**Step 1: Install Rtools40**

Installing [Rtools40](https://cran.r-project.org/bin/windows/Rtools/) is straightforward. Make sure to download the 64 bit version and add the PATH variable as described on the website.

**Step 2: Install the necessary support software**

Install [MiKTeX](https://miktex.org/) and [Inno Setup](https://jrsoftware.org/isinfo.php) to their default locations if they are not already. Install [qpdf](http://qpdf.sourceforge.net/) anywhere if you wish to use it.

**Step 3: Obtaining the R-windows build scripts**

Scripts are attached on the repository to refer.

**Step 4: Create blas patch**

The build uses the same process which works in R3+; it hooks OpenBLAS into the build via the ATLAS variables in the Makefiles. However, the new system downloads the R source code as part of the build process, so changes to src/extra/blas/Makefile.win are made using patch file instead of manual edits. Therefore, save the following text as a file named blas.diff in the same subdirectory as the extracted/cloned files.

--- a/src/extra/blas/Makefile.win

+++ b/src/extra/blas/Makefile.win

@@ -12,7 +12,7 @@

../../../$(BINDIR)/Rblas.dll: blas00.o ../../gnuwin32/dllversion.o

@$(ECHO) -------- Building $@ --------

$(DLL) -shared $(DLLFLAGS) -o $@ $^ Rblas.def \

- -L../../../$(IMPDIR) -lR -L"$(ATLAS\_PATH)" -lf77blas -latlas

+ -L../../../$(IMPDIR) -lR -L"$(ATLAS\_PATH)" -lopenblas

else

../../../$(BINDIR)/Rblas.dll: blas.o cmplxblas.o ../../gnuwin32/dllversion.o

@$(ECHO) -------- Building $@ --------

**Step 5: Adjust existing files**

Make the following changes to the build files to ensure that OpenBLAS is pulled from pacman (the **pac**kage **man**ager, not the Namco character) and that the proper libraries are accessed at the right times.

**full-build.sh**

Add the highlighted line to full-build.sh after the call to cairo, tk, and curl. This is where other packages like nlopt or xml will be added eventually too.

pacman -S --needed --noconfirm mingw-w64-{i686,x86\_64}-{cairo,tk,curl}

pacman -S --needed --noconfirm mingw-w64-{i686,x86\_64}-openblas

[**MkRules.local.in**](http://MkRules.local.in)

Edit EOPTS (extra optimization flags) in [MkRules.local.in](http://MkRules.local.in) as in prior versions of R. Personally, I use -march=native -pipe as I build bespoke R installers for each machine I use. To use the same installer on different computers I recommend -mtune=generic -pipe. See the [GCC 8.3 documentation](https://gcc.gnu.org/onlinedocs/gcc-8.3.0/gcc/x86-Options.html#x86-Options) for more on optimization flags.  
Next, add and edit the following lines under #Enable features:

USE\_ATLAS = YES

ATLAS\_PATH="/mingw$(WIN)/lib/"

This tells the build to use the OpenBLAS libraries and where to find both the 32 and 64 bit versions.  
To use qpdf, add the following under # For building docs/installer, substituting the proper path.

QPDF = /path/to/qpdf

**PKGBUILD**

Adjusting this file is more complicated. Add the highlighted lines below to PKGBUILD in their appropriate locations:

# Maintainer: Jeroen Ooms

\_realname=r-installer

pkgbase=${\_realname}

pkgname="${\_realname}"

pkgver=4.0.9000

pkgrel=1

pkgdesc="The R Programming Language"

arch=('any')

makedepends=("${MINGW\_PACKAGE\_PREFIX}-bzip2"

"${MINGW\_PACKAGE\_PREFIX}-gcc"

"${MINGW\_PACKAGE\_PREFIX}-gcc-fortran"

"${MINGW\_PACKAGE\_PREFIX}-cairo"

"${MINGW\_PACKAGE\_PREFIX}-curl"

"${MINGW\_PACKAGE\_PREFIX}-icu"

"${MINGW\_PACKAGE\_PREFIX}-libtiff"

"${MINGW\_PACKAGE\_PREFIX}-libjpeg"

"${MINGW\_PACKAGE\_PREFIX}-libpng"

"${MINGW\_PACKAGE\_PREFIX}-pcre2"

"${MINGW\_PACKAGE\_PREFIX}-tcl"

"${MINGW\_PACKAGE\_PREFIX}-tk"

"${MINGW\_PACKAGE\_PREFIX}-xz"

"${MINGW\_PACKAGE\_PREFIX}-zlib"

"${MINGW\_PACKAGE\_PREFIX}-openblas"

"texinfo"

"texinfo-tex"

"sed")

options=('staticlibs')

license=("GPL")

url="<https://www.r-project.org/>"

# Default source is R-devel (override via $rsource\_url)

source=(R-source.tar.gz::"${rsource\_url:-<https://cran.r-project.org/src/base-prerelease/R-devel.tar.gz>}"

<https://curl.haxx.se/ca/cacert.pem>

[MkRules.local.in](http://MkRules.local.in)

shortcut.diff

create-tcltk-bundle.sh

blas.diff)

# Automatic untar fails due to embedded symlinks

noextract=(R-source.tar.gz)

sha256sums=('SKIP'

'SKIP'

'SKIP'

'SKIP'

'SKIP'

'SKIP')

prepare() {

# Verify that InnoSetup is installed

INNOSETUP="C:/Program Files (x86)/Inno Setup 6/ISCC.exe"

msg2 "Testing for $INNOSETUP"

test -f "$INNOSETUP"

"$INNOSETUP" 2>/dev/null || true

# Put pdflatex on the path (assume Miktex 2.9)

msg2 "Checking if pdflatex and texindex can be found..."

export PATH="$PATH:/c/progra~1/MiKTeX 2.9/miktex/bin/x64"

pdflatex --version

texindex --version

# Extract tarball with symlink workarounds

msg2 "Extracting R source tarball..."

rm -rf ${srcdir}/R-source

mkdir -p ${srcdir}/R-source

MSYS="winsymlinks:lnk" tar -xf ${srcdir}/R-source.tar.gz -C ${srcdir}/R-source --strip-components=1

cd "${srcdir}/R-source"

# Ship the CA bundle

cp "${srcdir}/cacert.pem" etc/curl-ca-bundle.crt

# Ship the TclTk runtime bundle

msg2 "Creating the TclTk runtime bundle"

mkdir -p Tcl/{bin,bin64,lib,lib64}

${srcdir}/create-tcltk-bundle.sh

# Add your patches here

patch -Np1 -i "${srcdir}/shortcut.diff"

patch -Np1 -i "${srcdir}/blas.diff"

}

build() {

msg2 "Copying source files for 32-bit build..."

rm -Rf ${srcdir}/build32

MSYS="winsymlinks:lnk" cp -Rf "${srcdir}/R-source" ${srcdir}/build32

# Build 32 bit version

msg2 "Building 32-bit version of base R..."

cd "${srcdir}/build32/src/gnuwin32"

sed -e "s|@win@|32|" -e "s|@texindex@||" -e "s|@home32@||" "${srcdir}/[MkRules.local.in](http://MkRules.local.in)" > MkRules.local

#make 32-bit SHELL='sh -x'

make 32-bit

# Build 64 bit + docs and installers

msg2 "Building 64-bit distribution"

cd "${srcdir}/R-source/src/gnuwin32"

TEXINDEX=$(cygpath -m $(which texindex))

sed -e "s|@win@|64|" -e "s|@texindex@|${TEXINDEX}|" -e "s|@home32@|${srcdir}/build32|" "${srcdir}/[MkRules.local.in](http://MkRules.local.in)" > MkRules.local

make distribution

}

check(){

# Use cloud mirror for CRAN unit test

#export R\_CRAN\_WEB="<https://cran.rstudio.com>"

# Run 64 bit checks in foreground

cd "${srcdir}/R-source/src/gnuwin32"

echo "===== 64 bit checks ====="

make check-all

}

package() {

# Derive output locations

REVISION=$((read x; echo ${x:10}) < "${srcdir}/R-source/SVN-REVISION")

CRANDIR="${srcdir}/R-source/src/gnuwin32/cran"

# This sets TARGET variable

$(sed -e 's|set|export|' "${CRANDIR}/target.cmd")

# Copy CRAN release files

cp "${srcdir}/R-source/SVN-REVISION" "${pkgdir}/SVN-REVISION.${target}"

cp "${CRANDIR}/NEWS.${target}.html" ${pkgdir}/

cp "${CRANDIR}/README.${target}" ${pkgdir}/

# Determine which webpage variant to ship from target (for example "R-3.4.1beta")

case "$target" in

\*devel|\*testing)

cp "${CRANDIR}/rdevel.html" "${pkgdir}/"

;;

\*patched|\*alpha|\*beta|\*rc)

cp "${CRANDIR}/rpatched.html" "${pkgdir}/"

cp "${CRANDIR}/rtest.html" "${pkgdir}/"

;;

R-4\*)

cp "${CRANDIR}/index.html" "${pkgdir}/"

cp "${CRANDIR}/md5sum.txt" "${pkgdir}/"

cp "${CRANDIR}/rw-FAQ.html" "${pkgdir}/"

cp "${CRANDIR}/release.html" "${pkgdir}/"

REVISION="$target"

;;

\*)

echo "Unknown release type: $target"

exit 1

;;

esac

# Helper for appveyor script

echo "set revision=${REVISION}" >> "${CRANDIR}/target.cmd"

cp "${CRANDIR}/target.cmd" ${pkgdir}/

}

**Step 6: Build R**

Launch Rtools40 via msys.exe which creates a shell window. Navigate to the subdirectory where the build files are using Unix style. For example, if the files are in “c:\R\R40”, navigate to “/c/R/R40”. Invoke the build via “./full-build.sh” which will probably run for hours. The process updates all the necessary components, builds both the 32 bit and 64 bit versions of R using OpenBLAS, checks the results for each version, and packages the build into an executable installer file. When done, the desired R-devel-win.exe will be either in the above subdirectory or in /src/R-source/src/gnuwin32/installer, if not both. This the executable which installs R 4+ for Windows with OpenBLAS!

**Final notes, for now**

Installing and using this modified version of R should pose no problems, nor should installing most packages from source. There are some packages which rely on compiled libraries—such as XML or nlopt—for which extra steps are needed to build from source. However, these are few and may be installed as binaries for now. I’d appreciate thoughts and corrections in the comments; good luck!