à partir des sources sur 'arm64', pcre2 (au moins jusqu'à la version 10.39) doit être construit sans le support JIT (par défaut) car les erreurs de segmentation de la build R si cela est activé, alors exécutez make check sur votre build.

[6] Pour des raisons de licence il s'agit de la version 5.2 de readline : pour ceux qui souhaitent une version plus récente il est simple de la compiler à partir de ses sources.

• Prise en charge des périphériques graphiques basés sur Cairo. Voir Section C.3.3 [Graphiques du Caire], page 63. • Une installation TeX. Voir Section C.3.5 [Autres bibliothèques], page 64. •

texi2any d'une distribution 'texinfo', qui nécessite Perl (actuellement une partie par défaut de macOS mais il a été annoncé qu'il ne le sera peut-être plus à l'avenir). Une version de texi2any a été incluse dans la distribution binaire de R et il existe un composant texinfo sur https://mac.r-project.org/bin/.

Pour construire R lui-même à partir des sources avec les compilateurs C/C++ dans les outils de ligne de commande (ou Xcode) et gfortran depuis l'installateur mentionné ci-dessous, utilisez un fichier config.site contenant

```
CC=clang
OBJC=$CC
FC="/opt/gfortran/bin/gfortran -mtune=native"
CPPFLAGS='-isystem $LOCAL/include' CXX=clang+
+ et configurez
```

par quelque chose comme ./configure -C

```
\ --enable-R-shlib --
enable-memory-profiling \ --x-includes=/opt/X11/ include --x-libraries=/
opt/X11/lib \ --with-tcl-config=$LOCAL/lib/tclConfig.sh \ --with-tk-config=$LOCAL/lib/
tkConfig.sh \ PKG_CONFIG_PATH= $LOCAL/lib/pkgconfig:/usr/lib/
pkgconfig (Voir ci-dessous pour d'autres options pour Tcl/Tk.)
Pour une construction 'arm64', d'autres indicateurs sont souhaitables dans config.site :
```

```
CFLAGS="-falign-functions=8 -g -O2"
FFLAGS="-g -O2 -mmacosx-version-min=11.0"
FCFLAGS="-g -O2 -mmacosx-version-min=11.0"
```

(le premier indicateur de CFLAGS est nécessaire pour interfonctionner avec gfortran sans erreur de segmentation dans certains packages, et certaines versions de gfortran ont ciblé la version actuelle de macOS (contrairement à clang, provoquant des avertissements de l'éditeur de liens).

Pour installer des packages en utilisant du code compilé, il faut les outils de ligne de commande (ou Xcode) et les compilateurs appropriés, par exemple les compilateurs C/C++ de ces outils et/ou gfortran. Certains packages ont des exigences supplémentaires telles que le composant pkgconfig (et définir PKG_CONFIG_ PATH= comme ci-dessus).

Un client Subversion peut être obtenu depuis https://mac.r-project.org/tools/, par exemple en

```
curl -OL https://mac.r-project.org/tools/subversion-1.14.0-darwin15.6.tar.gz tar xf subversion-1.14.0-darwin15.6.tar.gz
sudo cp subversion-1.14 .0-darwin15.6/svn $LOCAL/bin (Il existe
également une version arm64 sur cette page Web.)
```

Si vous créez un logiciel ou installez des packages sources avec cmake (ou une marque non Apple) pour « Apple Silicon », assurez-vous qu'il contient l'architecture « arm64 » (utilisez le fichier pour être sûr). L'exécution de compilateurs Apple à partir d'un exécutable 'x86_64' générera du code 'x86_64'. . ..

La mise à jour d'une version « arm64 » peut échouer en raison du bogue décrit sur https://openradar. appspot.com/FB8914243 mais les builds ab initio fonctionnent. C'est beaucoup plus rare depuis macOS 13.

Si vous utilisez le SDK7 macOS 13 , vous devrez peut-être ajouter quelque chose comme -mmacosx-version-min=12.0 à « CFLAGS ».

---

[7] ls -l 'xcrun -show-sdk-path' dans un terminal vous montrera quel SDK est sélectionné.

Avertissements de l'éditeur de liens comme

ld : avertissement : impossible de créer un déroulement compact pour _sort_ : registre 26
enregistré ailleurs que dans le cadre
ld : avertissement : ld : avertissement :
impossible de créer un déroulement compact pour _arcoef_ : les registres 23 et 24 ne sont pas enregistrés contigu
ld : avertissement : impossible de créer un déroulement compact pour ___emutls_get_address : les registres 23 et
24 ne sont pas enregistrés de manière contiguë dans le cadre

peut être ignoré. Ceux-ci proviennent du code Fortran compilé, y compris de ses bibliothèques d'exécution.

Les paramètres de sécurité par défaut peuvent rendre difficile l'installation de packages Apple qui n'ont pas été « notariés »8 par Apple. Et pas seulement les packages, comme cela a été vu pour les exécutables contenus dans les archives tar/zip (par exemple, pour pandoc). Habituellement, vous pouvez utiliser « Ouvrir avec » (Contrôle/droit/clic avec deux doigts dans le Finder), puis sélectionner « Installateur » et « Ouvrir » si vous recevez un autre message d'avertissement. Cela s'applique parfois également aux « builds nocturnes » de https://mac.R-project. org/.

Si vous rencontrez des problèmes avec la « quarantaine » pour les archives tar téléchargées dans un navigateur, envisagez d'utiliser curl -OL pour télécharger (comme illustré ci-dessus) ou xattr -c pour supprimer les attributs étendus. configurez les

valeurs par défaut sur --with-internal-tzcode sur macOS. L'implémentation native était inutilisable sur les versions antérieures (avec un time_t 32 bits et/ou des tables de fuseau horaire manquant d'informations au-delà de la plage 32 bits). Pour, par exemple, macOS 12.6, l'option --without-internal-tzcode peut être utilisée pour remplacer cela et R contient suffisamment de solutions de contournement (par exemple, le code natif ne reconnaît pas les dates avec un tm_year négatif, c'est-à-dire les dates antérieures à 1900) pour R. passer ses contrôles. Il existe cependant des divergences, notamment en Europe dans les années 1900 et 1940, même si la base de données Olson contient les informations correctes.

C.3.2 Compilateur Fortran Il existe une

version « universelle » (Intel et arm64) de gfortran 12.2 sur https://mac.r-project.org/tools/gfortran-12.2-universal.pkg . Cela s'installe dans /opt/gfortran.

Le lien symbolique /opt/gfortran/SDK doit pointer vers le chemin souhaité vers le SDK (par défaut, le SDK des outils de ligne de commande). Cela peut être mis à jour en exécutant /opt/gfortran/bin/gfortran-update-sdk ou manuellement.

Si le lien symbolique est rompu, le pilote émettra un avertissement et utilisera xcrun -show-sdk-path pour essayer de trouver un SDK et d'utiliser son chemin. (Le chemin du SDK est utilisé lors de l'utilisation de gfortran pour établir un lien, donc pas lors de la construction de R mais lors de l'installation de quelques packages.)

C.3.3 Graphiques du Caire Les

périphériques graphiques basés sur le Caire tels que cairo_ps, cairo_pdf, X11(type = "cairo") et les types de périphériques basés sur le Caire bmp jpeg, png et tiff ne sont pas ceux par défaut sur macOS et sont beaucoup moins utilisés que les appareils à base de quartz. Cependant, le seul périphérique SVG de la distribution R, svg, est basé sur Cairo.

La prise en charge de Cairo est facultative et peut être ajoutée de plusieurs manières, qui nécessitent toutes pkg-config. configure ajoutera le support de Cairo si pkg-config trouve le paquet cairo, sauf si --without-cairo
est utilisé.

Un moyen de lier statiquement Cairo consiste à télécharger et décompresser les composants cairo, fontconfig, freetype, pixman et zlib-system-stub (et n'ont pas /opt/X11/lib/pkgconfig dans PKG_CONFIG_PATH). Certaines versions statiques de fontconfig nécessitent libxml2 (du composant xml2) et d'autres expat, fournies par macOS mais nécessitant un fichier $LOCAL/lib/pkgconfig/expat.pc du type

Nom : expatrié

8 Voir https://developer.apple.com/documentation/xcode/notarizing_macos_software_before_distribution.

Version : 2.2.8
Description : URL de l'analyseur XML pour
expatriés : http://www.libexpat.org Libs :
-lexpat Cflags :

Notez que la liste des composants est susceptible de changer : exécuter pkg-config cairo --exists --print-errors devrait vous indiquer si d'autres sont requis.

La meilleure expérience de police des graphiques Cairo sera de l'utiliser en combinaison avec Pango qui correspondra à celui pris en charge sur la plupart des autres Unix-alikes. configure utilise pkg-config pour déterminer si tous les logiciels externes requis par Pango et Cairo sont disponibles : exécuter pkg-config pangocairo --exists --print-errors devrait montrer si l'installation est suffisante et sinon, ce qui manque. Au moment de la rédaction, l'utilisation de composants prédéfinis Cairo, fontconfig, freetype, ffi, fribidi, gettext, icu, glib, harfbuzz, pango, pcre, pixman et xml2 suffisait.

C.3.4 Autres compilateurs C/C++ D'autres

distributions de clang peuvent être disponibles sur https://github.com/llvm/llvm-project/ releases/ (récemment uniquement pour arm64 et généralement non signées/non notariées, ce qui les rend difficiles à utiliser) . Ceux-ci incluent notamment la prise en charge d'OpenMP, ce que Apple n'offre pas. Certains d'entre eux incluent la prise en charge des désinfectants ASAN et UBSAN.

Supposons qu'une de ces distributions soit installée sous $LOCAL/llvm. Utilisez un fichier config.site contenant

CC="$LOCAL/llvm/bin/clang -isysroot /Library/Developer/CommandLineTools/SDKs/MacOSX.sd CXX="$LOCAL/llvm/bin/clang+
+ -isysroot/Library/Developer/CommandLineTools/SDKs/MacOSX OBJC= $CC FC=$LOCAL/gfortran/bin/gfortran LDFLAGS="-
L$LOCAL/
llvm/lib -L$LOCAL/lib"

R_LD_LIBRARY_PATH=$LOCAL/llvm/lib:$LOCAL/lib

Si l'emplacement du SDK change (ou si Xcode fournit le SDK plutôt que les outils de ligne de commande), il peut être trouvé en exécutant xcrun -show-sdk-path.

Le soin apporté à la spécification des chemins de bibliothèque est de garantir que la bibliothèque d'exécution OpenMP, ici $LOCAL/llvm/lib/libomp.dylib, est trouvée en cas de besoin. Si cela fonctionne, vous devriez voir la ligne

vérifier si la réduction OpenMP SIMD est prise en charge... oui

dans la sortie de configuration. De plus, « R_LD_LIBRARY_PATH » doit être défini pour rechercher la dernière version des bibliothèques d'exécution C++ plutôt que celles du système.

Il est généralement possible de construire R avec GCC (construit à partir des sources, à partir d'une distribution le bouton, à partir de Homebrew, ∴ gfortran.). Lors du dernier test[9], il n'était pas possible d'utiliser gcc pour construire quartz(), donc une configuration --without-aqua peut être requise.

Il est généralement possible d'ajouter un peu de support OpenMP aux compilateurs Apple Clang : voir https://mac.r-project.org/openmp/ . Notez que cette approche est quelque peu fragile car elle nécessite une bibliothèque libomp.dylib correspondant à la version du compilateur utilisé — et par exemple, au moment de la rédaction, aucune n'était proposée pour les compilateurs actuels dans Xcode/CLT 14.3.

C.3.5 Autres bibliothèques

Des versions précompilées de la plupart des sections A.2 [Bibliothèques et programmes utiles], page 41, sont disponibles sur https://mac.r-project.org/bin//.

[9] avec gcc 10.2.

La bibliothèque Accelerate10 peut être utilisée via l'option de configuration

--with-blas="-framework Accélérer"

pour fournir des versions potentiellement plus performantes des routines BLAS et LAPACK.11 il inclut également un LAPACK complet qui peut être utilisé via --with-lapack : cependant, la version de LAPACK qu'il contient est souvent très ancienne (et n'est pas utilisée sauf si --with-lapack est spécifié). Certaines versions CRAN de R peuvent être commutées12 pour utiliser le BLAS d'Accelerate.

Le threading dans Accelerate est contrôlé par « Grand Central Dispatch » et ne nécessiterait pas le contrôle de l'utilisateur. Le test nls.R dans les statistiques du package a souvent échoué avec Accelerate BLAS sur Intel macOS.

En regardant en haut de /Library/Frameworks/R.framework/Resources/etc/Makeconf, vous verrez les compilateurs et les options de configuration utilisés pour le package binaire CRAN pour R : au moment de la rédaction, les options autres que celles par défaut

--enable-memory-profiling --enable-R-framework --x-libraries=/opt/
X11/lib --x-includes=/opt/X11/include

ont été utilisées. (--enable-R-framework implique --enable-R-shlib.)

La principale implémentation de TEX utilisée par les développeurs est MacTeX13 (https://www.tug.org/mactex/) : l'installation complète fait environ 8,5 Go, mais une version beaucoup plus petite (« Basic TeX ») est disponible sur https:// /www.tug.org/mactex/morepackages.html auquel vous devrez ajouter quelques packages pour construire R, par exemple pour la version 2022, nous devions ajouter 14 helvetic, inconsolata et texinfo, ce qui a porté cela à environ 310 Mo.15 'TeX Live Utility' (disponible via la page d'accueil de MacTeX) fournit un moyen graphique pour gérer les packages TEX. Ceux-ci contiennent des exécutables qui s'exécutent nativement sur « x86_64 » et « arm64 ».

La vérification approfondie des packages nécessite ghostscript (qui fait partie de la distribution complète de MacTeX ou séparément de https://www.tug.org/mactex/morepackages.html) et qpdf (de https://mac.r-project.org/bin //, dont une version se trouve dans le répertoire bin d'une installation binaire de R, généralement /Library/Frameworks/R.framework/Resources/bin/qpdf).

R CMD check --as-cran utilise 'HTML Tidy'. macOS a une version dans /usr/bin/tidy datant de 2006 qui est bien trop ancienne et est ignorée. Les versions à jour peuvent être installées à partir de http://binaries.html-tidy.org/.

Une bizarrerie de macOS est que le chemin par défaut a /usr/local/bin après /usr/bin, contrairement à la pratique courante sur les Unix-alikes. Cela signifie que si vous installez des outils à partir des sources, ils seront installés par défaut sous /usr/local et ne remplaceront pas les versions du système.

L'installation parallèle des packages utilisera le délai d'attente de l'utilitaire s'il est disponible. Une version à double architecture peut être téléchargée depuis https://www.stats.ox.ac.uk/pub/bdr/timeout : rendez-la exécutable (chmod 755 timeout) et placez-la quelque part sur votre chemin.

C.3.6 En-têtes et bibliothèques Tcl/Tk Si vous prévoyez

d'utiliser le package tcltk pour R, vous devrez installer une distribution de Tcl/Tk.
Il existe deux alternatives. Si vous utilisez R.app, vous souhaiterez utiliser Tcl/Tk basé sur X11 (tel qu'utilisé sur d'autres Unix-alikes), qui est installé sous $LOCAL/lib dans le cadre du binaire CRAN pour R.16. Cela peut nécessiter des options de configuration .

---

dix https://developer.apple.com/documentation/accelerate.

11 Il a été signalé que pour certaines chaînes d'outils non Apple, CPPFLAGS devait contenir -D__ACCELERATE__ : non nécessaire pour le clang de LLVM. https://cran.r-project.org/

12 bin/macosx/RMacOSX-FAQ.html#Which-BLAS-is-used-and-how-can-it-be-changed_003f Une installation TEX essentiellement équivalente peut être obtenue

13 par les scripts d'installation d'Unix TeX Live.

14 Par exemple, via tlmgr, installez helvetic inconsolata texinfo .

15 L'ajout de tous les packages nécessaires pour vérifier CRAN a augmenté cette taille à environ 600 Mo.

16 Seul ce composant peut être sélectionné à partir du programme d'installation pour R : sur l'écran « Type d'installation », sélectionnez « Personnaliser », puis uniquement le composant « Tcl/Tk 8.6.11 ».

--with-tcltk=$LOCAL/lib

ou

--with-tcl-config=$LOCAL/lib/tclConfig.sh --with-tk-config=$LOCAL/
lib/tkConfig.sh Notez que cela nécessite une installation

XQuartz correspondante.

Il existe également une version native (« Aqua ») de Tcl/Tk qui produit des widgets dans le style natif de macOS : cela ne fonctionnera pas avec R.app en raison de conflits avec le menu macOS, mais pour ceux qui utilisent uniquement la ligne de commande R, ceci fournit une interface beaucoup plus intuitive à Tk pour les utilisateurs Mac expérimentés. Les versions antérieures de macOS étaient livrées avec une distribution Aqua Tcl/Tk, mais il ne s'agissait souvent pas de versions récentes de Tcl/Tk. Il est préférable d'installer Tcl/Tk 8.6.x depuis les sources[17] ou une distribution binaire depuis https://www.activestate.com/products/tcl/. Pour ce dernier, configurez R avec

--with-tcl-config=/Library/Frameworks/Tcl.framework/tclConfig.sh --with-tk-config=/Library/Frameworks/
Tk.framework/tkConfig.sh

Si vous avez besoin de savoir quelle distribution de Tk est utilisée au moment de l'exécution,

utilisez library(tcltk)
tclvalue(.Tcl("tk windowingsystem")) # "x11" ou "aqua"

Notez que certaines extensions Tcl/Tk ne prennent en charge que l'interface X11 : cela inclut Tktable et le package CRAN tkrplot (https://CRAN.R-project.org/package=tkrplot).

C.3.7 Java

macOS n'est pas livré avec un runtime Java (JRE) installé et une mise à niveau de macOS peut en supprimer un s'il est déjà installé : il est destiné à être installé lors de la première utilisation. Vérifiez si un JRE est installé en exécutant java -version dans une fenêtre de terminal : si Java n'est pas installé[18], cela devrait vous inviter à l'installer.[19] Des versions d'OpenJDK peuvent également être disponibles, par exemple depuis Adoptium (https://adoptium.net). , Azul (https://www.azul.com/downloads/zulu-community/) ou https://jdk.java.net/ . Nous vous recommandons d'installer une version avec un support à long terme, par exemple 11 ou 17 ou (attendu en 2023-09) 21 mais pas 12-16 ni 18-20 qui ont/ont eu une durée de vie de 6 mois. (Notez que ces sources peuvent utiliser des désignations inhabituelles pour les versions Intel macOS telles que x86 64 bits et x64.)

Les distributions binaires de R sont construites sur une version spécifique (par exemple 11.0.6 ou 17.0.1) de Java donc Sudo R CMD javareconf devra probablement être exécuté avant d'utiliser des packages utilisant Java.

Pour voir quelles versions compatibles de Java sont actuellement installées, exécutez celle appropriée de /usr/libexec/

java_home -V -a x86_64 /usr/libexec/java_home -V -a
arm64 Si nécessaire, définissez la variable

d'environnement JAVA_HOME pour choisir entre celles-ci. , à la fois lorsque R est construit à partir des sources et lorsque R CMD javareconf est exécuté.

La configuration et la construction de R recherchent à la fois un JRE et la prise en charge de la compilation de programmes JNI (utilisés pour installer les packages rJava (https://CRAN.R-project.org/package=rJava) et JavaGD (https://CRAN. R-project.org/package=JavaGD)); ce dernier nécessite un JDK (Java SDK). La plupart des distributions de Java 9 ou version ultérieure contiennent un JDK complet.

Le processus de construction essaie de comprendre quel JRE/JDK utiliser, mais il peut avoir besoin d'aide, par exemple en définissant la variable d'environnement JAVA_HOME. Pour sélectionner une version d' Adoptium (https:// adoptium.net), définissez par exemple

JAVA_HOME=/Bibliothèque/Java/JavaVirtualMachines/termurin-17.jdk/Contents/Home

[17] Configurez Tk avec --enable-aqua.

[18] Dans le cas peu probable où la version signalée ne démarre pas avec 1.8.0, 11 ou une version ultérieure, vous devrez mettre à jour votre Java.

[19] Pas au moment de la rédaction de cet article pour « arm64 ».

dans config.site. Pour Java 17 depuis https://jdk.java.net/ (qui n'est plus disponible), utilisez

```
JAVA_HOME=/chemin/vers/jdk-17.jdk/Contents/Home
```

Pour une version 'arm64', la première version de Java officiellement prise en charge est la 17. Le moyen actuellement le plus simple d'installer Java est d' Adoptium (https://adoptium.net) (qui appelle l'architecture 'aarch64') : cela s'installe dans un emplacement standard Apple et fonctionne donc avec /usr/bin/java. D'autres versions sont disponibles sur https://www.azul.com/downloads/zulu-community/?os=macos&architecture=arm-64-bit&package=jdk et depuis OpenJDK sur https://jdk.java.net/, pour lesquelles JAVA_HOME devra peut-être être défini à la fois lors de la configuration de R

et au moment de l'exécution.

Pour utiliser Java (en particulier le package binaire rJava (https://CRAN.R-project.org/ package=rJava)) avec une distribution binaire CRAN (« x86_64 ») de R sur macOS « arm64 », installez une version Intel de un Java JRE à partir de l'un des sites liés ci-dessus, puis exécutez sudo R CMD javareconf.

Notez qu'il est nécessaire de définir la variable d'environnement NOAWT sur 1 pour installer la plupart des Packages utilisant Java.

C.3.8 Cadres

La version CRAN de R est installée en tant que framework, qui est sélectionné par l'option

```
./configure --enable-R-framework
```

(Ceci est destiné à être utilisé avec une chaîne d'outils Apple : d'autres peuvent ne pas prendre en charge correctement les frameworks, mais ceux de LLVM le font.)

Il n'est nécessaire que si vous souhaitez créer R pour une utilisation avec la console R.app et implique --enable-R-shlib pour créer R en tant que bibliothèque dynamique. Cette option configure R pour qu'il soit construit et installé en tant que framework appelé R.framework. Le chemin d'installation par défaut de R.framework est /Library/Frameworks mais cela peut être modifié au moment de la configuration en spécifiant l'indicateur --enable-R-framework[=DIR] (ou --prefix) ou au moment de l'installation via

```
make prefix=/where/you/want/R.framework/to/go installer
```

Notez que l'installation en tant que framework n'est pas standard (en particulier dans un emplacement non standard) et que les utilitaires Unix peuvent ne pas la prendre en charge (par exemple, le fichier pkg-config libR.pc sera placé dans un endroit inconnu de pkg-config).

C.3.9 Création de R.app La création

de la console GUI R.app est un projet distinct, utilisant Xcode. Avant de compiler R.app, assurez-vous que la version actuelle de R est installée dans /Library/Frameworks/R.framework et fonctionne en ligne de commande (il peut s'agir d'une installation binaire).

Les sources actuelles peuvent être consultées par

```
svn co https://svn.r-project.org/R-packages/trunk/Mac-GUI
```

et construit en chargeant le projet R.xcodeproj (sélectionnez la cible R et une configuration appropriée), ou depuis la ligne de commande par exemple

```
xcodebuild -target R -version de configuration
```

Voir aussi le fichier INSTALL dans la caisse ou directement sur https://svn.r-project.org/R-packages/trunk/Mac-GUI/INSTALL.

R.app n'a pas besoin d'être installé d'une manière spécifique. La création de R.app aboutit au bundle R.app qui apparaît sous la forme d'une seule icône R. Ce bundle d'applications peut être exécuté n'importe où et il est d'usage de le placer dans le dossier /Applications.

C.3.10 Création de packages binaires Les packages

binaires CRAN macOS sont distribués sous forme d'archives tar avec le suffixe .tgz pour les distinguer des archives tar sources. On peut tarer un package installé existant ou utiliser R CMD INSTALL --build.

Il y a cependant quelques détails importants. • Les

distributions CRAN macOS actuelles sont destinées à Big Sur, il est donc sage de s'assurer que les compilateurs génèrent du code qui s'exécutera sur Big Sur ou une version ultérieure. Avec les compilateurs recommandés
on peut utiliser

```
CC="clang -mmacosx-version-min=11.0"
CXX="clang++ -mmacosx-version-min=11.0"
FC="/opt//gfortran/bin/gfortran -mmacosx-version-min=11.0" ou définissez la variable
```

d'environnement

```
exporter MACOSX_DEPLOYMENT_TARGET=11.0
```

- L'utilisation de l'indicateur -Werror=partial-availability peut aider à déclencher des erreurs de compilation sur la fonction. la nationalité n'est pas à Big Sur.
- Vérifiez que tout code compilé n'est pas lié dynamiquement aux bibliothèques uniquement sur votre machine, par exemple en utilisant otool -L ou objdump -macho -dylibs-used. Cela peut inclure les bibliothèques d'exécution C++ et Fortran sous / opt/R/x86_64/lib ou /opt/R/arm64/lib : on peut utiliser install_name_tool pour les pointer vers les versions du système ou celles livrées avec R, par exemple install_name_tool -change /usr/local/llvm/lib/libc++.1.dylib \ /usr/lib/libc++.1.dylib \ pkg.so

```
install_name_tool -change /opt/gfortran/
lib/gcc/aarch64-apple-darwin20.0/12.2.0/libgfortran.5.dylib \ /Library/Frameworks/R.framework/Resources/lib/
libgfortran.5.dylib \ pkg .donc
```

```
install_name_tool -change /opt/gfortran/
lib/gcc/aarch64-apple-darwin20.0/12.2.0/libquadmath.0.dylib \ /Library/Frameworks/R.framework/Resources/lib/
libquadmath.0.dylib \ pkg .donc
```

(où les détails dépendent des compilateurs et de la version CRAN macOS R).

- Pour le code C++, il est possible que des appels soient générés vers des points d'entrée ne se trouvant pas dans le système / usr/lib/libc++.1.dylib. L'étape précédente permet de tester cela par rapport à la bibliothèque système sur le système d'exploitation de build, mais pas par rapport aux versions antérieures. Il peut être possible de contourner cela en créant des liens statiques vers libc++.a et libc++abi.a par quelque chose comme SHLIB_CXXLD = /usr/local/llvm/bin/

```
clang PKG_LIBS = /usr/local/llvm/lib/libc++.a /usr/local/llvm/lib/
libc++abi.a
```

dans src/Makevars. Il serait également possible de lier statiquement les bibliothèques d'exécution Fortran libgfortran.a et libquadmath.a si le compilateur Fortran avait des versions ultérieures (mais les gfortran 8 à 13 ont tous la version 5).

Les packages binaires CRAN sont construits avec le compilateur Apple sur la plus ancienne version prise en charge de macOS, ce qui évite les deux premières ainsi que tout problème avec les bibliothèques C++.

C.3.11 Construction pour Intel sur 'arm64' Si l'on souhaite

construire R pour Intel sur un Mac Big Sur 'arm64', ajoutez la cible pour les compilateurs C et C++ : CC="clang -arch x86_64

OBJC=$CC
CXX="clang++ -arch x86_64"
FC="/opt//gfortran/bin/gfortran -arch x86_64 -mtune=native -mmacosx-version-min=11"

et installez le compilateur Fortran et le logiciel externe décrit ci-dessus pour les versions Intel (et ayez /opt/R/x86_64/bin avant /opt/R/arm64/bin dans votre chemin).

Pour définir l'architecture correcte (qui sera automatiquement détectée comme aarch64), utilisez quelque chose comme /

path/to/configure --build=x86_64-apple-darwin20

C.3.12 Installateur

Les scripts pour le packaging CRAN de R peuvent être trouvés sous https://svn.r-project.org/ R-dev-web/trunk/QA/Simon/ R4/ : commencez par le fichier README dans ce répertoire.

C.4 FreeBSD

Il y a eu peu de rapports récents sur FreeBSD : il existe un « port » sur https://svnweb.freebsd. org/ports/head/math/. Les versions récentes de FreeBSD utilisent Clang et les en-têtes et le runtime libc++ C++, mais le « port » est configuré pour utiliser GCC.

L'utilisation d'ICU pour le classement et l'option de configuration --with-internal-tzcode sont des solutions de contournement souhaitables.

C.5 OpenBSD Ingo Feinerer

a installé la version R 3.2.2 sur OpenBSD 5.8 arch 'amd64' (leur nom pour 'x86_64').
Les détails de la version (et des correctifs appliqués) se trouvent sur https://cvsweb.openbsd.org/cgi-bin/cvsweb/ports/math/ R/ . (La rétrogradation de l'exigence de zlib vers la version 1.2.3 est contraire à l'avis des développeurs R.)

C.6 Cygwin La version

32 bits n'a jamais fonctionné assez bien pour réussir le make check de R, et le support résiduel des expériences précédentes a été supprimé dans R 3.3.0.

La version 64 bits n'a jamais été prise en charge.

C.7 Nouvelles plates-formes Il existe

un certain nombre de sources de problèmes lors de l'installation de R sur une nouvelle plate-forme matérielle/OS.
Ceux-ci inclus

Arithmétique à virgule flottante : R nécessite une arithmétique conforme à la norme CEI 60559, également connue sous le nom d' IEEE 754. Cela impose l'utilisation du plus et du moins l'infini et du NaN (pas un nombre) ainsi que des détails spécifiques d'arrondi. Bien que presque tous les FPU actuels puissent prendre en charge cela, la sélection d'un tel support peut s'avérer pénible. Le problème est qu'il n'y a pas d'accord sur la manière de définir le comportement de signalisation ; Sun/Sparc, SGI/IRIX et Linux 'ix86' ne nécessitent aucune action particulière, FreeBSD nécessite un appel à (la macro) fpsetmask(0) et OSF1 exigeait que le calcul soit effectué avec un indicateur -ieee_ with_inexact, etc. Avec les compilateurs Intel sur 32 Sur les machines Intel 64 bits et 64 bits, il faut explicitement désactiver les modes de mise à zéro et de dénormalisation à zéro. Certains processeurs ARM, notamment A12Z et M1 (Apple Silicon), utilisent par défaut le mode runfast, qui inclut le rinçage à zéro et le nan par défaut et doivent donc être désactivés. Avec le mode nan par défaut, la charge utile NaN utilisée pour la représentation des valeurs numériques NA est perdue même lors d'opérations simples avec des valeurs finies. Sur une nouvelle plateforme, vous devez découvrir la recette magique et ajouter du code pour la faire fonctionner. Cela peut souvent être fait via le fichier config.site qui réside dans le répertoire de niveau supérieur.

Méfiez-vous de l'utilisation de niveaux d'optimisation élevés, du moins au début. Sur de nombreux compilateurs, cela réduit le degré de conformité au modèle IEEE . Par exemple, en utilisant -fast sur les compilateurs Oracle

a provoqué une définition incorrecte du NaN de R, et -ffast-math de gcc et -Ofast de clang ont donné des résultats incorrects.

Objets partagés : il semble y avoir très peu d'accord entre les plates-formes sur ce qui doit être fait pour créer des objets partagés. il existe de nombreuses combinaisons différentes d'indicateurs pour les compilateurs et les chargeurs. GNU libtool ne peut pas (encore) être utilisé, car il ne prend actuellement pas entièrement en charge Fortran : il faudrait un wrapper shell pour cela). La technique que nous utilisons consiste d'abord à interroger le système X window sur ce qu'il fait (en utilisant xmkmf), puis à remplacer cela dans les situations que nous connaissons mieux (pour les outils de la collection de compilateurs GNU et/ou les plates-formes que nous connaissons) .
Cela fonctionne généralement, mais vous devrez peut-être remplacer manuellement les résultats. L'analyse des entrées manuelles pour cc et ld révèle généralement l'incantation correcte. Une fois que vous connaissez la recette, vous pouvez modifier le fichier config.site (en suivant les instructions qu'il contient) afin que le build utilise ces options.

Il semble que gcc 3.4.x et versions ultérieures de Linux 'ix86' aient vaincu les tentatives du code LA-PACK pour éviter les calculs entièrement dans des registres à précision étendue, donc le fichier src/modules/lapack/dlamc.f devra peut-être être compilé sans optimisation. ou avec des drapeaux supplémentaires. Définissez la variable de configuration SAFE_FFLAGS sur les indicateurs à utiliser pour ce fichier.

Si vous parvenez à faire fonctionner R sur une nouvelle plate-forme, veuillez nous en informer afin que nous puissions modifier les procédures de configuration pour inclure cette plateforme.

Si vous rencontrez des difficultés pour faire fonctionner R sur votre plateforme, n'hésitez pas à utiliser la liste de diffusion « R-devel » pour poser des questions. Nous avons eu pas mal de pratique dans le portage de R sur de nouvelles plates-formes.

Fonction et index variable

C

je

M

R.

DANS

Index des concepts

Index des variables d'environnement