APL64 ⎕RNET System Function

Contents

Summary

□RNET

The APL64 system function □RNET is a .Net interface to the R statistical and graphics toolkit. □RNET is cross-platform, and uses the x64 R-engine for greater capacity and performance.

There is only one instance of the R engine when using □RNET in an instance of APL64. The R-Engine is automatically disposed when the APL64 instance ends. □RNET is available in the developer and runtime versions of APL64.

Suggestions for installing R are available here.

Notes On Examples

Creating the R instance in APL64

The first time the □RNET system function is used in an APL64 instance, the □RNET 'InitR' action must be used to specify the installation folder of the x64 R version installed on the workstation.

The examples may not illustrate the initial, required use of the □RNET 'InitR' action. For examples which illustrate the □RNET 'InitR' action, the installationPathOfR may need to be modifed to reflect the installation of R on the workstation.

R Packages

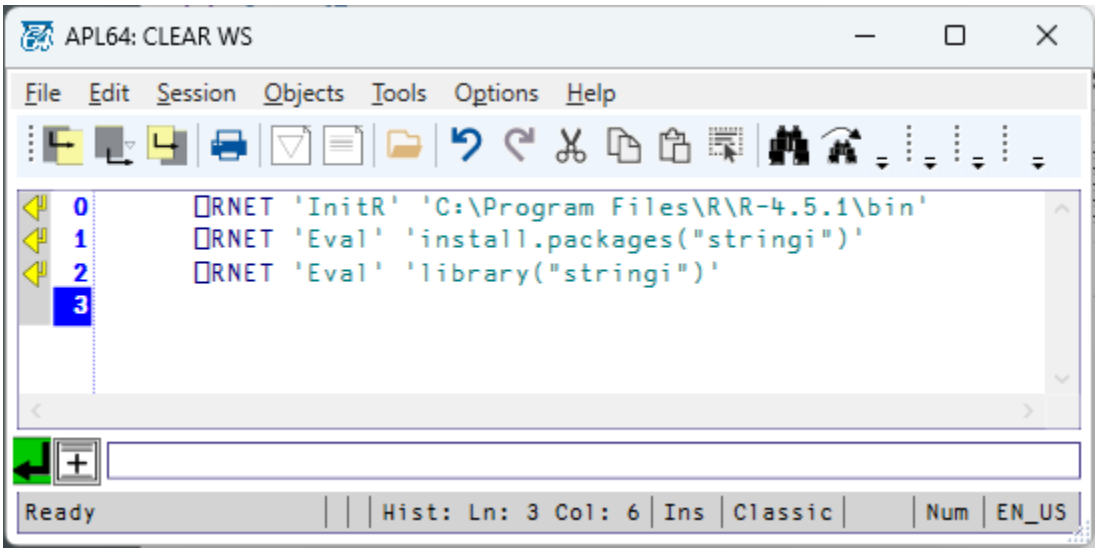
R packages are extensions to the R toolkit. R packages are installed using the R expression install.Packages(packageName). The are additional arguments to install.Packages() to specify the web server source. Using install.Packages() may present a list of web servers where the R-package is available. Installed R packages are loaded using R expression library(packageName).

Some examples require that specific R packages are installed and loaded, but the R expressions to install and load those packages may not be included in the example source code.

It may be necessary to run APL64 with elevated credentials, to successfully install an R-package.

Example:

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Eval' 'install.packages("stringi")'
⎕RNET 'Eval' 'library("stringi")'
```



⎕RNET Syntax

⎕RNET is a monadic system function.

[result] ← ⎕RNET action [actionArgumentsArray]

⎕RNET Actions

The ⎕RNET action text is case-insensitive.

R symbols are case sensitive

CreateDataFrame Actions

An R-Dataframe is an R data structure which is analogous to an APL64 rank-2 matrix.

- The ⎕RNET 'Cdf' action will create an R-dataframe using the supplied APL64 matrix. The APL64 programmer can optionally supply row and column names. If row and column names are not supplied, they will be created by this action.
- The ⎕RNET 'CdfNm' action will create an R-dataframe using the supplied APL64 matrix. The row and column names of the dataframe are assumed to be the first row and first column of the supplied (nested) APL64.

The ⎕RNET actions, Cdf and CdfNm to create a dataframe have these arguments:

- dfName is a text vector or string scalar which will be the name of the resulting dataframe within the R environment.
- dataMatrix is an APL64 rank-2 array.
- checkRows is a Boolean scalar. If 1, R checks for consistency in the length of the rows and their names, ensuring all columns have the same number of elements.
- checkCols is a Boolean scalar. If 1, R checks the names of the variables (columns) to ensure they are syntactically valid and unique.
- stringAsFactors is a Boolean scalar. If 1, text data in the input will be converted to R-factors.

For any column of the APL64 data matrix used to create the dataframe, the elements of that column must have the same R data type.

All APL64 text data type elements are converted to the .Net string data type which is compatible with the R text data type.

When the dataframe is created, each column of the APL64 source data matrix is independently checked for its data type to determine Boolean data type columns, if any.

APL and .Net store arrays in row-major order. R stores arrays in column-major order. This behavior of R may cause a display of dataframe to appear transposed compared to the source APL or .Net array. The rows of an APL or .Net array are considered the columns of an R array.

Cdf: Create DataFrame From Data Matrix

Syntax:

⎕RNET 'Cdf' dfName dataMat checkRows checkCols stringAsFactors [colNames] [rowNames]

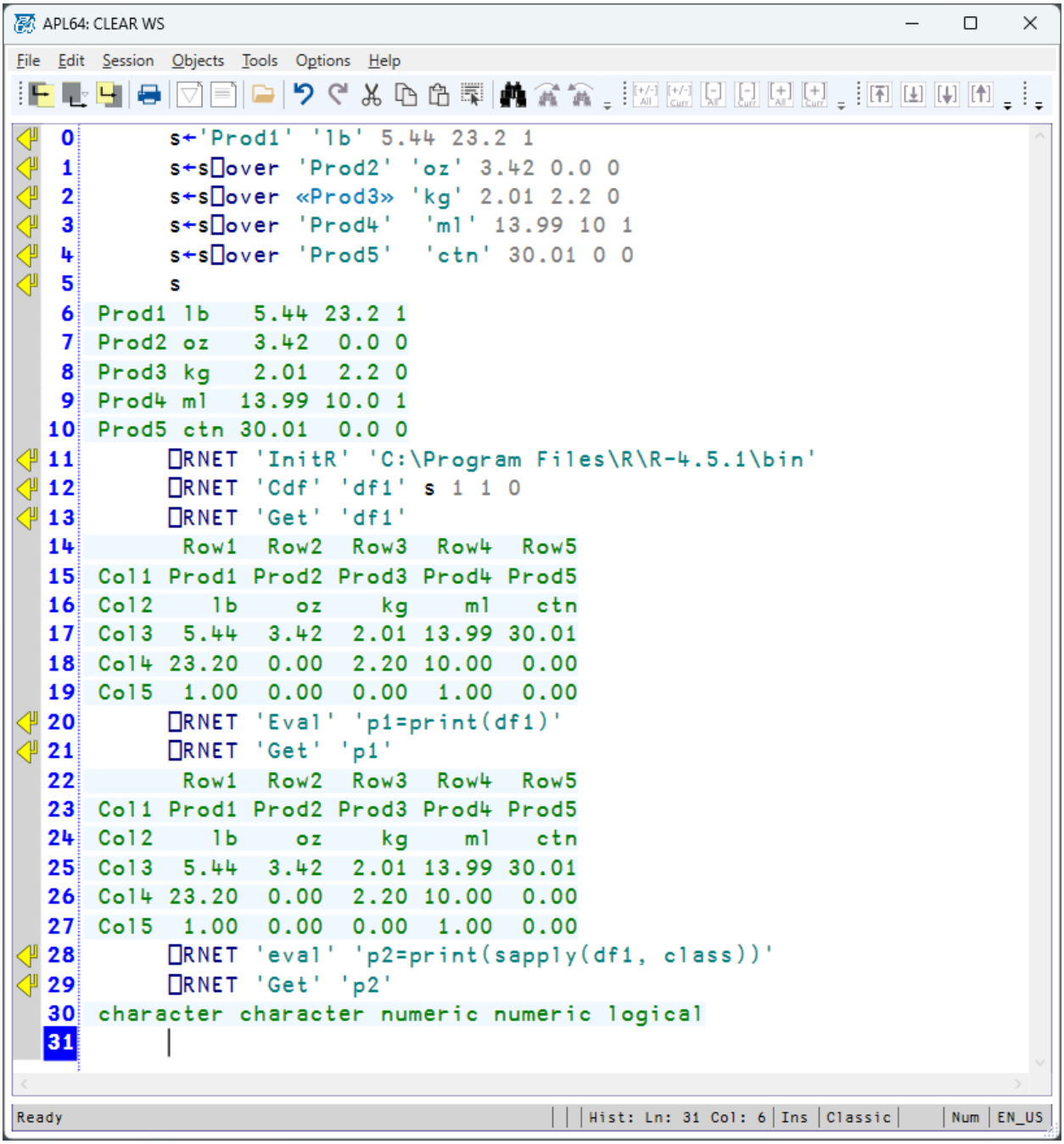
dataMat is an APL rank-2 matrix. The element values in this matrix are not used to create row or column names.

colNames is an optional argument. If provided, it is an array of text-valued elements. If not provided, default column names will be created: Col1, …

rowNames is an optional argument. If provided, it is an array of text-valued elements. If not provided, default column names will be provided: Row1, …

Example:

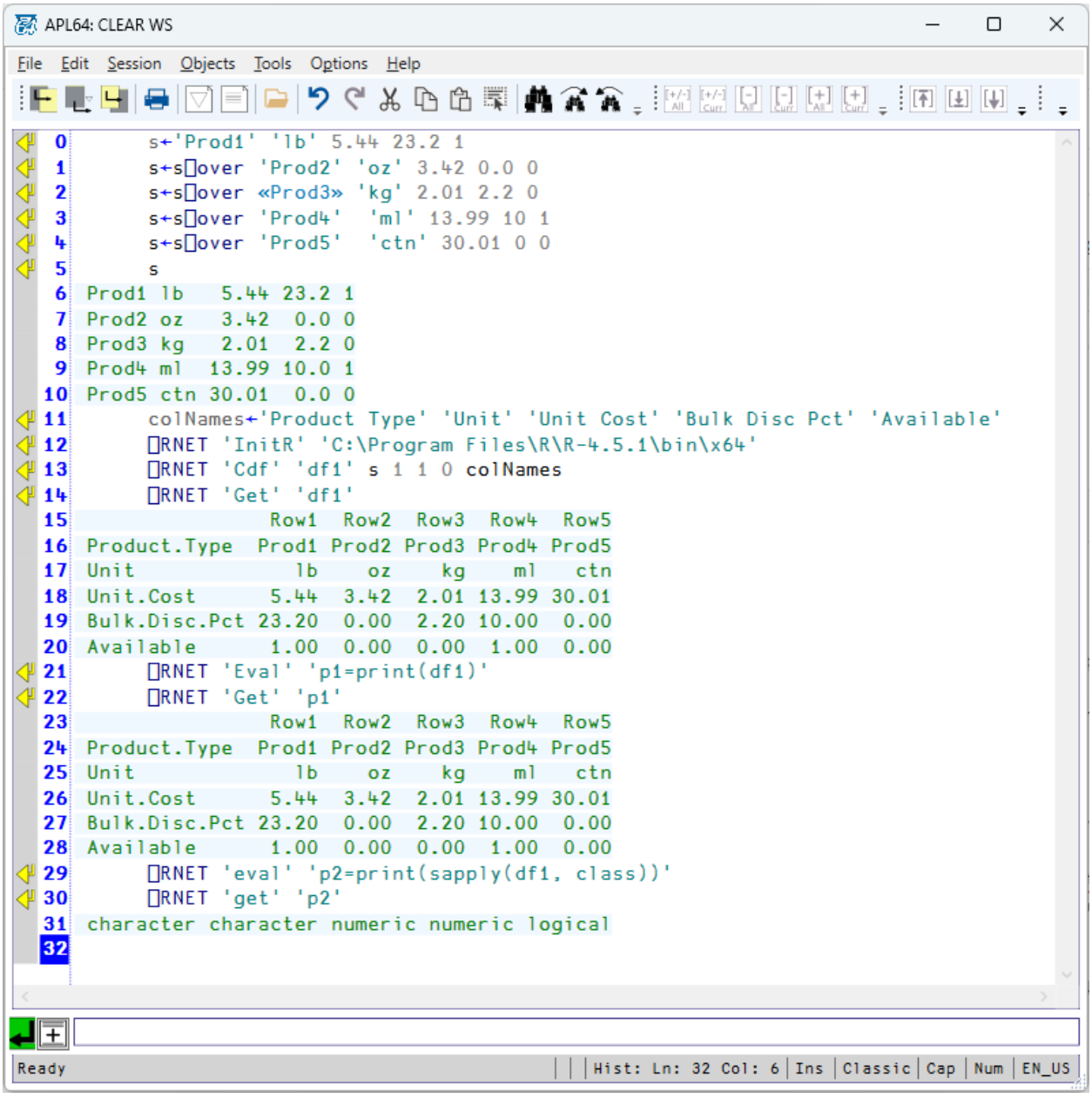
```
s←'Prod1' 'lb' 5.44 23.2 1
s←s□over 'Prod2' 'oz' 3.42 0.0 0
s←s□over «Prod3» 'kg' 2.01 2.2 0
s←s□over 'Prod4'  'ml' 13.99 10 1
s←s□over 'Prod5'  'ctn' 30.01 0 0
s
□RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
□RNET 'Cdf' 'df1' s 1 1 0
□RNET 'Get' 'df1'
□RNET 'Eval' 'p1=print(df1)'
□RNET 'Get' 'p1'
□RNET 'eval' 'p2=print(sapply(df1, class))'
□RNET 'Get' 'p2'
```



Example:

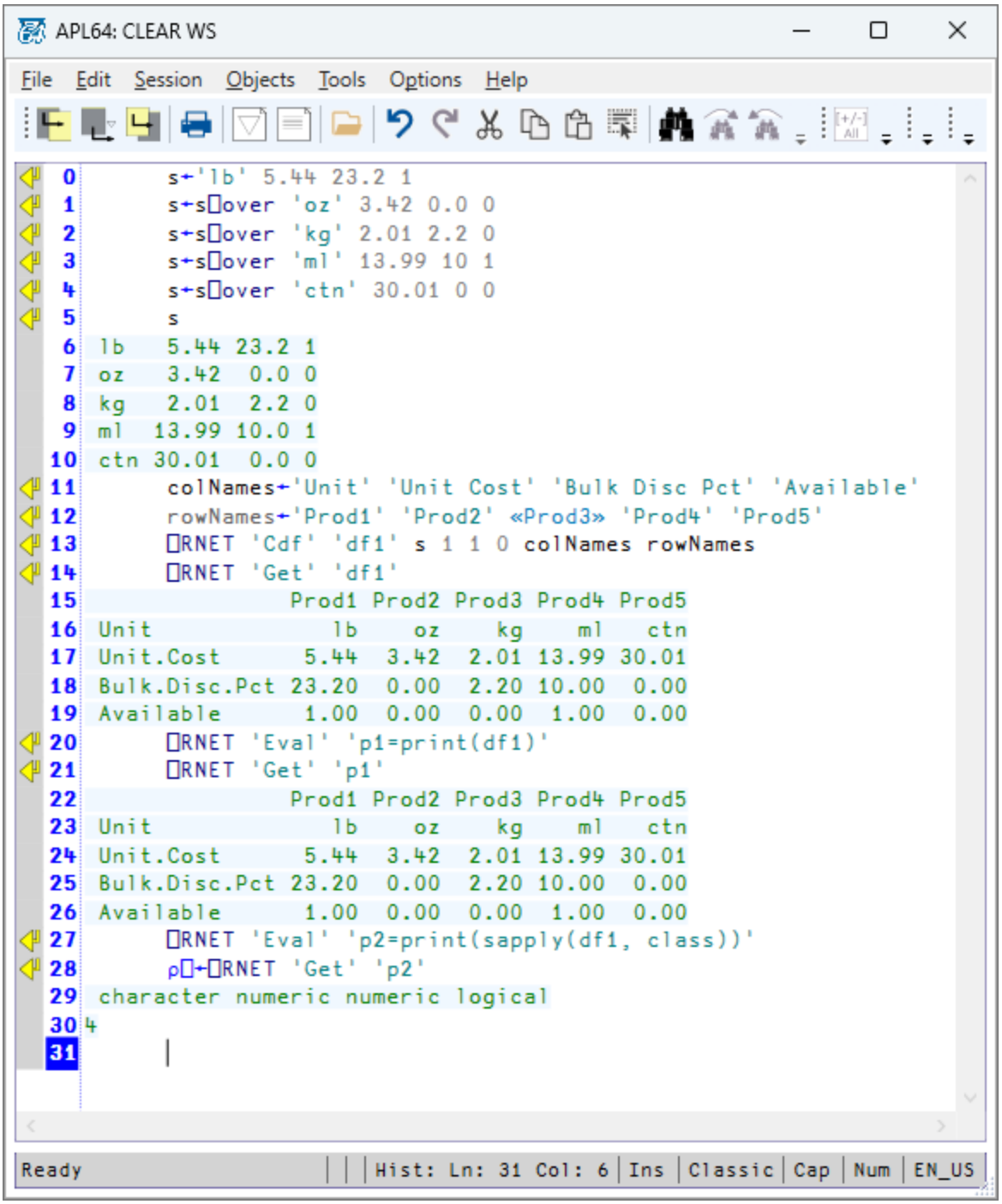
```
s←'Prod1' 'lb' 5.44 23.2 1
s←s⍁over 'Prod2' 'oz' 3.42 0.0 0
s←s⍁over «Prod3» 'kg' 2.01 2.2 0
s←s⍁over 'Prod4'  'ml' 13.99 10 1
s←s⍁over 'Prod5'  'ctn' 30.01 0 0
```

```
s
colNames←'Product Type' 'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
]RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
]RNET 'Cdf' 'df1' s 1 1 0 colNames
]RNET 'Get' 'df1'
]RNET 'Eval' 'p1=print(df1)'
]RNET 'Get' 'p1'
]RNET 'eval' 'p2=print(sapply(df1, class))'
]RNET 'get' 'p2'
```



Example:

```
s←'lb' 5.44 23.2 1
s←s⍠over 'oz' 3.42 0.0 0
s←s⍠over 'kg' 2.01 2.2 0
s←s⍠over 'ml' 13.99 10 1
s←s⍠over 'ctn' 30.01 0 0
s
colNames←'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
rowNames←'Prod1' 'Prod2' «Prod3» 'Prod4' 'Prod5'
⍠RNET 'Cdf' 'df1' s 1 1 0 colNames rowNames
⍠RNET 'Get' 'df1'
⍠RNET 'Eval' 'p1=print(df1)'
⍠RNET 'Get' 'p1'
⍠RNET 'Eval' 'p2=print(sapply(df1, class))'
ρ⍠←⍠RNET 'Get' 'p2'
```



CdfNm: Create DataFrame From Nested Matrix

Syntax: ⎕RNET 'CDF' dfName nestedMat checkRows checkCols stringAsFactors

dfName is a text vector or string scalar

nestedMat is an APL rank-2 matrix. The first row and first columns of nestedMatrix will become the row and column names in the resulting dataframe. The 'inner' rows and columns of

nestedMatrix will become the data elements of the resulting dataframe determine Boolean data type columns, if any.

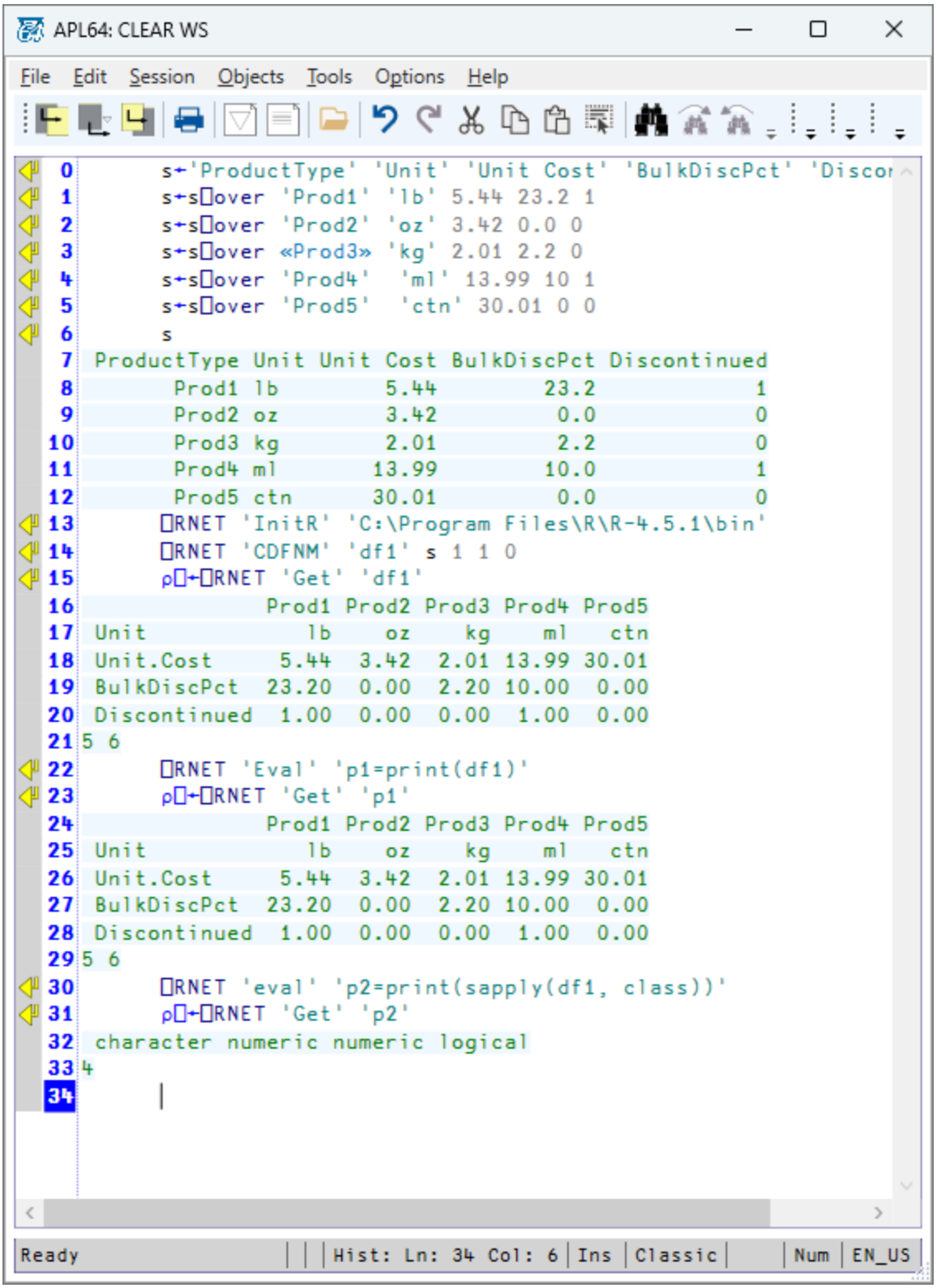
checkRows is a Boolean scalar. If 1, R checks for consistency in the length of the rows and their names, ensuring all columns have the same number of elements.

checkCols is a Boolean scalar. If 1, R checks the names of the variables (columns) to ensure they are syntactically valid and unique.

stringAsFactors is a Boolean scalar. If 1, text data in the input will be converted to R-factors.

Example:

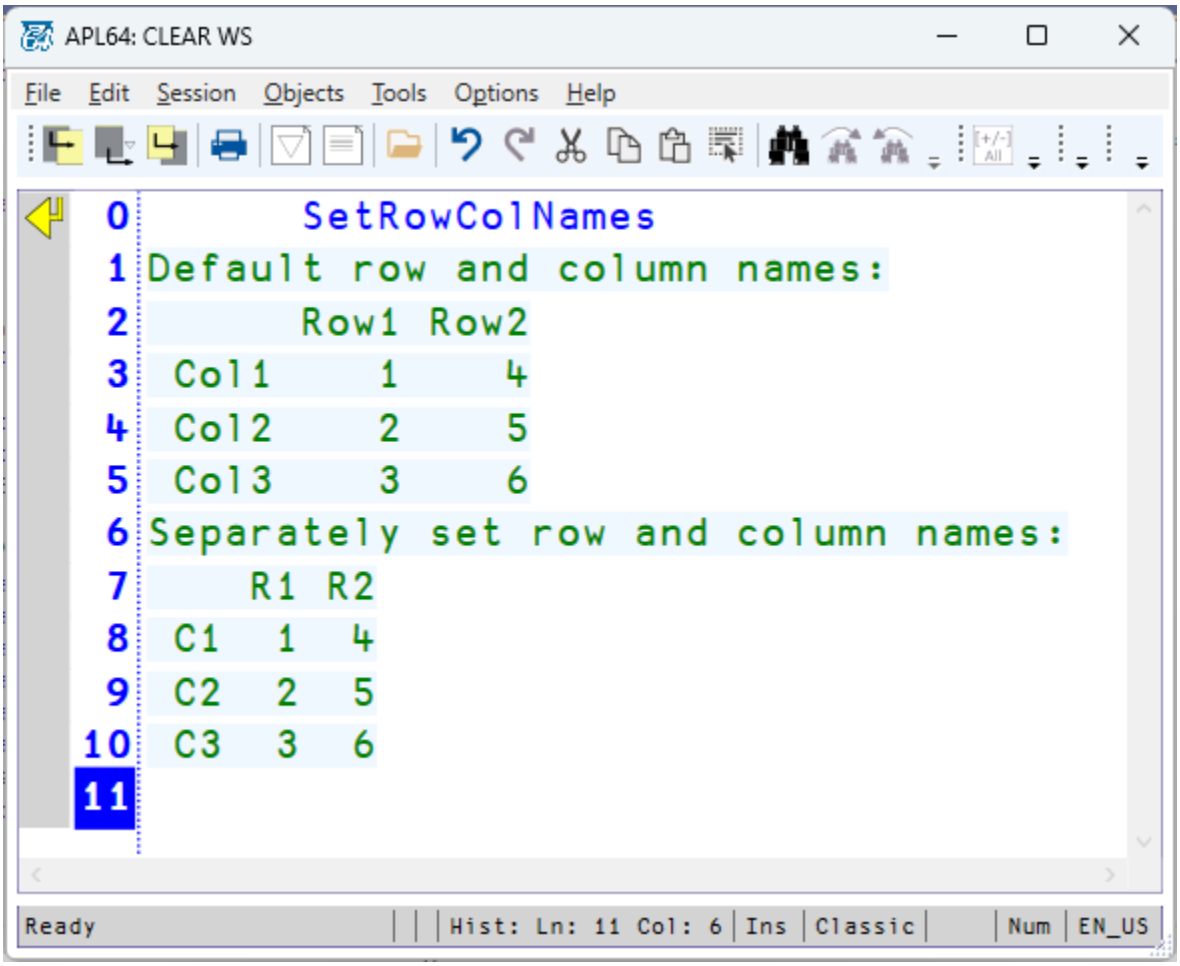
```
s←'ProductType' 'Unit' 'Unit Cost' 'BulkDiscPct' 'Discontinued'
s←s⍪over 'Prod1' 'lb' 5.44 23.2 1
s←s⍪over 'Prod2' 'oz' 3.42 0.0 0
s←s⍪over «Prod3» 'kg' 2.01 2.2 0
s←s⍪over 'Prod4'  'ml' 13.99 10 1
s←s⍪over 'Prod5'  'ctn' 30.01 0 0
s
⍪RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⍪RNET 'CDFNM' 'df1' s 1 1 0
ρ⍪←⍪RNET 'Get' 'df1'
⍪RNET 'Eval' 'p1=print(df1)'
ρ⍪←⍪RNET 'Get' 'p1'
⍪RNET 'eval' 'p2=print(sapply(df1, class))'
ρ⍪←⍪RNET 'Get' 'p2'
```



Set DataFrame Row or Column Names Separately

```
SetRowColNames
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
```

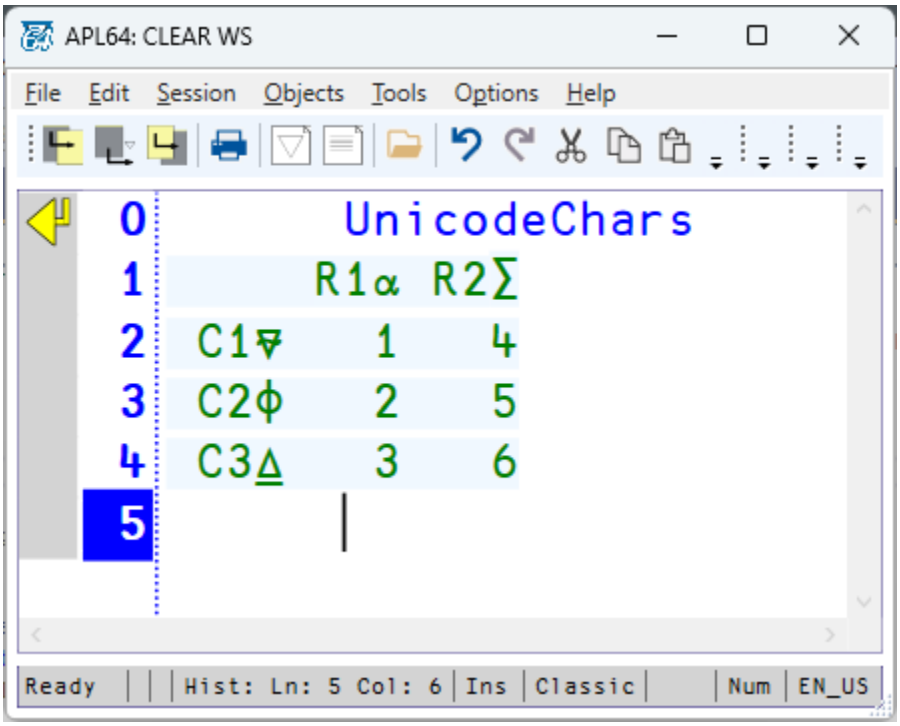
```
Y←2 3⍴⍳6
⎕RNET 'Cdf' 'df' Y 1 1 0
'Default row and column names:'
⎕RNET 'Get' 'df'
⎕RNET 'Set' 'colNames' ('C1' 'C2' 'C3')
⎕RNET 'Eval' 'colnames(df) = colNames'
⎕RNET 'Set' 'rowNames' ('R1' 'R2')
⎕RNET 'Eval' 'rownames(df) = rowNames'
'Separately set row and column names:'
⎕RNET 'Get' 'df'
```



Set DataFrame Row or Column Names with Unicode Glyphs

```
UnicodeChars
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
Y←2 3⍴⍳6
⎕RNET 'Cdf' 'df' Y 1 1 0
⎕RNET 'Eval' 'colnames(df) = c("C1⍒", "C2⌽", "C3⍋")'
```

```
⎕RNET 'Eval' 'rownames(df) = c("R1α", "R2\u2211")'
⍝ U2211: ∑
⎕RNET 'Get' 'df'
```



Evaluate

Synonyms: Eval, Ev

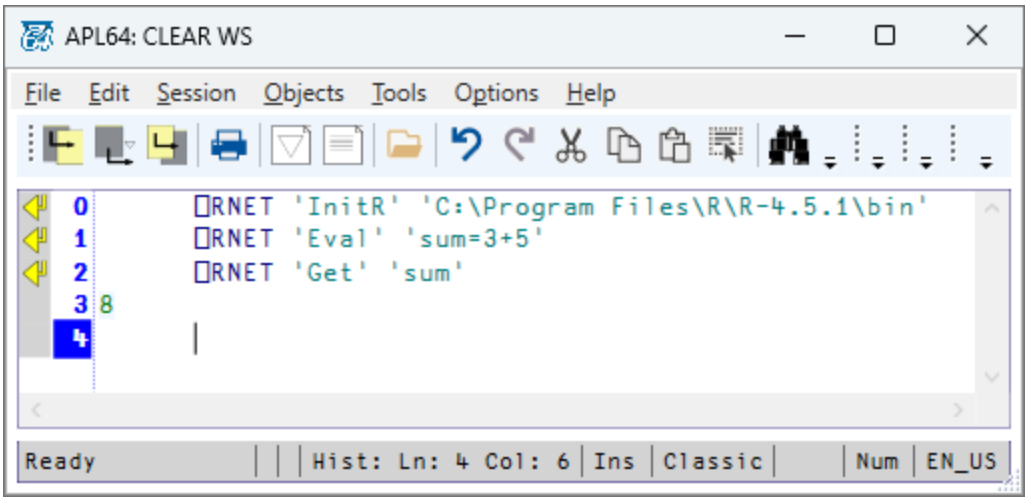
The Evaluate action will cause the R-engine to evaluate the 'textToEvaluate' within the R environment. The 'textToEvaluate' can be an APL64 multi-line string script.

Evaluate returns no result. To obtain the result of an R-expression, assign it to an R-variable, and use the ⎕RNET 'Get' action to obtain the result value in APL64. Some R data types have no representation in APL.

Syntax: ⎕RNET 'Evaluate' textToEvaluate [subValue1] [subValue2]…

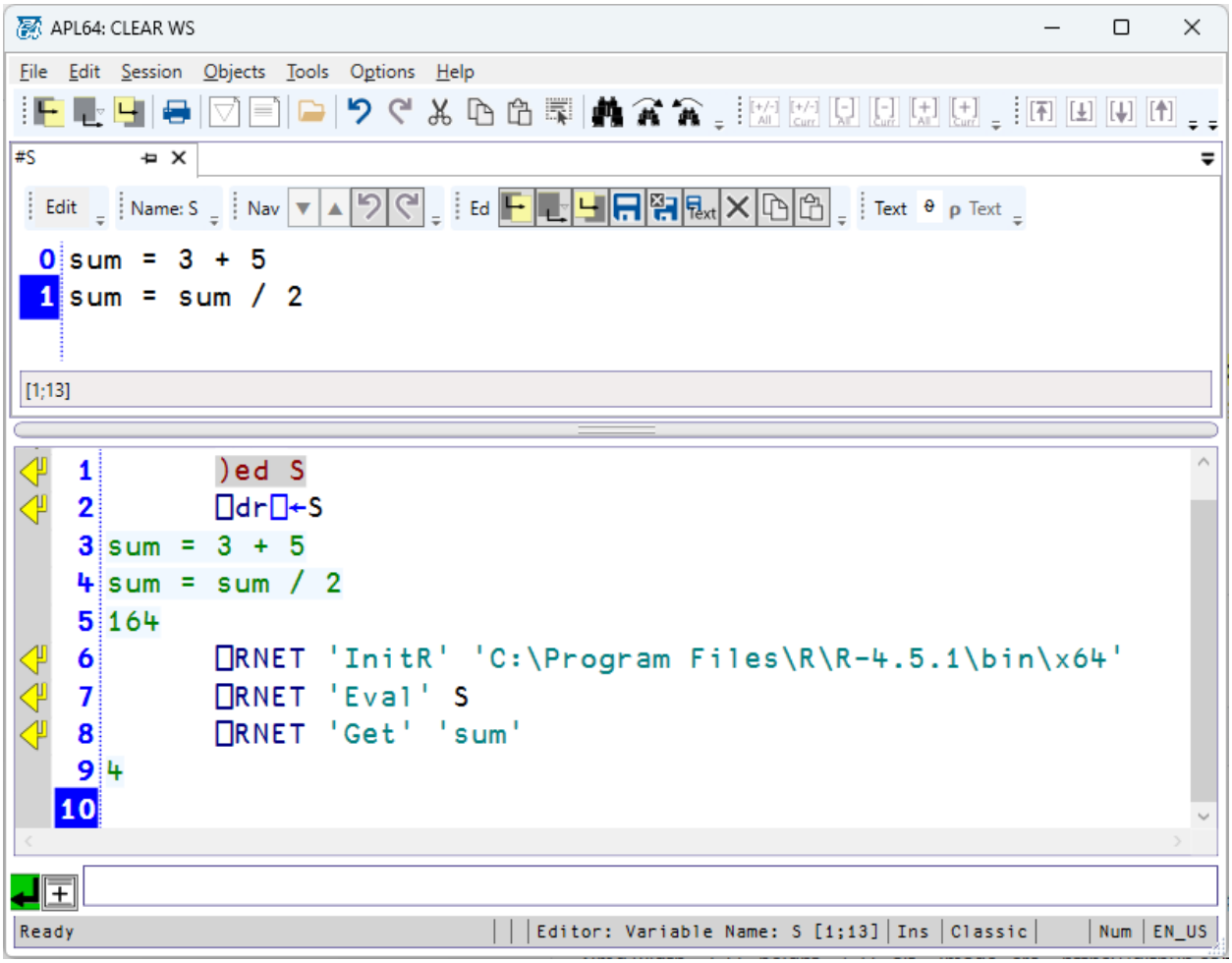
Example:

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Eval' 'sum=3+5'
⎕RNET 'Get' 'sum'
```

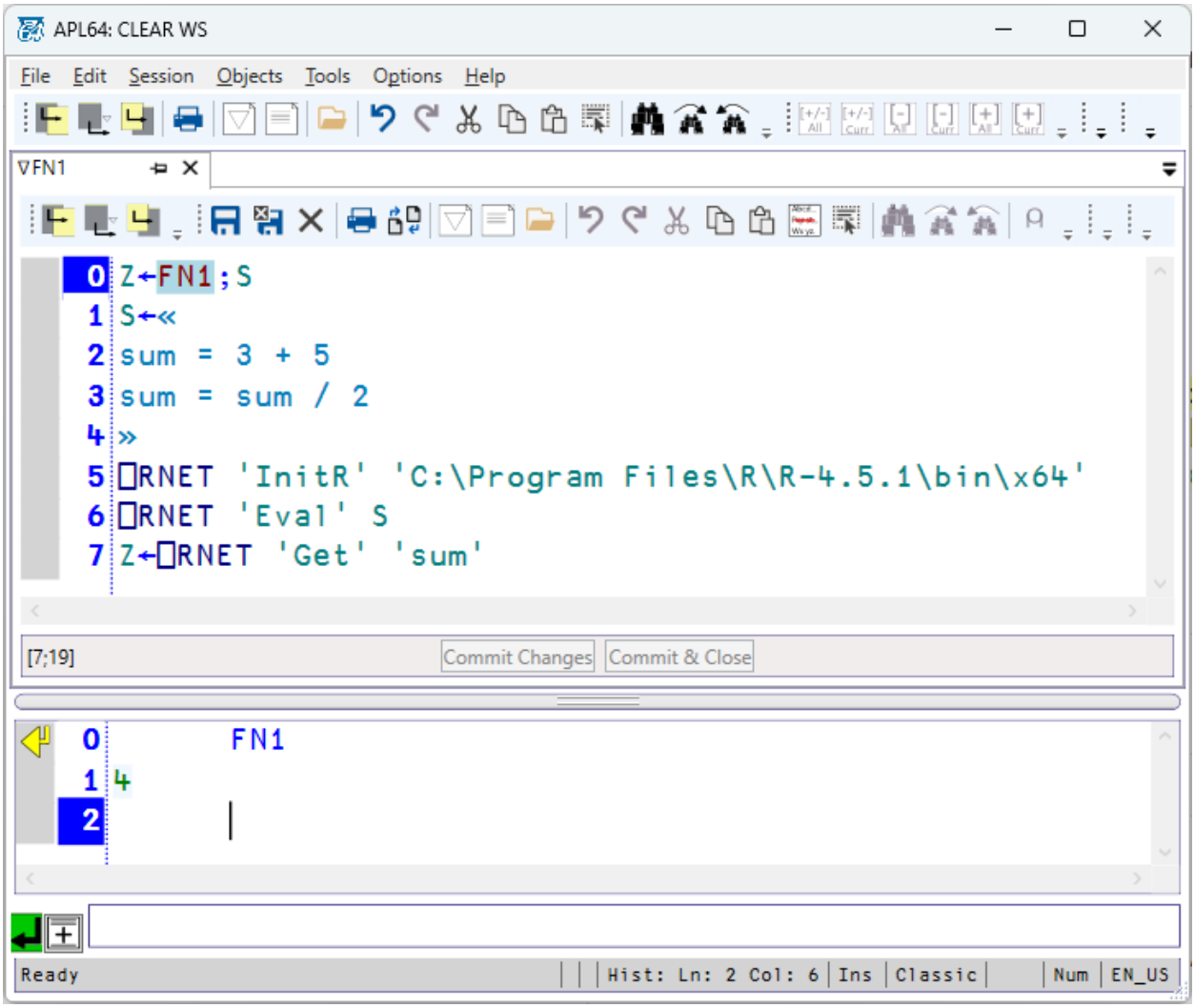


Evaluate with an APL64 multi-line string script

Example: Script created using the eAPL64 variable editor and used *ad hoc*



Example: Script created and used within an APL64 programmer-designed function

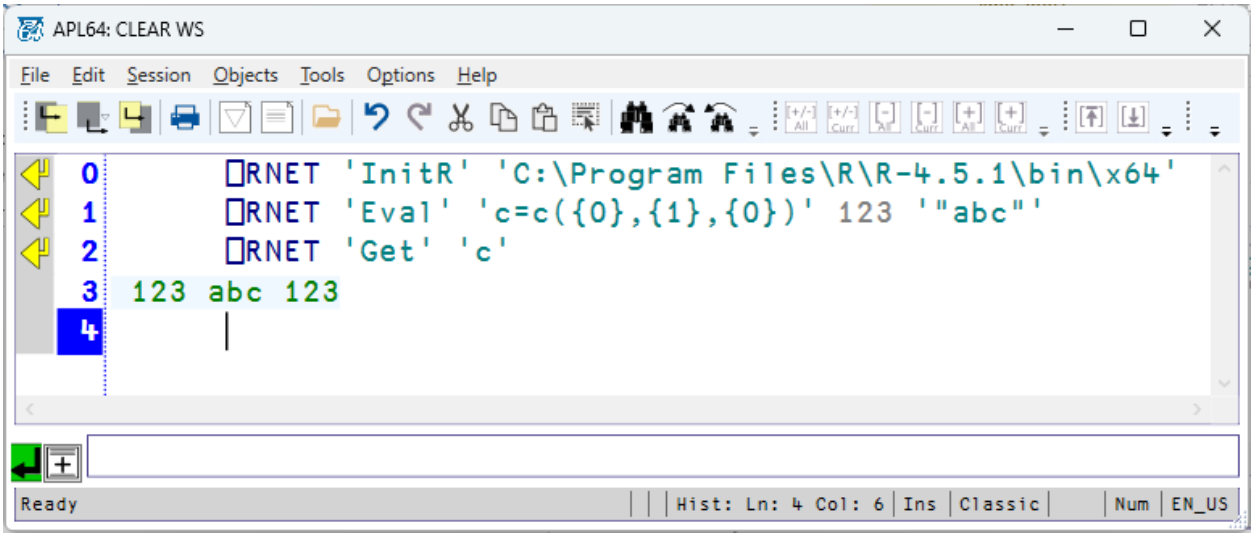


Value Substitution into the Evaluate Expression

APL64 values may be substituted into the evaluation expression. The substitution location in the Evaluation expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to evaluation of the evaluation expression.

Example

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Eval' 'c=c({0},{1},{0})' 123 '"abc"'
⎕RNET 'Get' 'c'
```

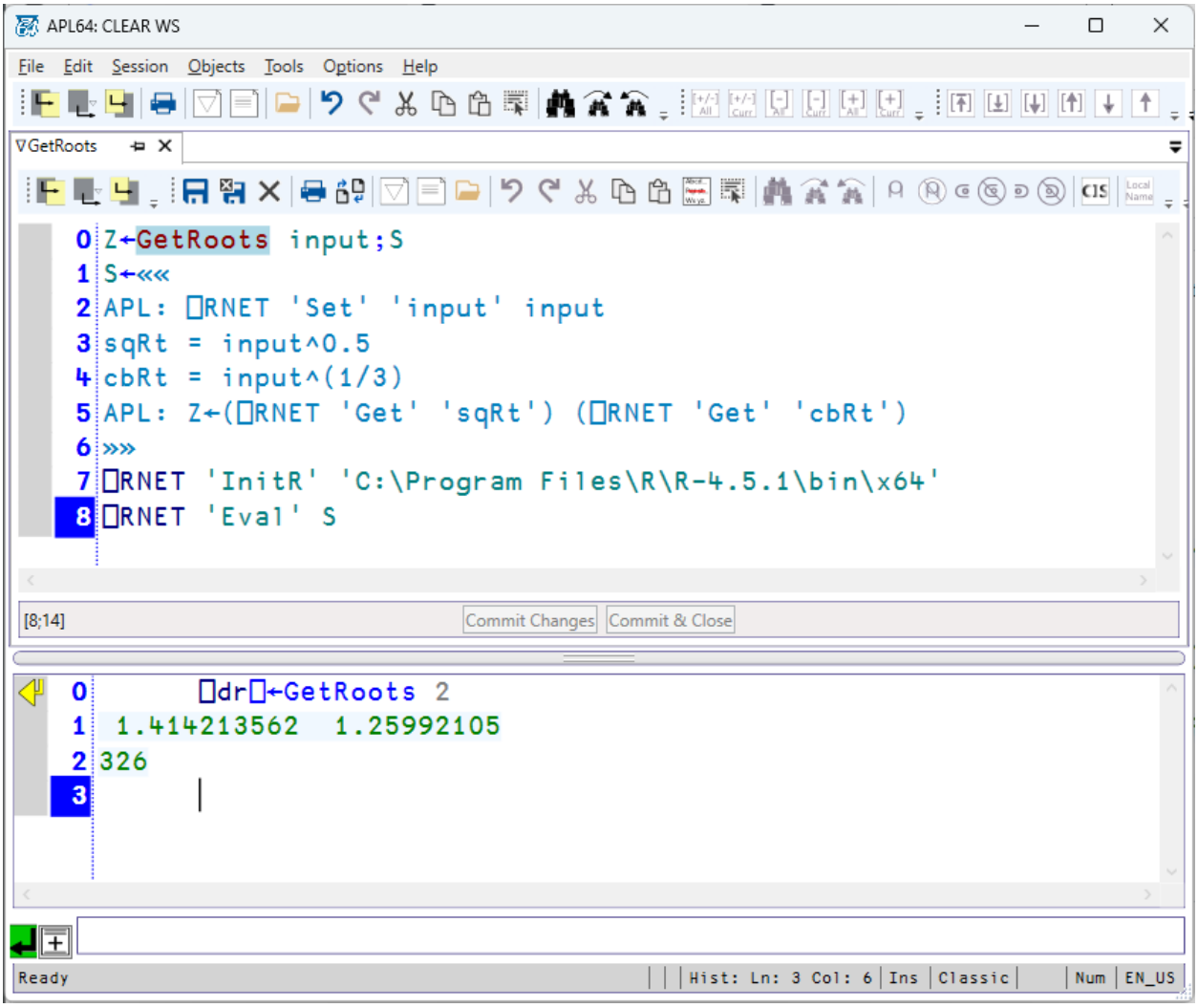


Transfer Evaluation Control between R and APL64 (APL: prefix)

Within an APL64 multi-line script, it is possible to temporarily transfer evaluation control from the R server environment to the APL64 client environment when that script is evaluated. To evaluate a script line within the APL64 client environment, prefix the APL64 executable statement on that line with 'APL:'.

Example: Using a multi-line string script

```
Z←GetRoots input;S
S←««
APL: ⎕RNET 'Set' 'input' input
sqRt = input^0.5
cbRt = input^(1/3)
APL: Z←(⎕RNET 'Get' 'sqRt') (⎕RNET 'Get' 'cbRt')
»»
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Eval' S
```



GetSymbol

Synonyms: GetSym, Get

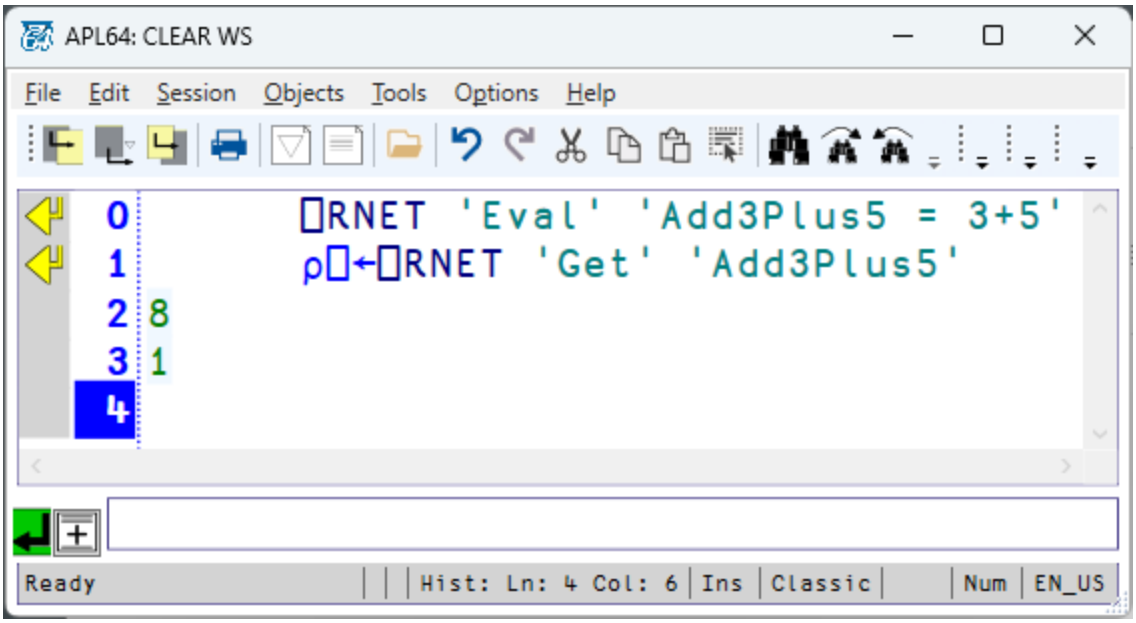
The ⎕RNET GetSymbol action fetches an APL64 representation from the R environment of the data associated with an R symbolName. symbolName can be an R-expression, which will be evaluated for its result. Some R data types have no representation in APL.

Syntax: aplValue ← ⎕RNET 'GetSymbol' symbolName [subValue1] [subValue2]…

Example:

In this example Add3Plus5 is an R-symbol name.

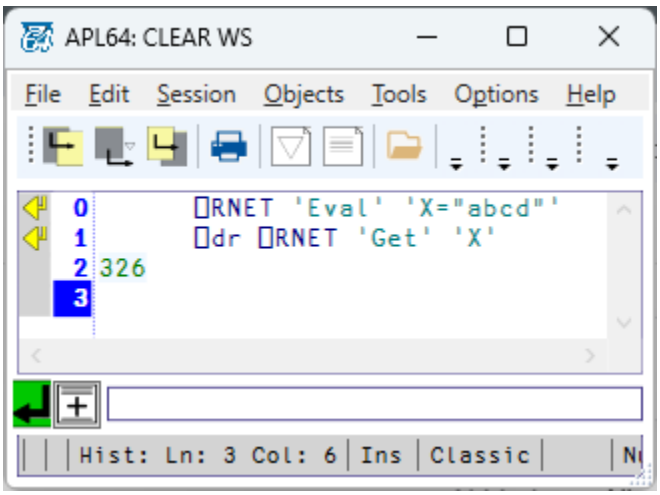
```
⎕RNET 'Eval' 'Add3Plus5 = 3+5'
ρ⎕←⎕RNET 'Get' 'Add3Plus5'
```



Example:

In this example X is an R-symbol name.

```
⎕RNET 'Eval' 'X="abcd"'
⎕dr ⎕RNET 'Get' 'X'
```

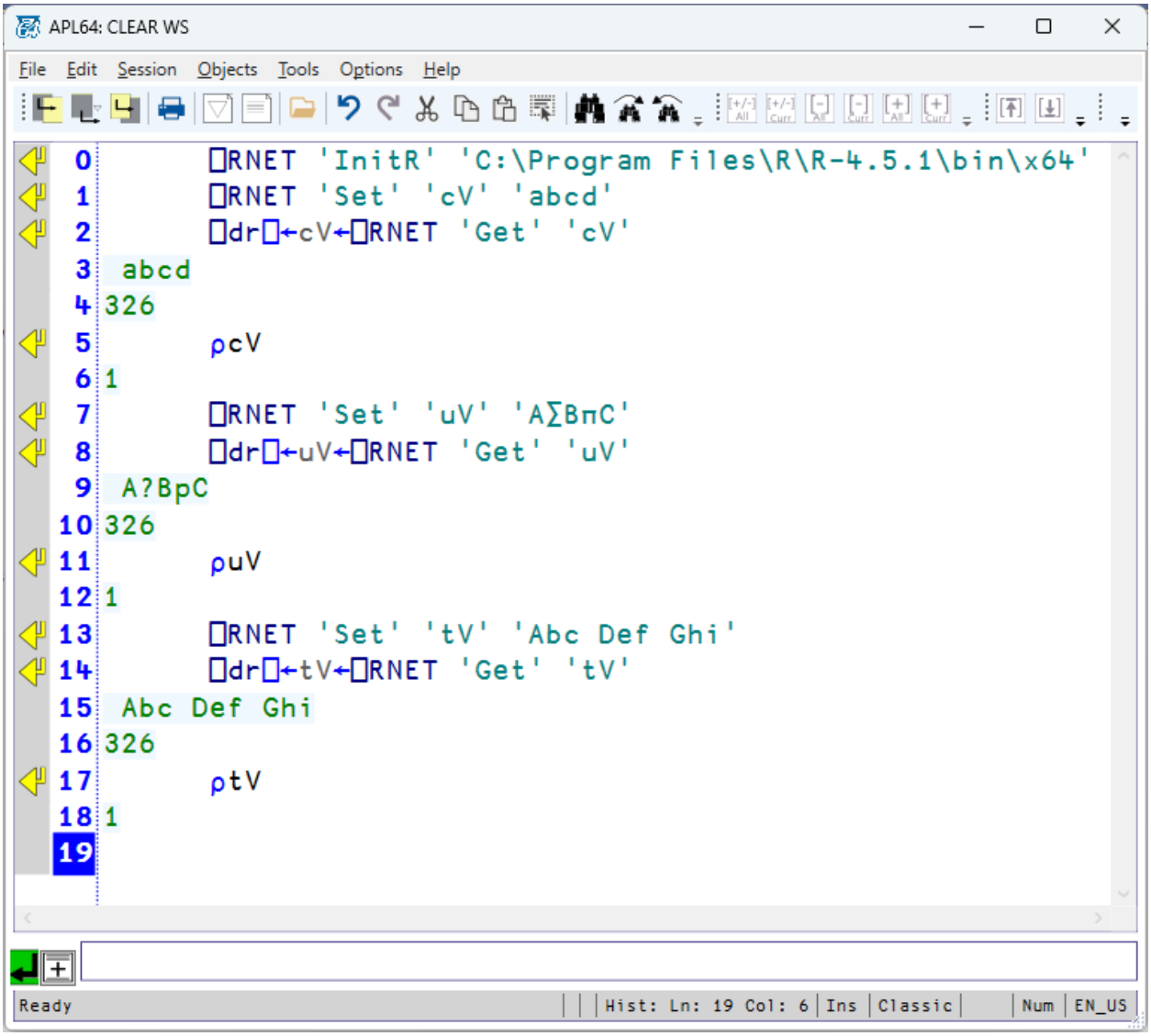


Example:

R has one text data type. When APL64 data is 'round tripped' through the R instance, the result may have a different APL64 datatype and shape. The appropriate R environment must be established for proper rendering of some Unicode glyphs.

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
```

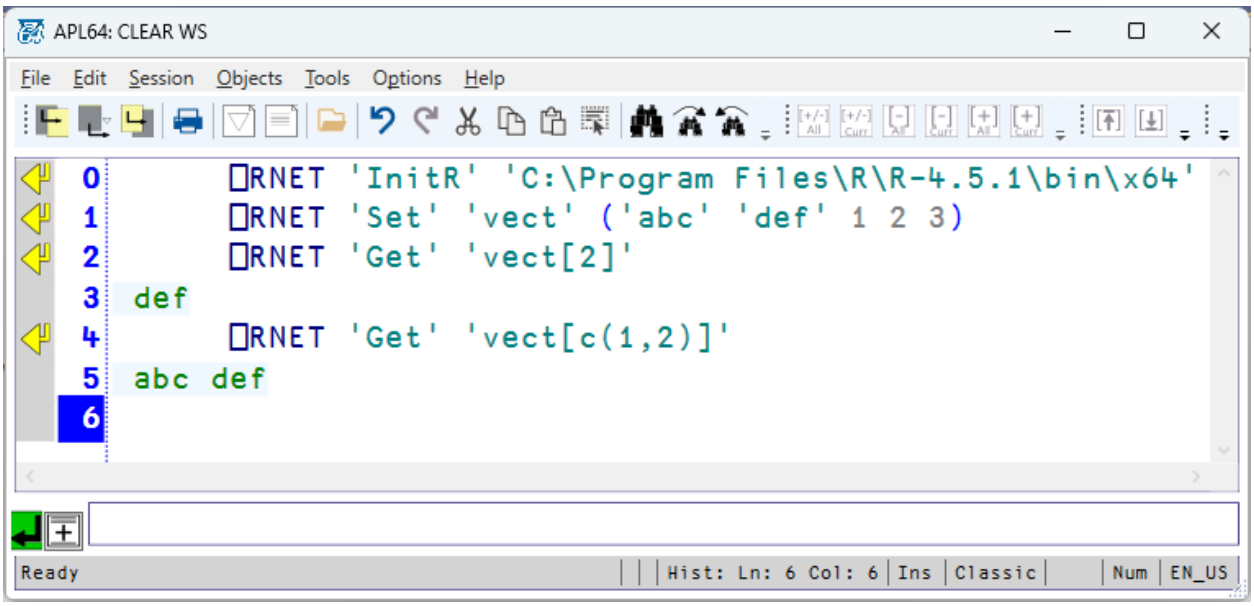
```
⎕RNET 'Set' 'cV' 'abcd'
⎕dr⎕←cV←⎕RNET 'Get' 'cV'
ρcV
⎕RNET 'Set' 'uV' 'A∑BπC'
⎕dr⎕←uV←⎕RNET 'Get' 'uV'
ρuV
⎕RNET 'Set' 'tV' 'Abc Def Ghi'
⎕dr⎕←tV←⎕RNET 'Get' 'tV'
ρtV
```



Example:

This example illustrates the ⎕RNET 'Get' action with R-expressions.

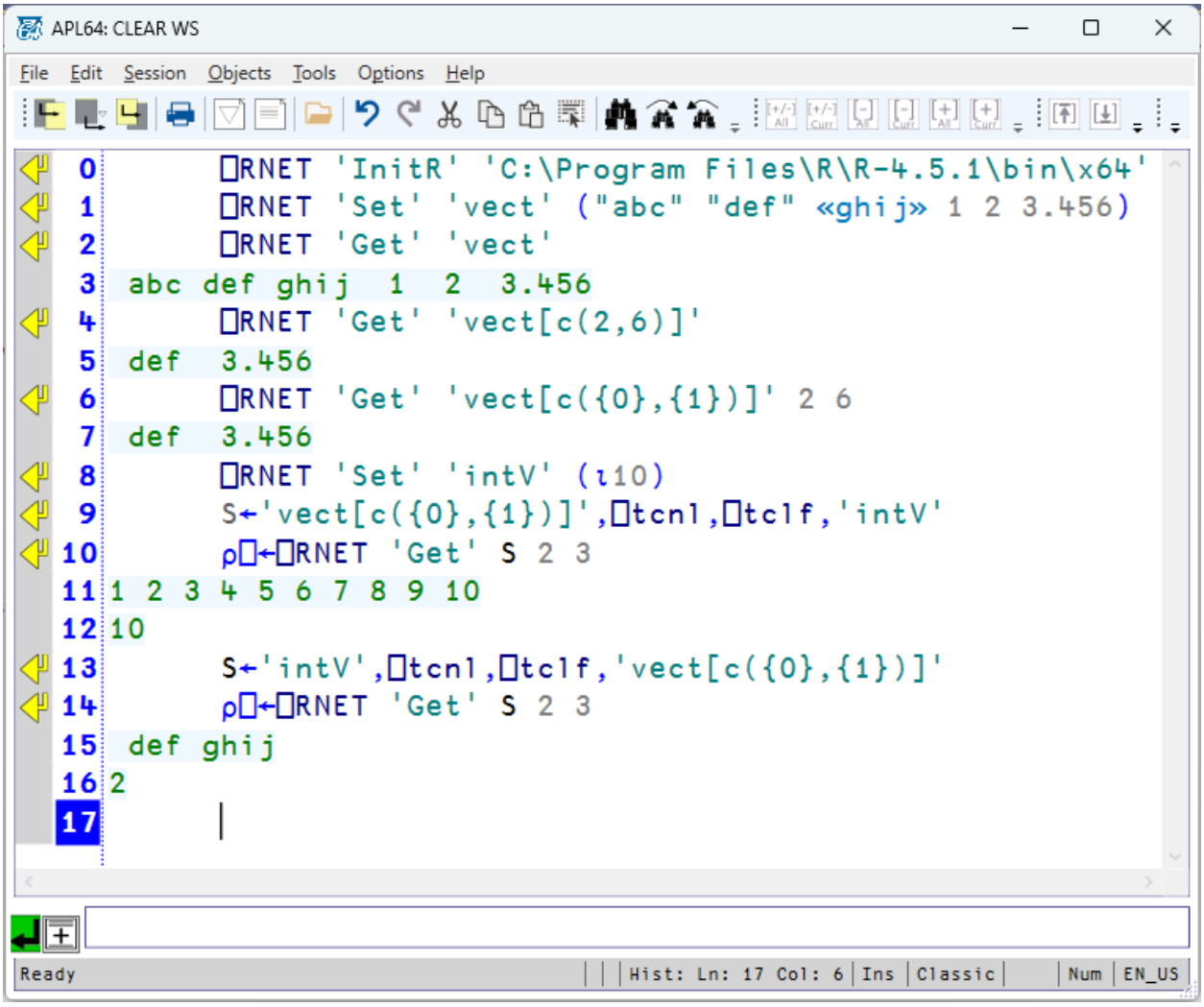
```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Set' 'vect' ('abc' 'def' 1 2 3)
⎕RNET 'Get' 'vect[2]'
⎕RNET 'Get' 'vect[c(1,2)]'
```



Value Substitution into the ⎕RNET 'Get' action R-expression

APL64 values may be substituted into the R-expression used in the ⎕RNET 'Get' action. The substitution location in the R-expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. If the R-expression emits more than one value, only the last emitted value is returned by the ⎕RNET 'Get' action.

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Set' 'vect' ("abc" "def" «ghij» 1 2 3.456)
⎕RNET 'Get' 'vect'
⎕RNET 'Get' 'vect[c(2,6)]'
⎕RNET 'Get' 'vect[c({0},{1})]' 2 6
⎕RNET 'Set' 'intV' (⍳10)
S←'vect[c({0},{1})]',⎕tcnl,⎕tclf,'intV'
⍴⎕←⎕RNET 'Get' S 2 3
S←'intV',⎕tcnl,⎕tclf,'vect[c({0},{1})]'
⍴⎕←⎕RNET 'Get' S 2 3
```



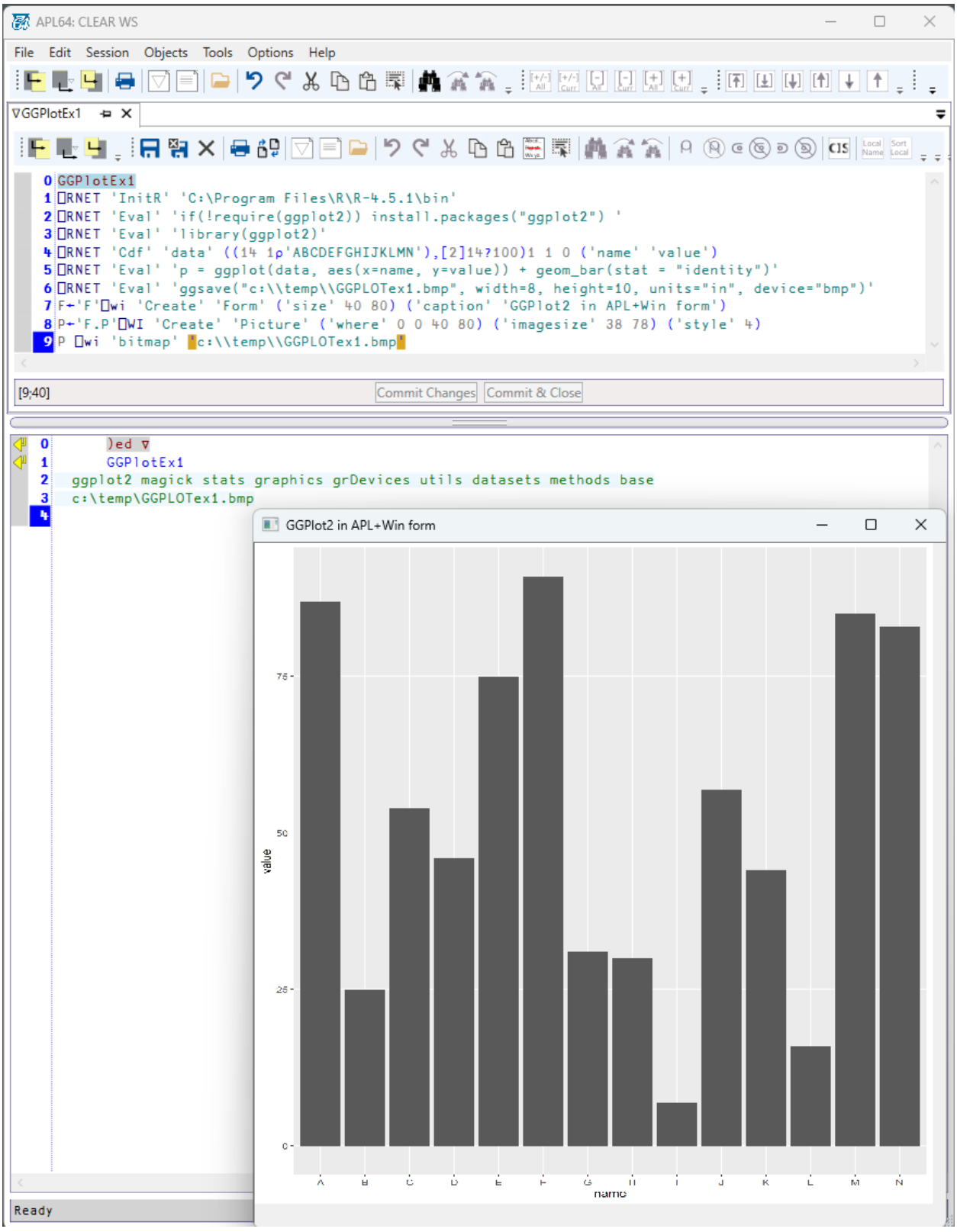
GGPLOT2

GGPlot2 is so simple to use in ⎕RNET that no ⎕RNET 'GGPlot2' action is necessary. The typical GGPlot2 workflow involves:

- Using ⎕RNET Eval or Set actions to pass APL64 data to the R instance
- If desired, use the ⎕RNET Eval or Set actions to assign a GGPlot2 result to an R variable.
- Using the ⎕RNET Eval action to run the R-function ggsave to render the GGPlot2 to a file in the desired format and dimensions and file type.
- The file-based result of GGPlot2 can be included in an APL64 ⎕wi Form using the ⎕WI Picture and bitmap features.
- For short R-scripts use ⎕RNET repetitively. For longer, multi-line R-scripts use the APL64 string continuation structure, script←««...»», within a function.

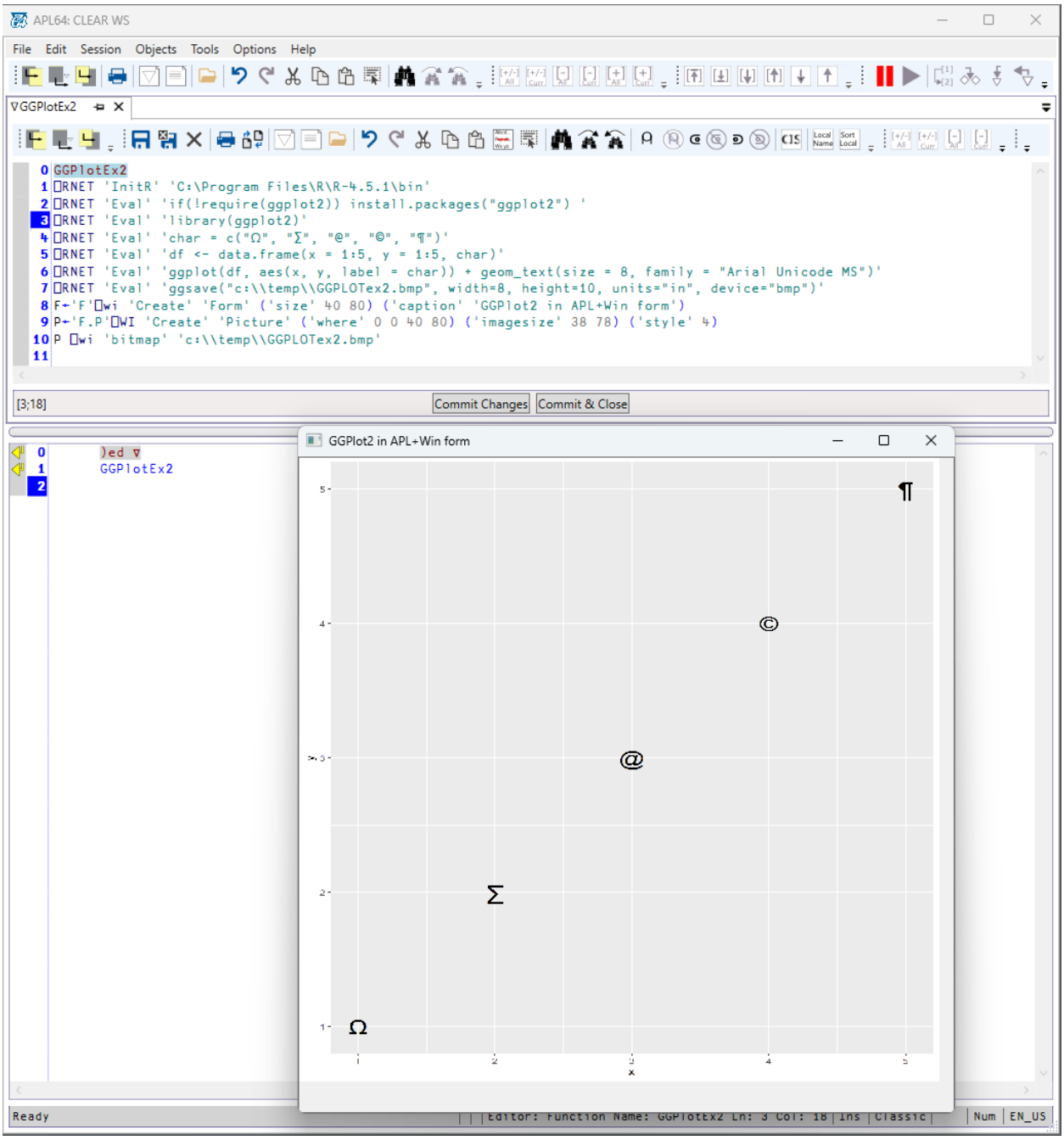
Example: Create this function and run it:

```
GGPlotEx1
□RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
□RNET 'Eval' 'if(!require(ggplot2)) install.packages("ggplot2") '
□RNET 'Eval' 'library(ggplot2)'
□RNET 'Cdf' 'data' ((14 1ρ'ABCDEFGHIJKLMN'),[2]14?100)1 1 0 ('name' 'value')
□RNET 'Eval' 'p = ggplot(data, aes(x=name, y=value)) + geom_bar(stat = "identity")'
□RNET 'Eval' 'ggsave("c:\\temp\\GGPLOTex1.bmp", width=8, height=10, units="in",
device="bmp")'
F←'F'□wi 'Create' 'Form' ('size' 40 80) ('caption' 'GGPlot2 in APLNow32 form')
P←'F.P'□WI 'Create' 'Picture' ('where' 0 0 40 80) ('imagesize' 38 78) ('style' 4)
P □wi 'bitmap' 'c:\temp\GGPLOTex1.bmp'
```



Example: Create this function and run it:

```
GGPlotEx2
□RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
□RNET 'Eval' 'if(!require(ggplot2)) install.packages("ggplot2") '
□RNET 'Eval' 'library(ggplot2)'
□RNET 'Eval' 'char = c("Ω", "∑", "@", "©", "¶")'
□RNET 'Eval' 'df <- data.frame(x = 1:5, y = 1:5, char)'
□RNET 'Eval' 'ggplot(df, aes(x, y, label = char)) + geom_text(size = 8, family = "Arial Unicode
MS")'
□RNET 'Eval' 'ggsave("c:\\temp\\GGPLOTex2.bmp", width=8, height=10, units="in",
device="bmp")'
F←'F'□wi 'Create' 'Form' ('size' 40 80) ('caption' 'GGPlot2 in APLNow32 form')
P←'F.P'□WI 'Create' 'Picture' ('where' 0 0 40 80) ('imagesize' 38 78) ('style' 4)
P □wi 'bitmap' 'c:\temp\GGPLOTex2.bmp'
```

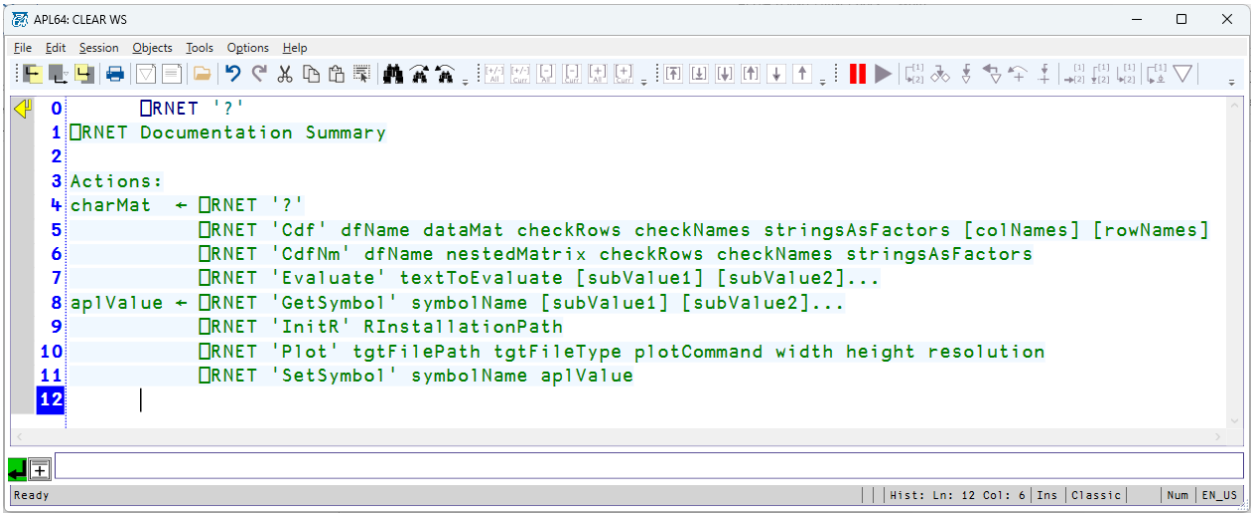


See here for many more GGPLOT2 examples.

Help

Synonym : ?

Syntax: charMat ← ⎕RNET '?'



InitR

Syntax: ⎕RNET 'InitR' installationPathOfR

This action creates an R-engine in the APL64 instance. This action must be used once in an APL64 instance before any other ⎕RNET actions are used. The installationPathOfR value will depend on how R was installed on the target workstation.

Example: For recent installations of R on Windows 11:

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
```

Example: For some installations of R environment variables are set, so that an abbreviated argument to the InitR action is acceptable:

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1'
```

Plot

The R Plot functionality is largely replaced by the superior GGPlot2 functionality. The ⎕RNET Plot action is not designed for use with GGPlot2. The magick R-package is required to use the ⎕RNET Plot action, and it will be installed the first time this action is used.

Syntax: ⎕RNET 'Plot' tgtFilePath tgtFileType plotCommand width height resolution

tgtFilePath is the full path of the file which contains the byte[] array for the plot

tgtFileType

The supported target file types for the ⎕RNET Plot action are described here. ⎕RNET does not validate the tgtFileType, because the list is dynamic. If the APL64 programmer-supplied

tgtFileType is not supported, the ⎕RNET Plot action creates a png-format file, the output file is empty, or an exception is thrown. The ⎕RNET Plot action does not modify the file extension of the tgtFilePath. Some supported tgtFileTypes:

- BMP: Bitmap, a simple uncompressed image format.
- GIF: Graphics Interchange Format, known for supporting animations and simple images.
- JPEG/JPG: Joint Photographic Experts Group, a lossy compressed format widely used for digital photography
- PDF: Portable Document Format, a document format that can contain images.
- PNG: Portable Network Graphics, a lossless format often used for web graphics.
- TIFF: Tagged Image File Format, a versatile format often used for high-quality images and printing.

plotCommand: Text of the plot commands to be evaluated. Multiple lines in the plotCommand must be separated by newline characters. The APL64 string continuation structure. plotCommand←««...»». is convenient for creating the plotCommand.

width is the desired width of the plot image in the units applicable to the tgtFileType

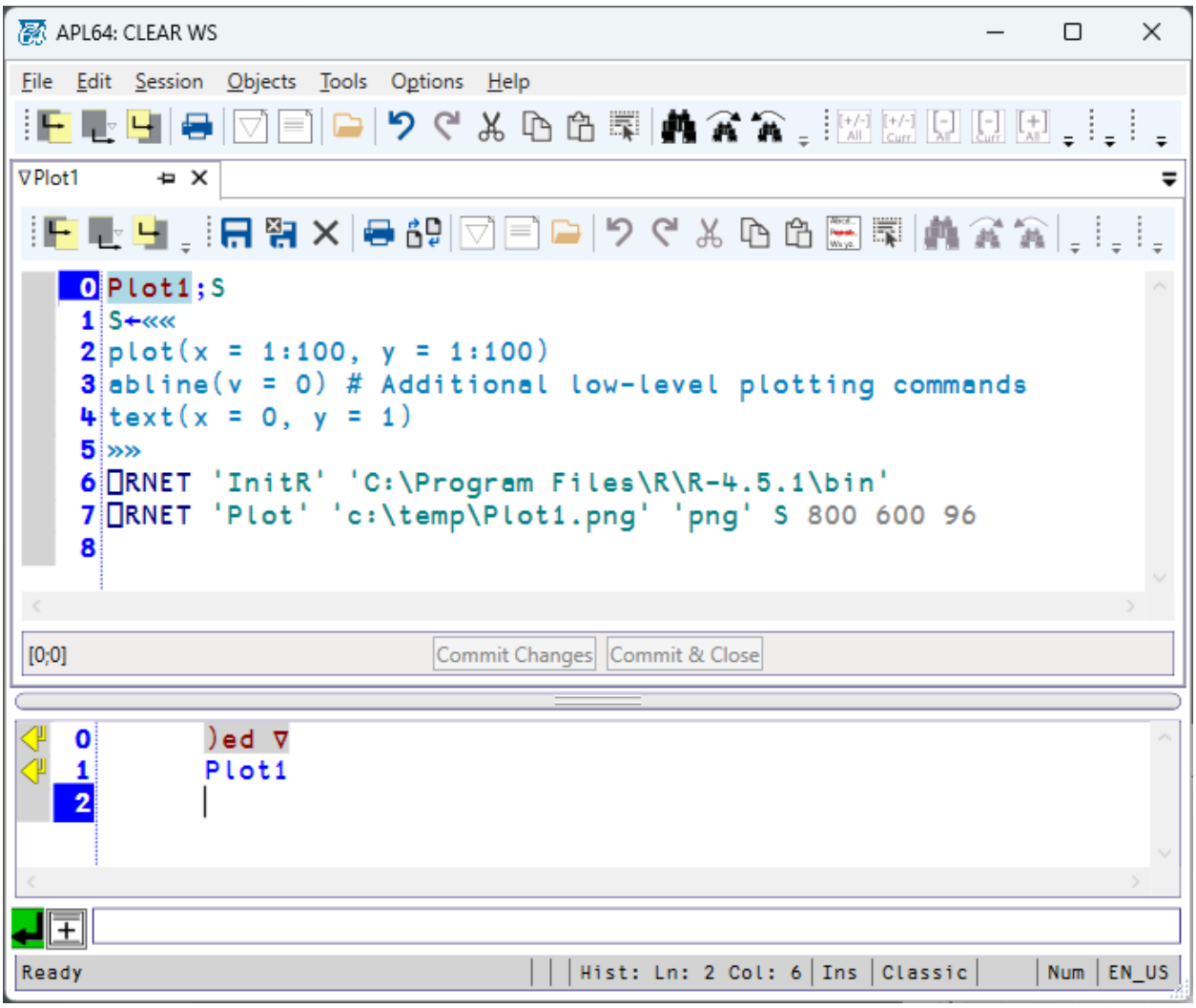
height is the desired height of the plot image in the units applicable to the tgtFileType

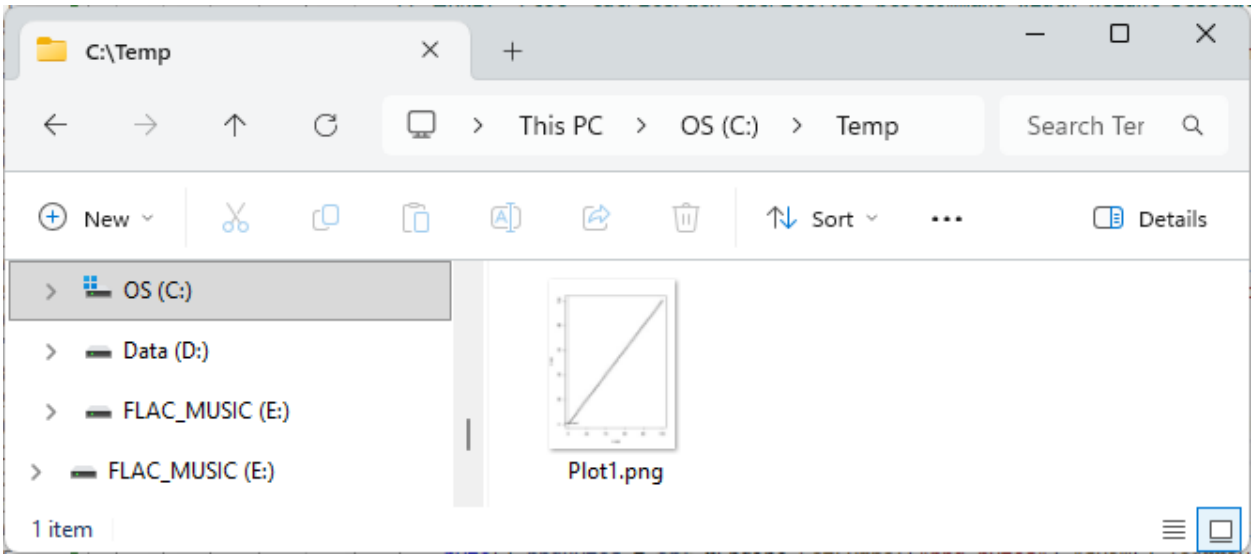
resolution is the desired resolution of the plot image in the units applicable to the tgtFileType

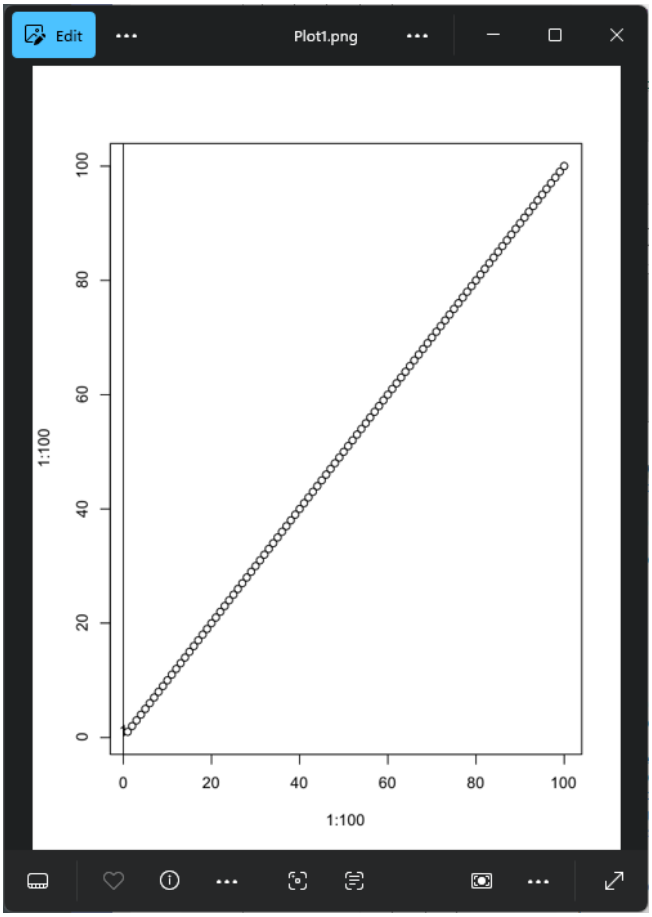
Example: Simple Plot

Create this function and run it:

```
Plot1;S
S←««
plot(x = 1:100, y = 1:100)
abline(v = 0) # Additional low-level plotting commands
text(x = 0, y = 1)
»»
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Plot' 'c:\temp\Plot1.png' 'png' S 800 600 96
```

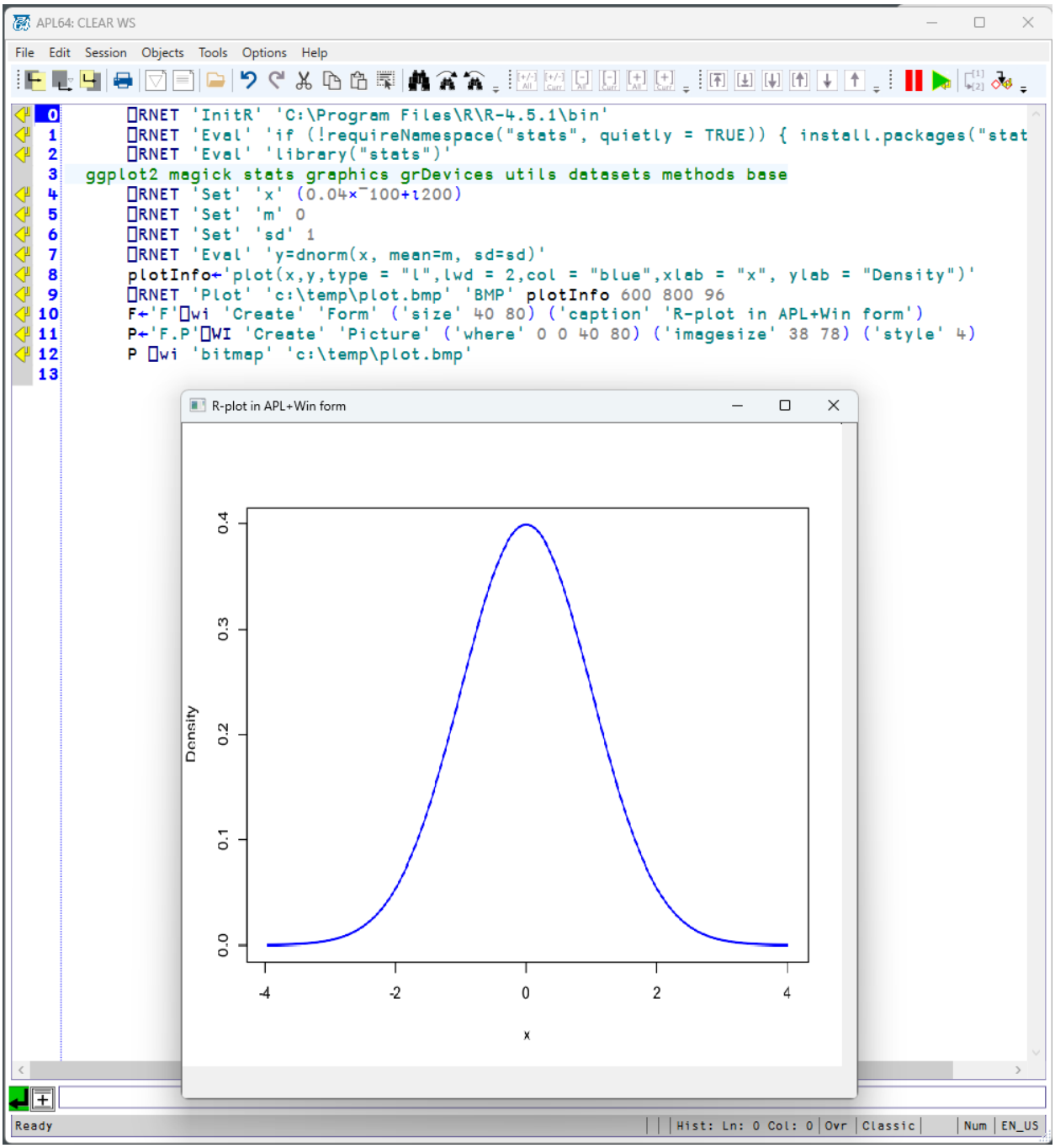






Example: Include Plot in □wi Form

```
□RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
□RNET 'Eval' 'if (!requireNamespace("stats", quietly = TRUE)) { install.packages("stats")}'
□RNET 'Eval' 'library("stats")'
□RNET 'Set' 'x' (0.04×¯100+⍳200)
□RNET 'Set' 'm' 0
□RNET 'Set' 'sd' 1
□RNET 'Eval' 'y=dnorm(x, mean=m, sd=sd)'
plotInfo←'plot(x,y,type = "l",lwd = 2,col = "blue",xlab = "x", ylab = "Density")'
□RNET 'Plot' 'c:\temp\plot.bmp' 'BMP' plotInfo 600 800 96
F←'F'□wi 'Create' 'Form' ('size' 40 80) ('caption' 'R-plot in APL+Win form')
P←'F.P'□WI 'Create' 'Picture' ('where' 0 0 40 80) ('imagesize' 38 78) ('style' 4)
P □wi 'bitmap' 'c:\temp\plot.bmp'
```



SetSymbol

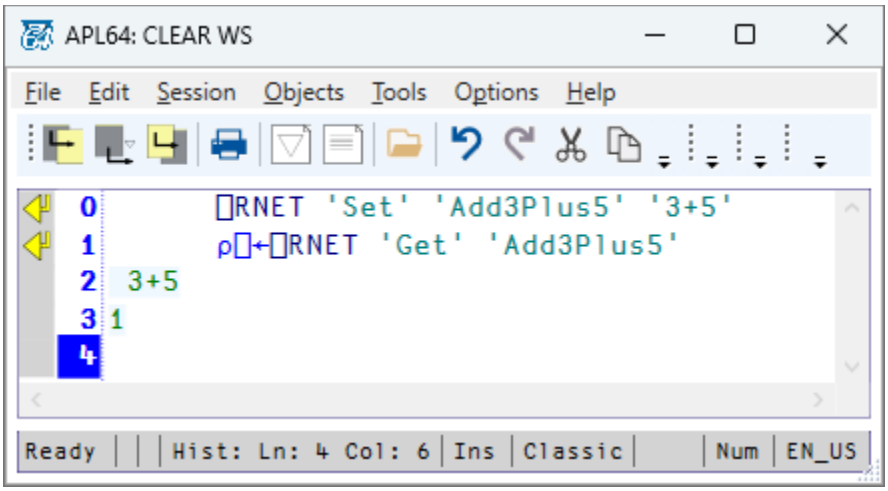
Synonyms: SetSym, Set

The ⎕RNET Set action is used to pass APL64 data to the R instance in APL64. APL64 supports Int32, double, Boolean and text data. All forms of APL64 text data are converted to the R text data type.

Syntax: rNetInstanceName ⎕RNET 'SetSymbol' symbolName aplValue

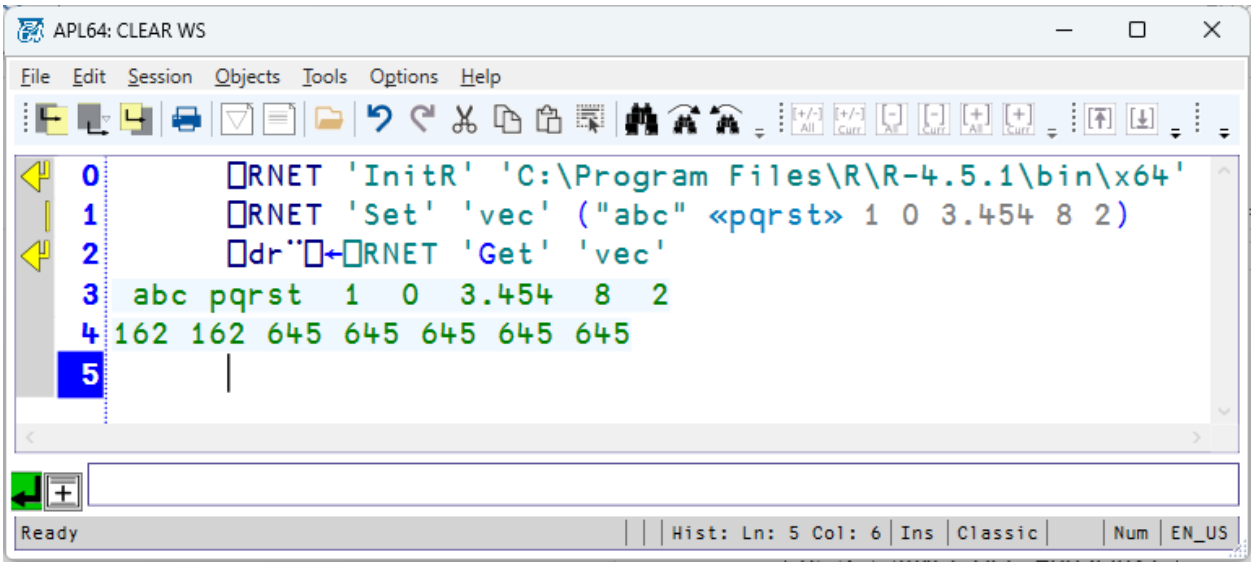
Example: APL64 scalar

```
⎕RNET 'Set' 'Add3Plus5' '3+5'
ρ⎕←⎕RNET 'Get' 'Add3Plus5'
```



Example: APL64 vector

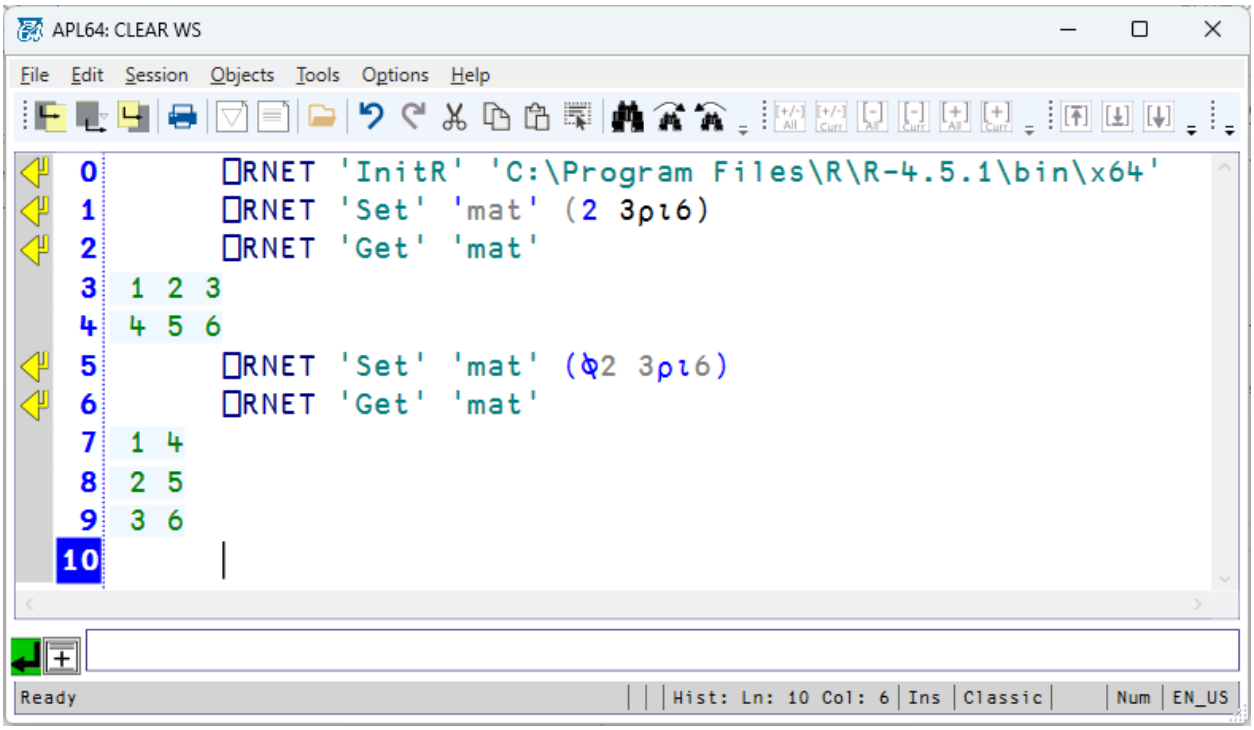
```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Set' 'vec' ("abc" «pqrst» 1 0 3.454 8 2)
⎕dr¨⎕←⎕RNET 'Get' 'vec'
```



Example: APL64 matrix

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Set' 'mat' (2 3⍴⍳6)
```

```
⎕RNET 'Get' 'mat'
⎕RNET 'Set' 'mat' (⍉2 3⍴⍳6)
⎕RNET 'Get' 'mat'
```



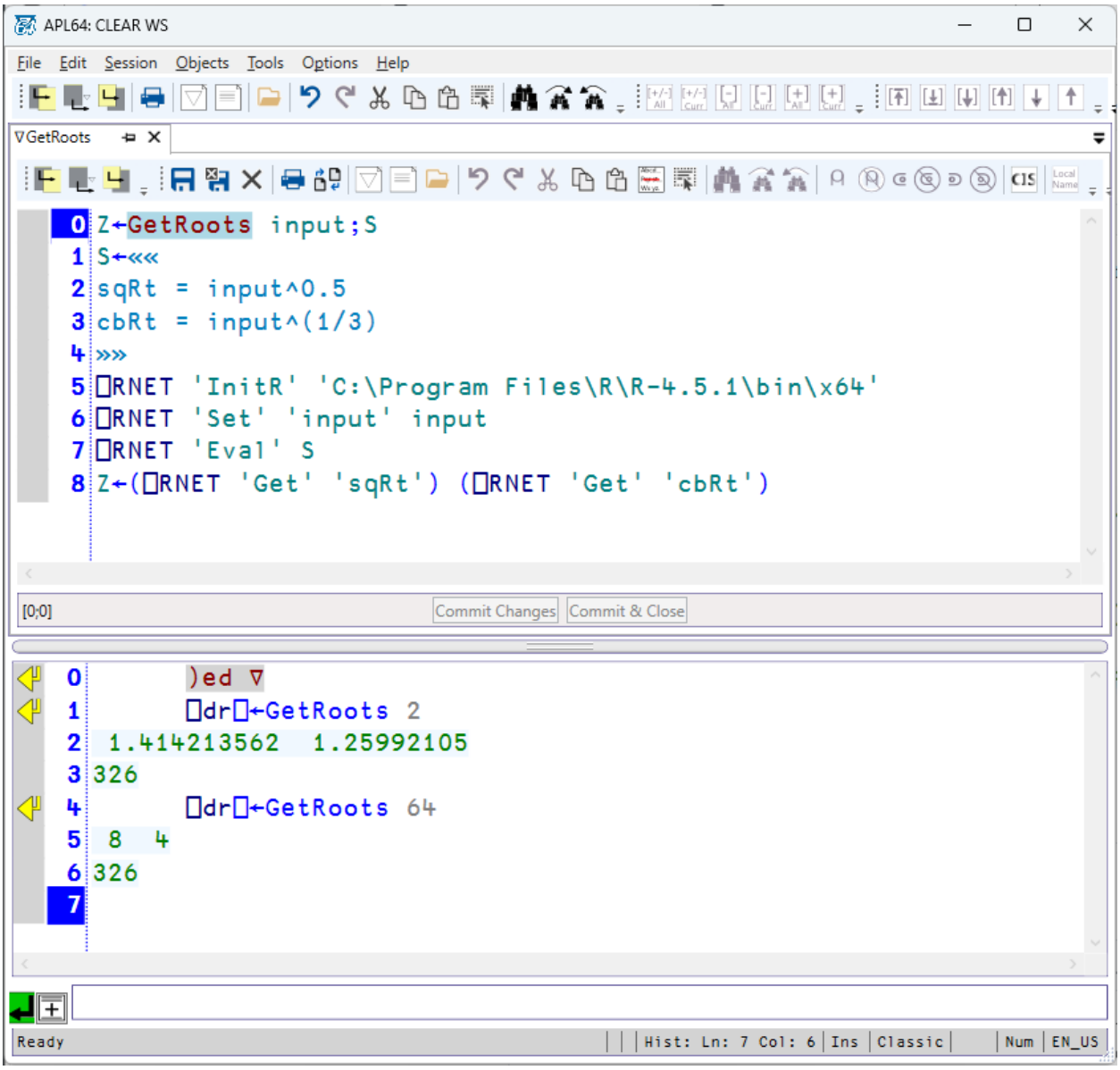
Example: APL64 homogeneous array

Creating and Executing R-Scripts

A multi-line R-script can be created in APL64 using the string continuation structure: varName←««...»». Each line of the script is evaluated individually.

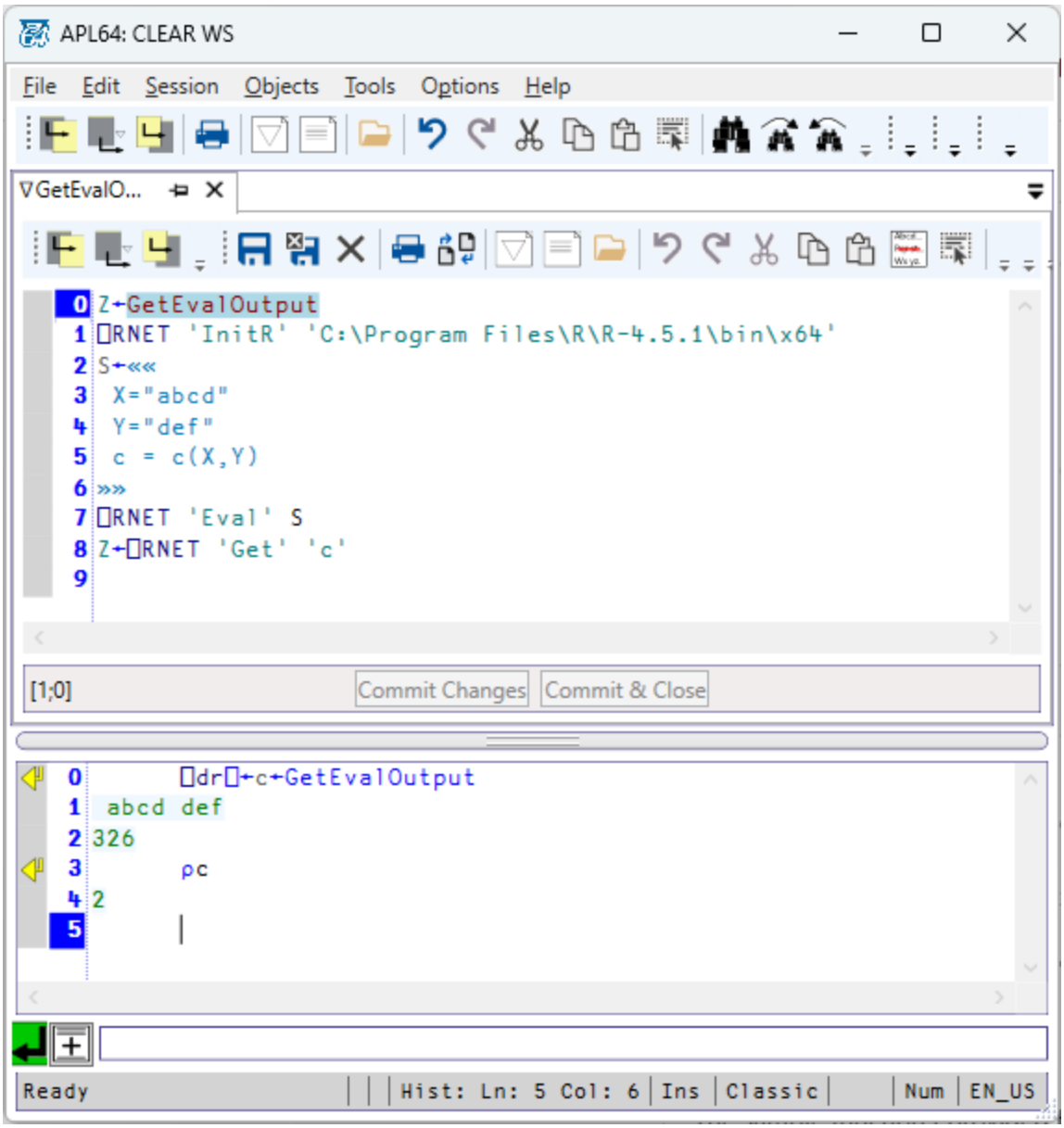
Example: Create this function, containing a multi-line R-script, and run it:

```
Z←GetRoots input;S
S←««
sqRt = input^0.5
cbRt = input^(1/3)
»»
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Set' 'input' input
⎕RNET 'Eval' S
Z←(⎕RNET 'Get' 'sqRt') (⎕RNET 'Get' 'cbRt')
```



Example: Create this function and run it:

```
Z←GetEvalOutput
 ⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
S←«««
 X="abcd"
 Y="def"
 c = c(X,Y)
»»»
 ⎕RNET 'Eval' S
Z←⎕RNET 'Get' 'c'
```

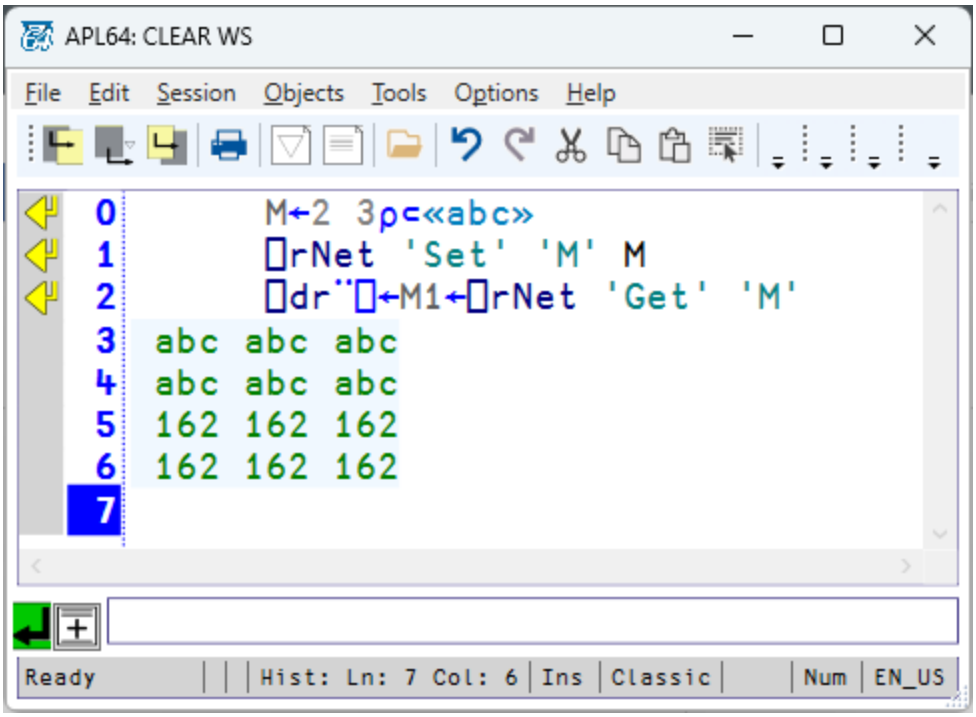


More Examples

Set and Get Symbols: ⎕rNet 'Set' and ⎕rNet 'Get'

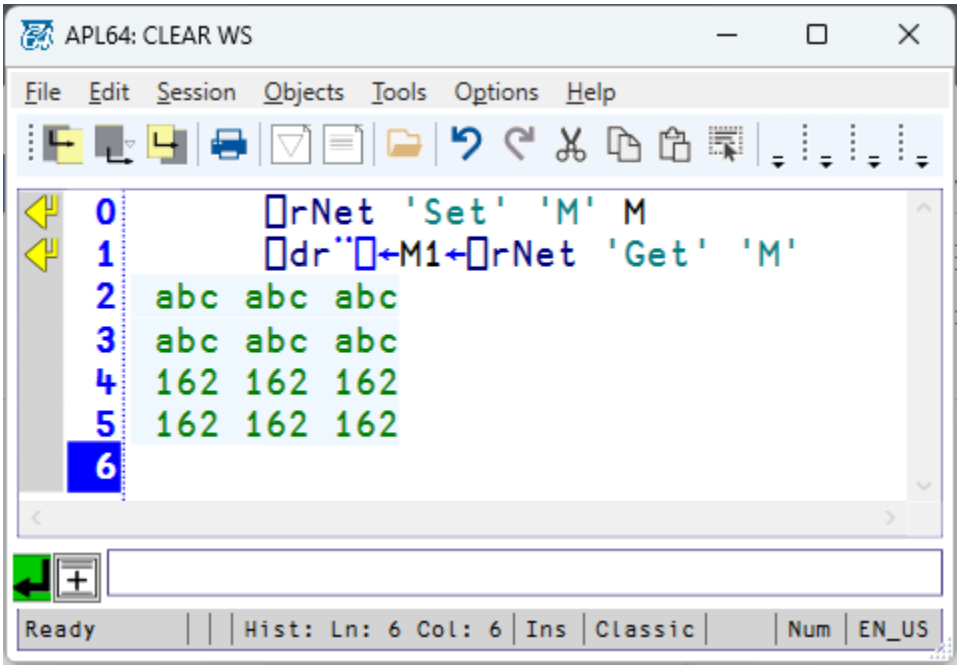
SetSymbol: Create Rank-2 Character Matrix from APL String Data

```
M←2 3⍴⊂«abc»
⎕rNet 'Set' 'M' M
⎕dr¨⎕←M1←⎕rNet 'Get' 'M'
```



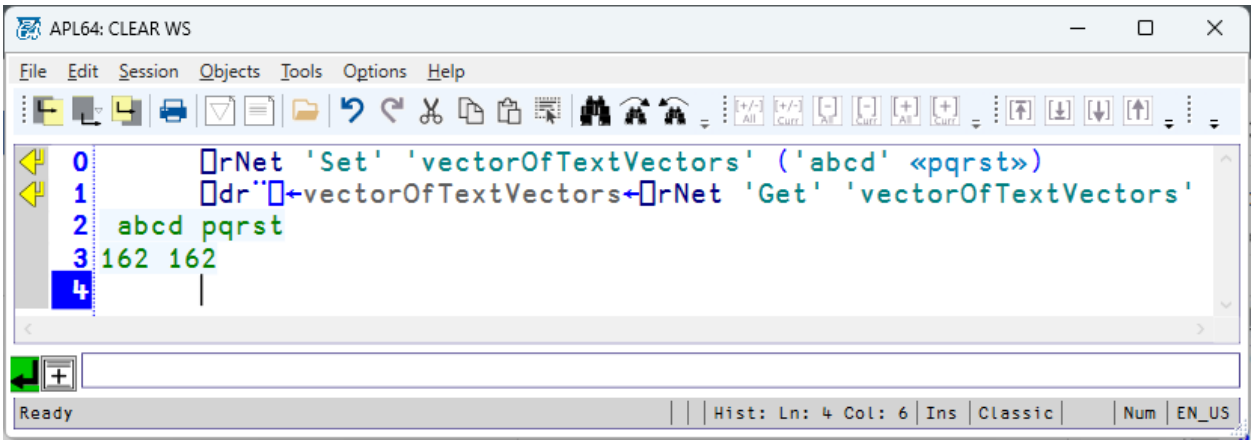
SetSymbol: Create Rank-2 Character Matrix from APL Character Data

```
⎕dr¨⎕←M←2 3⍴⊂'abc'
⎕rNet 'Set' 'M' M
⎕dr¨⎕←M1←⎕rNet 'Get' 'M'
```



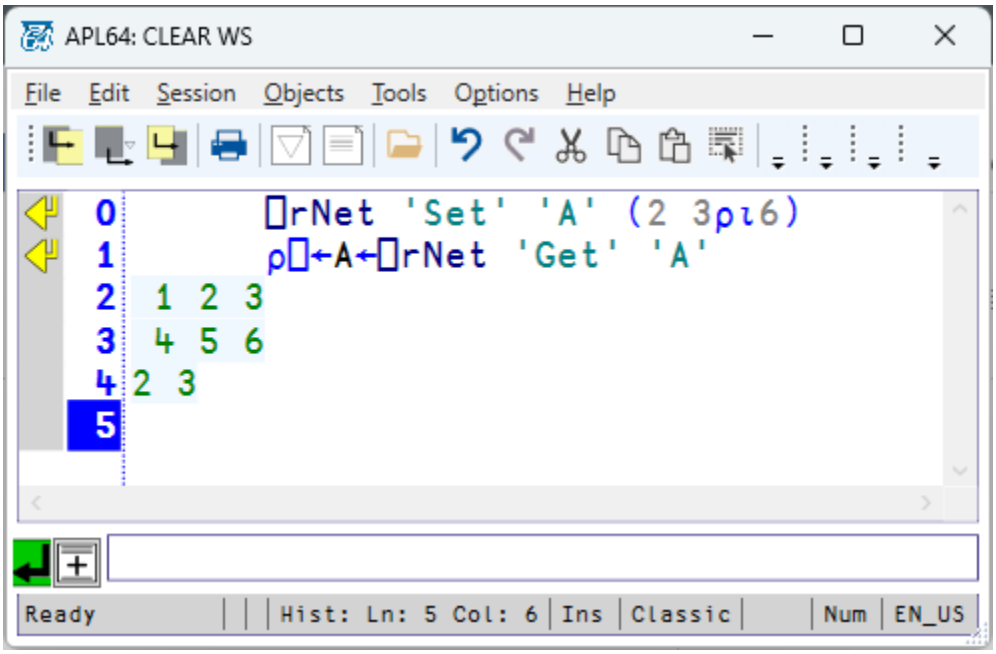
Set: Create R-Vector from APL Text

```
⎕rNet 'Set' 'vectorOfTextVectors' ('abcd' «pqrst»)
⎕dr¨⎕←vectorOfTextVectors←⎕rNet 'Get' 'vectorOfTextVectors'
```



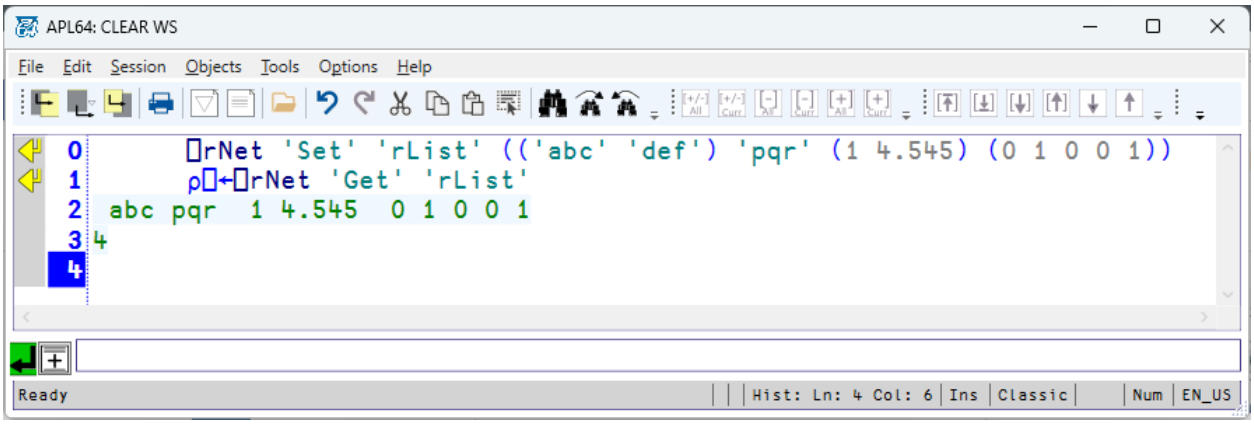
Set: Create Rank-2 R-Matrix using APL Data

```
⎕rNet 'Set' 'A' (2 3⍴⍳6)
⍴⎕←A←⎕rNet 'Get' 'A'
```



Set: Create Nested R-List Containing Mixed Data Types

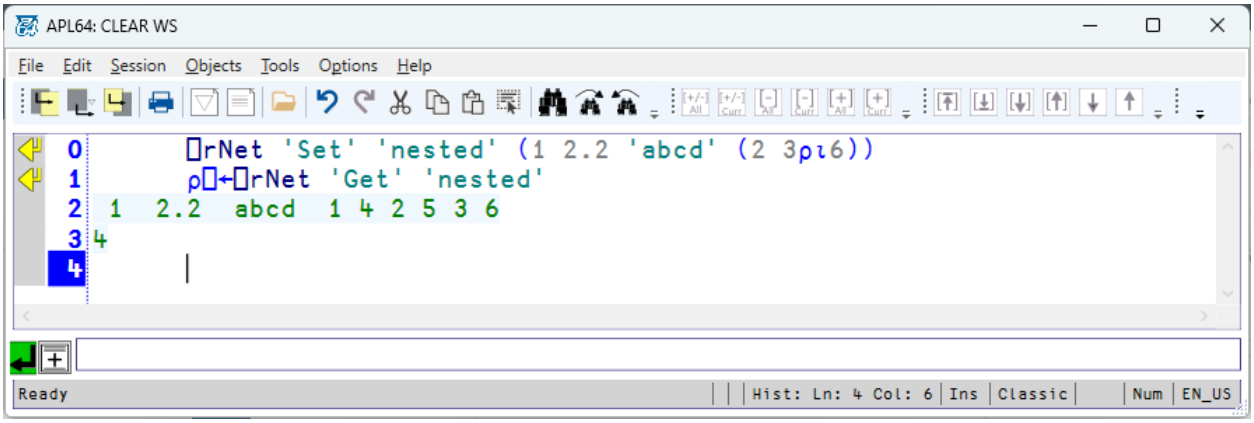
```
⎕rNet 'Set' 'rList' (('abc' 'def') 'pqr' (1 4.545) (0 1 0 0 1))
⎕rNet 'Get' 'rList'
```



Set: Create Nexted R-List Containing Mixed Data Types: APL Array of Rank > 2 is Raveled

A nested R-List can contain R-Lists, but not arrays of rank greater than 1.

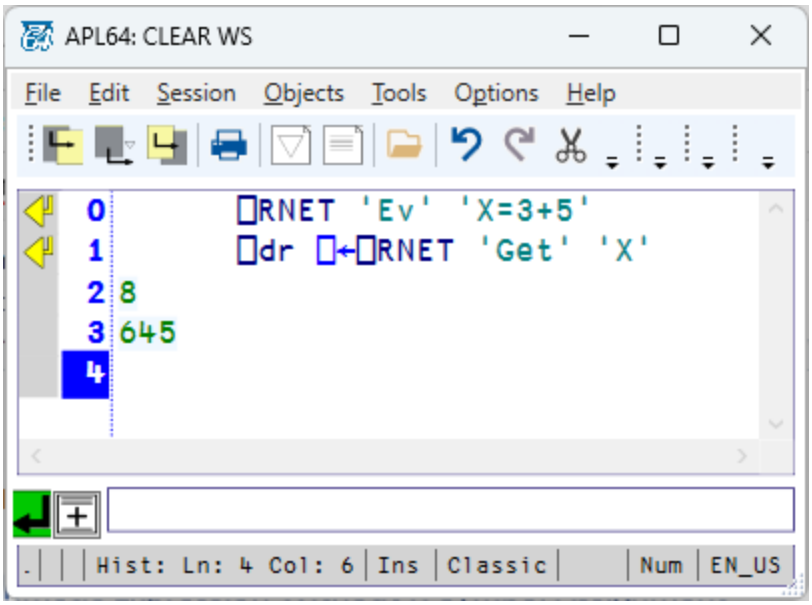
```
⎕rNet 'Set' 'nested' (1 2.2 'abcd' (2 3ρι6))
⎕rNet 'Get' 'nested'
```



Evaluate: ⎕←⎕RNET 'Eval'

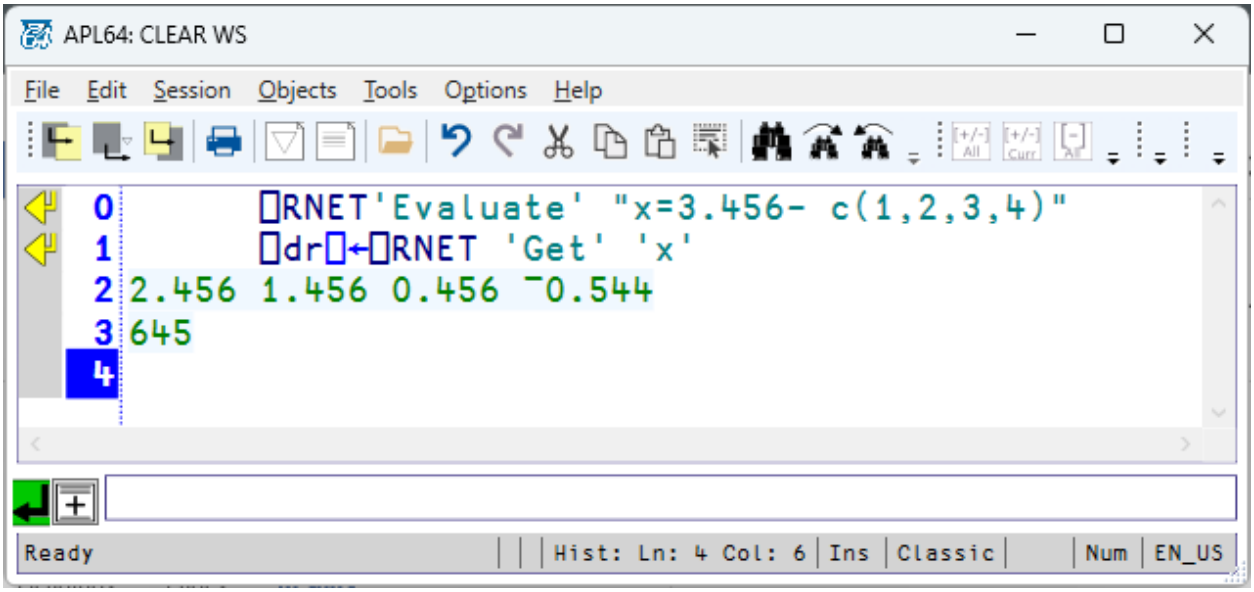
Evaluate: Arithmetic Expression Without R-Symbol Assignment

```
⎕RNET 'Ev' 'X=3+5'
⎕dr ⎕←⎕RNET 'Get' 'X'
```



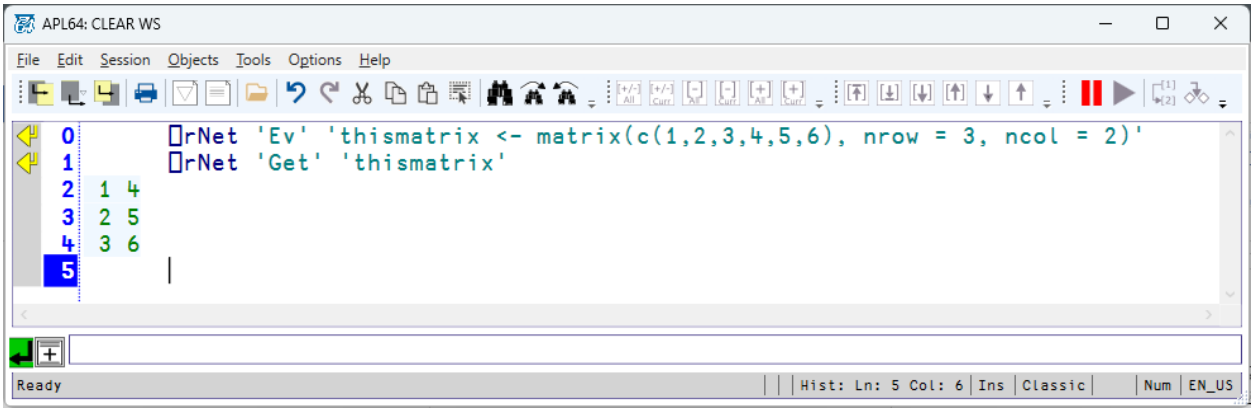
Evaluate: Create Numeric R-Vector With R-Symbol Assignment

```
⎕RNET'Evaluate' "x=3.456- c(1,2,3,4)"
⎕dr⎕←⎕RNET 'Get' 'x'
```



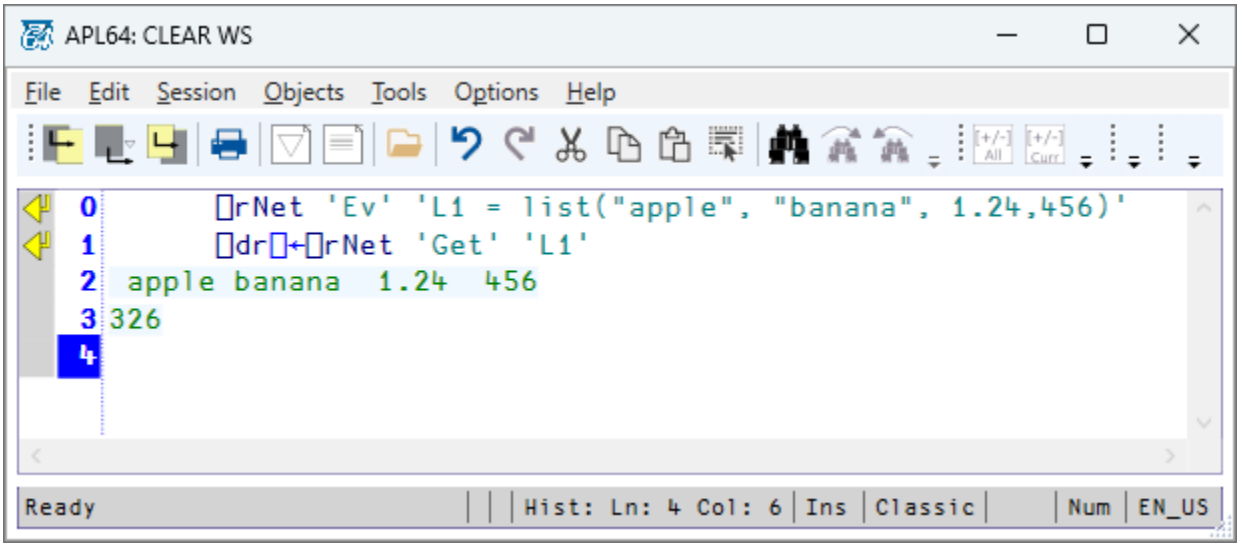
Evaluate: Create Rank-2 R-Matrix in R-Engine

```
rNet 'Ev' 'thismatrix <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)'
rNet 'Get' 'thismatrix'
```



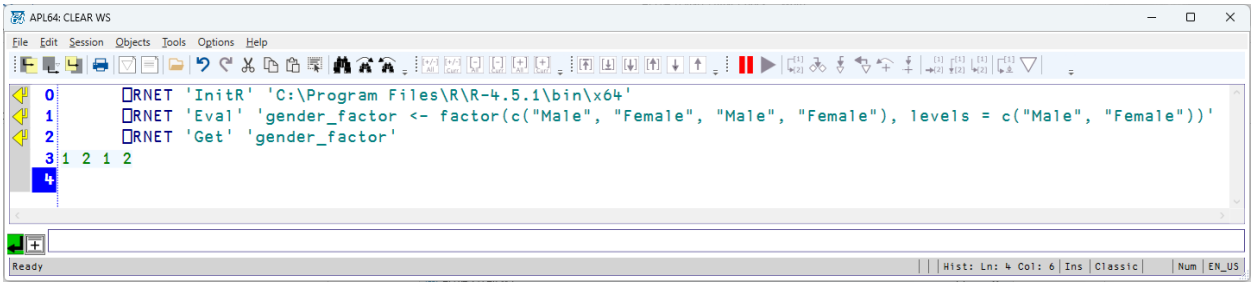
Evaluate: Create Simple R-List Containing Mixed Data Types

```
⎕rNet 'Ev' 'L1 = list("apple", "banana", 1.24,456)'
⎕dr⎕←⎕rNet 'Get' 'L1'
```



Evaluate: Create an R-factor

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'Eval' 'gender_factor <- factor(c("Male", "Female", "Male", "Female"), levels = c("Male", "Female"))'
⎕RNET 'Get' 'gender_factor'
```

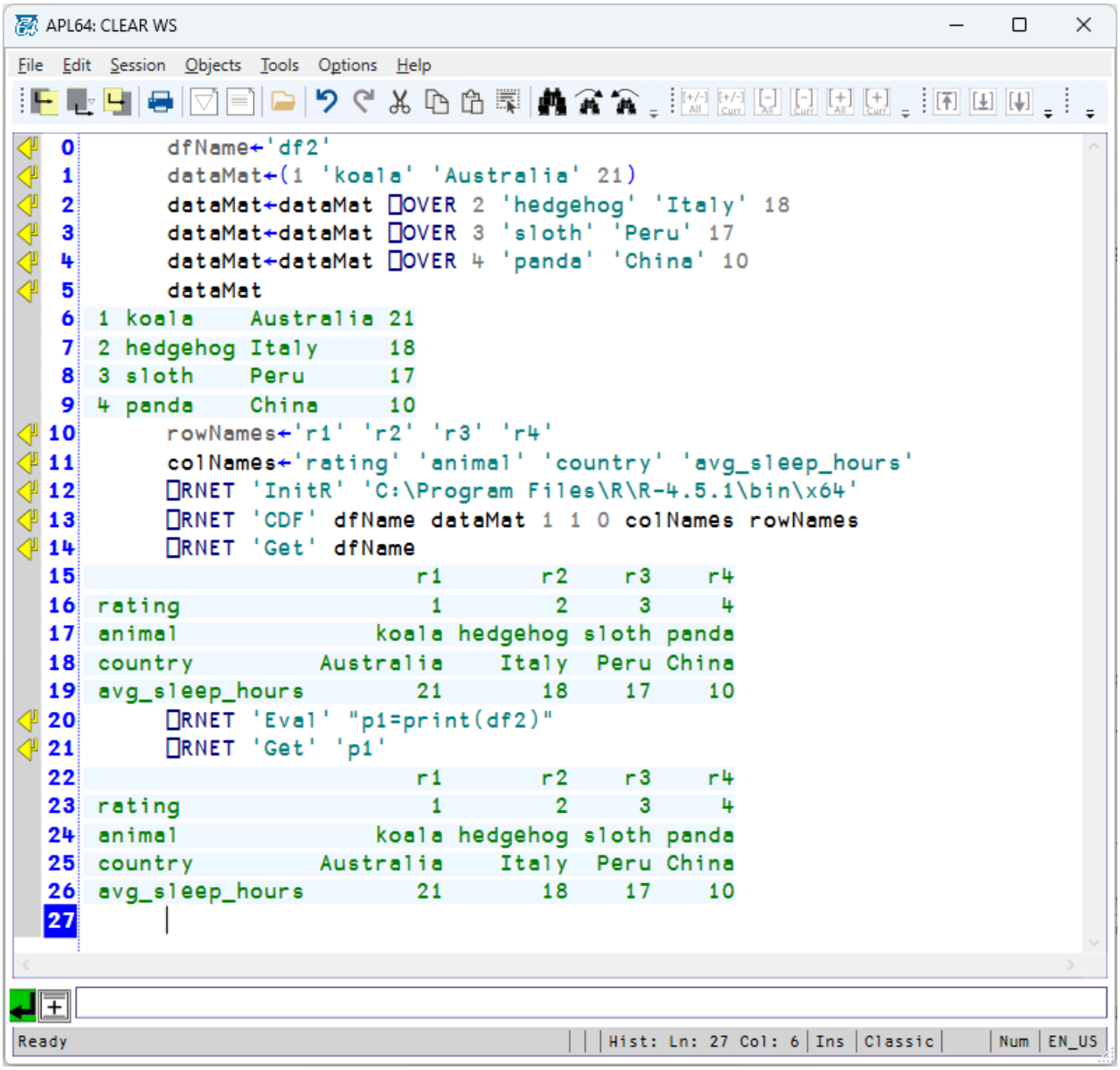


Creating DataFrames

Creating a DataFrame two ways

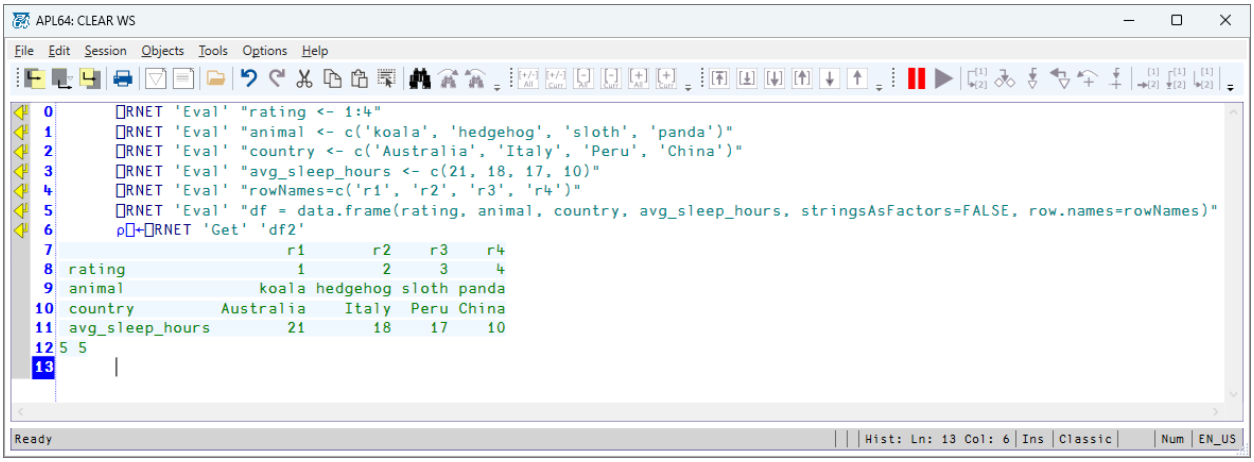
Using the ⎕RNET 'Cdf' action

```
dfName←'df2'
dataMat←(1 'koala' 'Australia' 21)
dataMat←dataMat ⎕OVER 2 'hedgehog' 'Italy' 18
dataMat←dataMat ⎕OVER 3 'sloth' 'Peru' 17
dataMat←dataMat ⎕OVER 4 'panda' 'China' 10
dataMat
rowNames←'r1' 'r2' 'r3' 'r4'
colNames←'rating' 'animal' 'country' 'avg_sleep_hours'
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕RNET 'CDF' dfName dataMat 1 1 0 colNames rowNames
⎕RNET 'Get' dfName
⎕RNET 'Eval' "p1=print(df2)"
⎕RNET 'Get' 'p1'
```



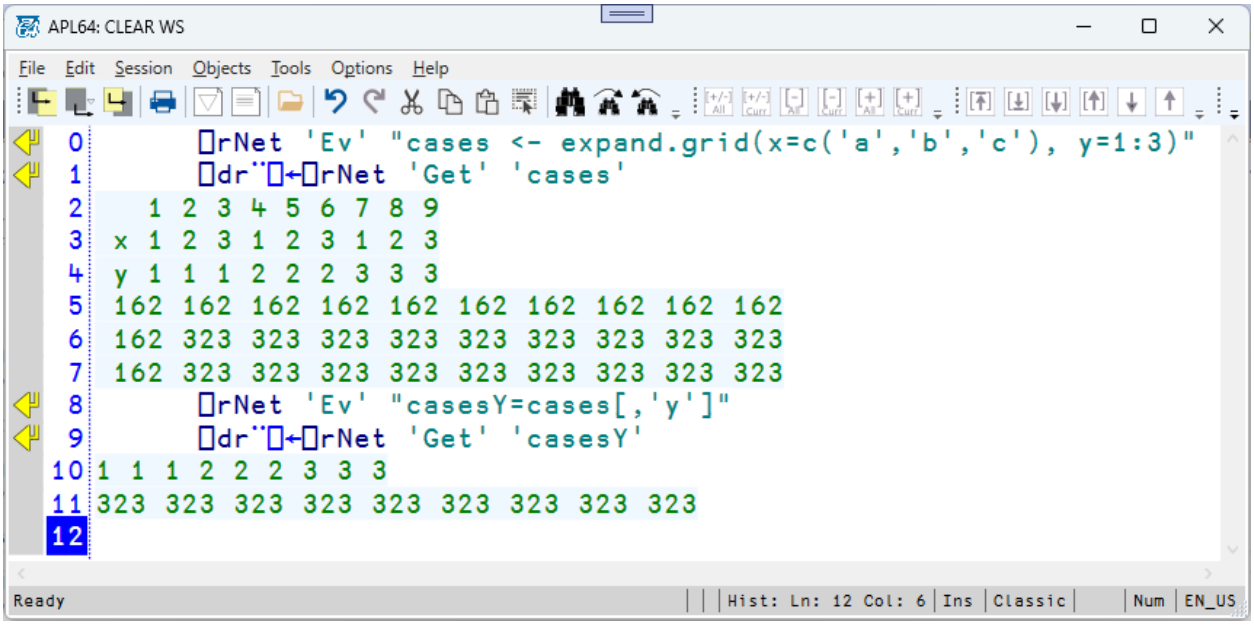
The same dataframe created without using the ⎕RNET ‘CreateDataFrame’ action:

```
⎕RNET 'Eval' "rating <- 1:4"
⎕RNET 'Eval' "animal <- c('koala', 'hedgehog', 'sloth', 'panda')"
⎕RNET 'Eval' "country <- c('Australia', 'Italy', 'Peru', 'China')"
⎕RNET 'Eval' "avg_sleep_hours <- c(21, 18, 17, 10)"
⎕RNET 'Eval' "rowNames=c('r1', 'r2', 'r3', 'r4')"
⎕RNET 'Eval' "df = data.frame(rating, animal, country, avg_sleep_hours, stringsAsFactors=FALSE, row.names=rowNames)"
ρ⎕←⎕RNET 'Get' 'df2'
```



Create R-DataFrame and obtain R-String with R Statements

```
⎕rNet 'Ev' "cases <- expand.grid(x=c('a','b','c'), y=1:3)"
⎕dr¨⎕←⎕rNet 'Get' 'cases'
⎕rNet 'Ev' "casesY=cases[,'y']"
⎕dr¨⎕←⎕rNet 'Get' 'casesY'
```

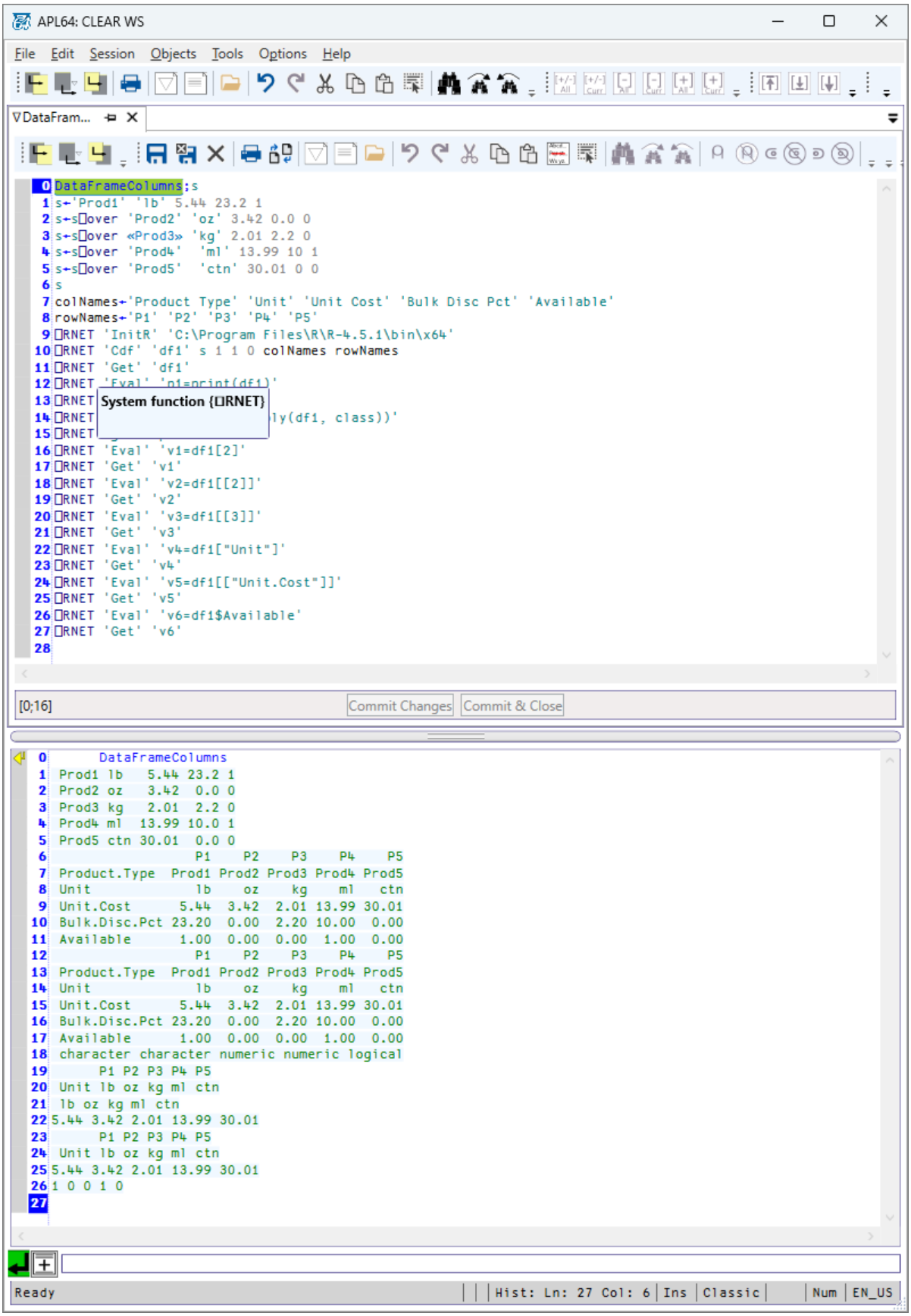


Accessing Dataframe Columns

Create this function and run it:

```
DataFrameColumns;s
s←'Prod1' 'lb' 5.44 23.2 1
s←s⍪'Prod2' 'oz' 3.42 0.0 0
s←s⍪«Prod3» 'kg' 2.01 2.2 0
s←s⍪'Prod4'  'ml' 13.99 10 1
s←s⍪'Prod5'  'ctn' 30.01 0 0
s
colNames←'Product Type' 'Unit' 'Unit Cost' 'Bulk Disc Pct' 'Available'
rowNames←'P1' 'P2' 'P3' 'P4' 'P5'
⍝RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⍝RNET 'Cdf' 'df1' s 1 1 0 colNames rowNames
⍝RNET 'Get' 'df1'
⍝RNET 'Eval' 'p1=print(df1)'
⍝RNET 'Get' 'p1'
⍝RNET 'eval' 'p2=print(sapply(df1, class))'
⍝RNET 'get' 'p2'
⍝RNET 'Eval' 'v1=df1[2]'
⍝RNET 'Get' 'v1'
⍝RNET 'Eval' 'v2=df1[[2]]'
⍝RNET 'Get' 'v2'
⍝RNET 'Eval' 'v3=df1[[3]]'
⍝RNET 'Get' 'v3'
⍝RNET 'Eval' 'v4=df1["Unit"]'
⍝RNET 'Get' 'v4'
```

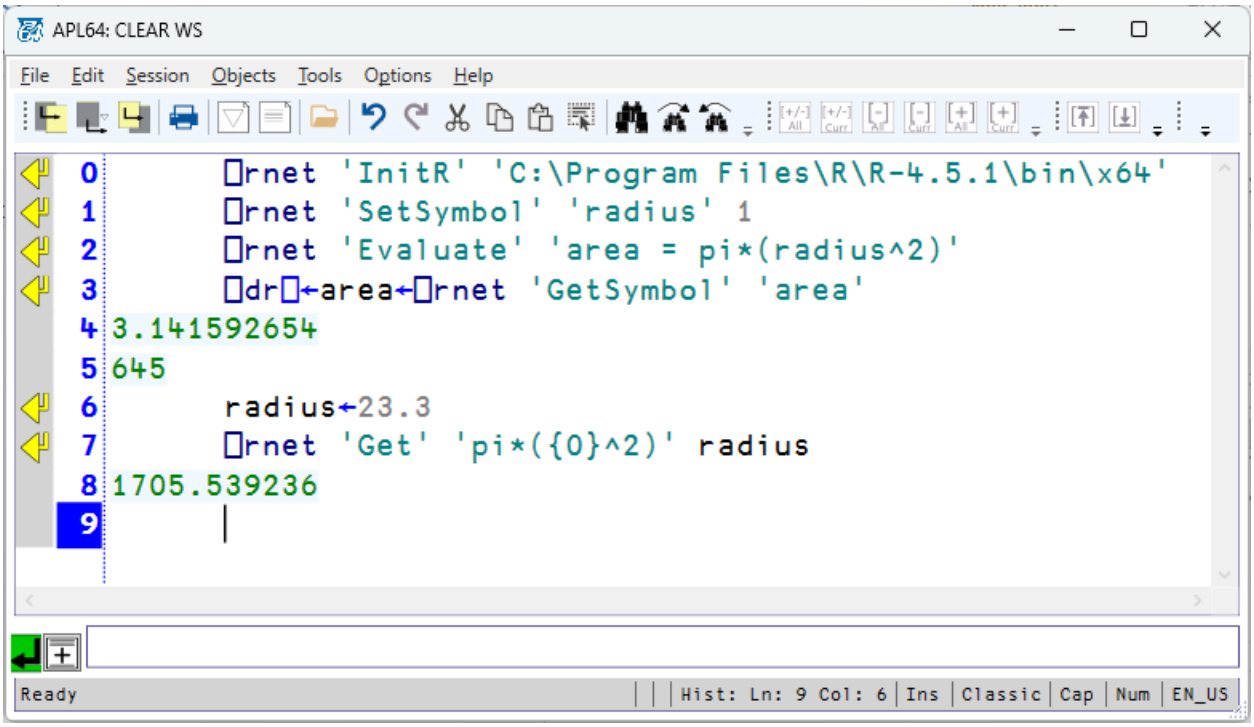
```
RNET 'Eval' 'v5=df1[["Unit.Cost"]]'
RNET 'Get' 'v5'
RNET 'Eval' 'v6=df1$Available'
RNET 'Get' 'v6'
```



Using R-Engine for *ad hoc* Calculations

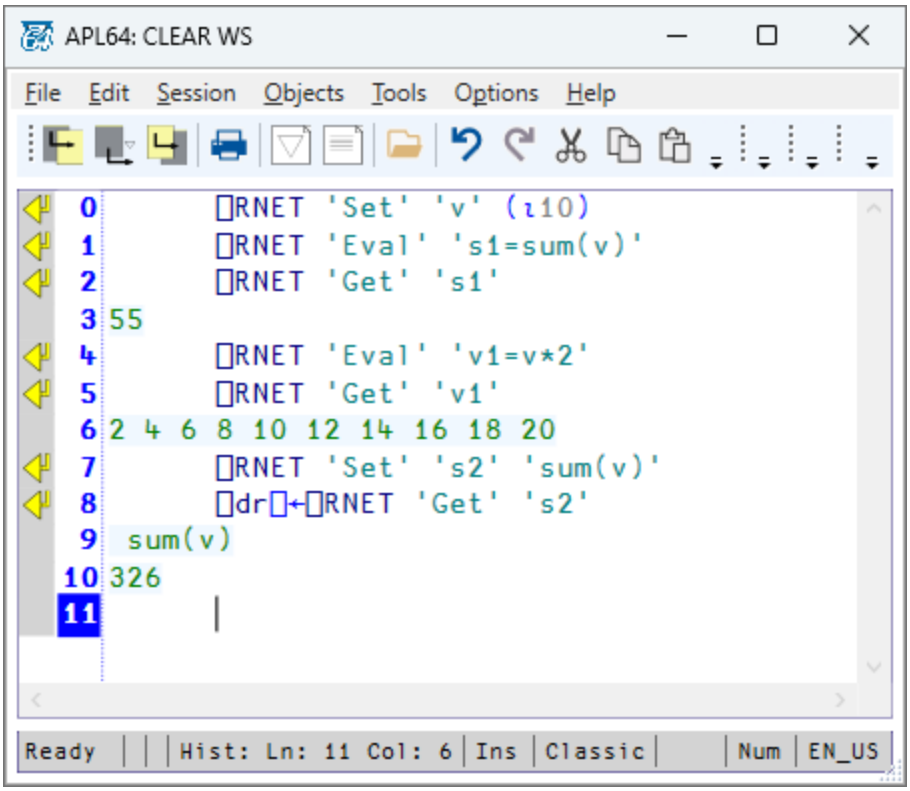
In this example, the input value, radius, is provided to the R engine and used to calculate the area of a circle using the R engine Evaluate action. The ⎕rnet 'Get' action with value substitution can simplify the operation:

```
⎕rnet 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
⎕rnet 'SetSymbol' 'radius' 1
⎕rnet 'Evaluate' 'area = pi*(radius^2)'
⎕dr⎕←area←⎕rnet 'GetSymbol' 'area'
radius←23.3
⎕rnet 'Get' 'pi*({0}^2)' radius
```



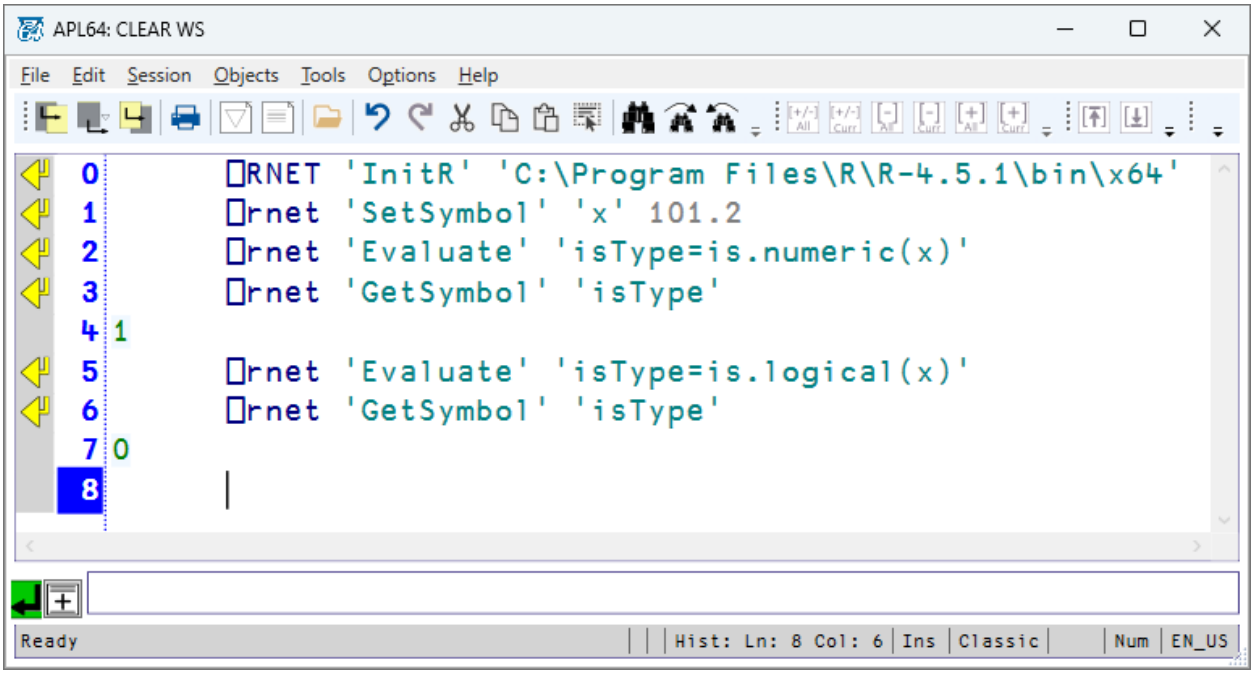
In this example the R sum() and multiply actions are used:

```
⎕RNET 'Set' 'v' (⍳10)
⎕RNET 'Eval' 's1=sum(v)'
⎕RNET 'Get' 's1'
⎕RNET 'Eval' 'v1=v*2'
⎕RNET 'Get' 'v1'
⎕RNET 'Set' 's2' 'sum(v)'
⎕dr⎕←⎕RNET 'Get' 's2'
```



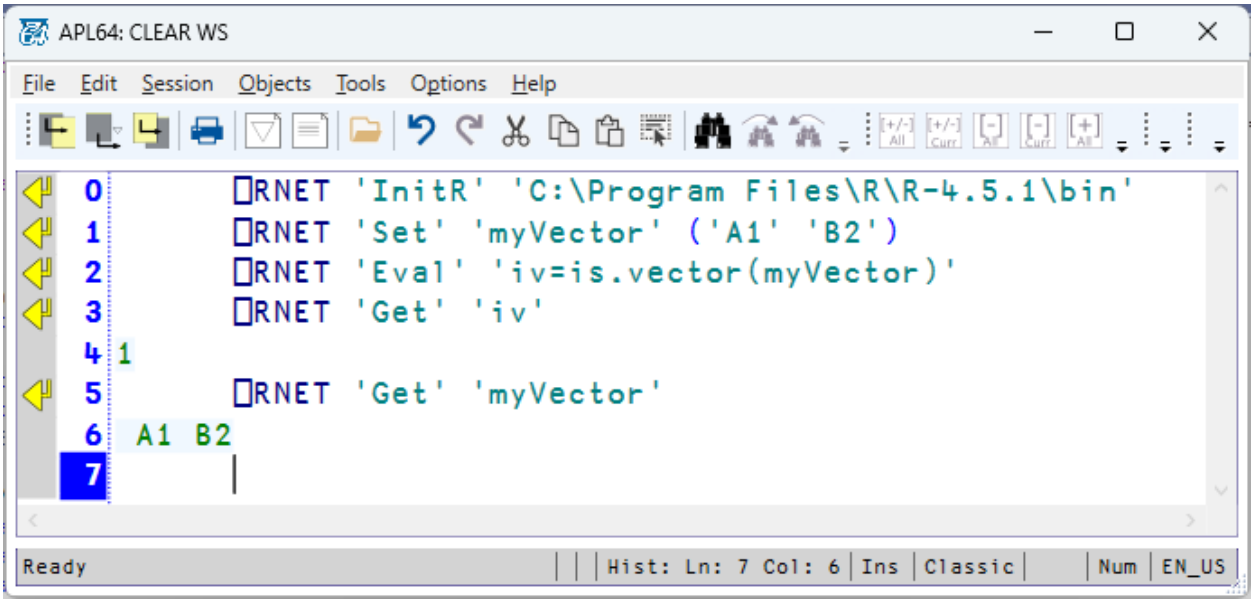
Check the Data Type of an R symbol

```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
rnet 'SetSymbol' 'x' 101.2
rnet 'Evaluate' 'isType=is.numeric(x)'
rnet 'GetSymbol' 'isType'
rnet 'Evaluate' 'isType=is.logical(x)'
rnet 'GetSymbol' 'isType'
```



Check if an R-object is a vector

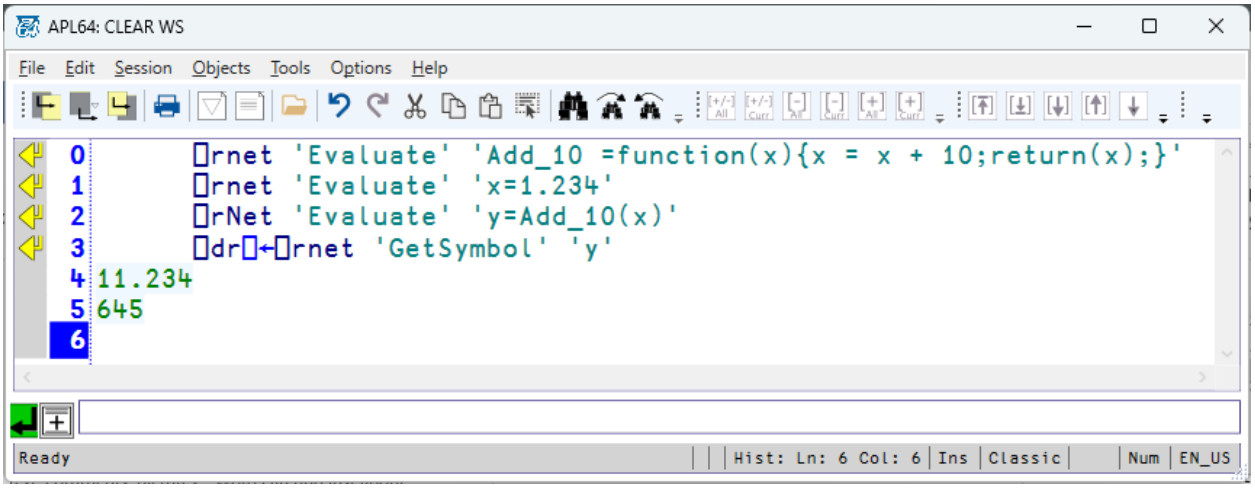
- [ ] RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin'
- [ ] RNET 'Set' 'myVector' ('A1' 'B2')
- [ ] RNET 'Eval' 'iv=is.vector(myVector)'
- [ ] RNET 'Get' 'iv'
- [ ] RNET 'Get' 'myVector'



Creating and Executing an R-Function with One Argument

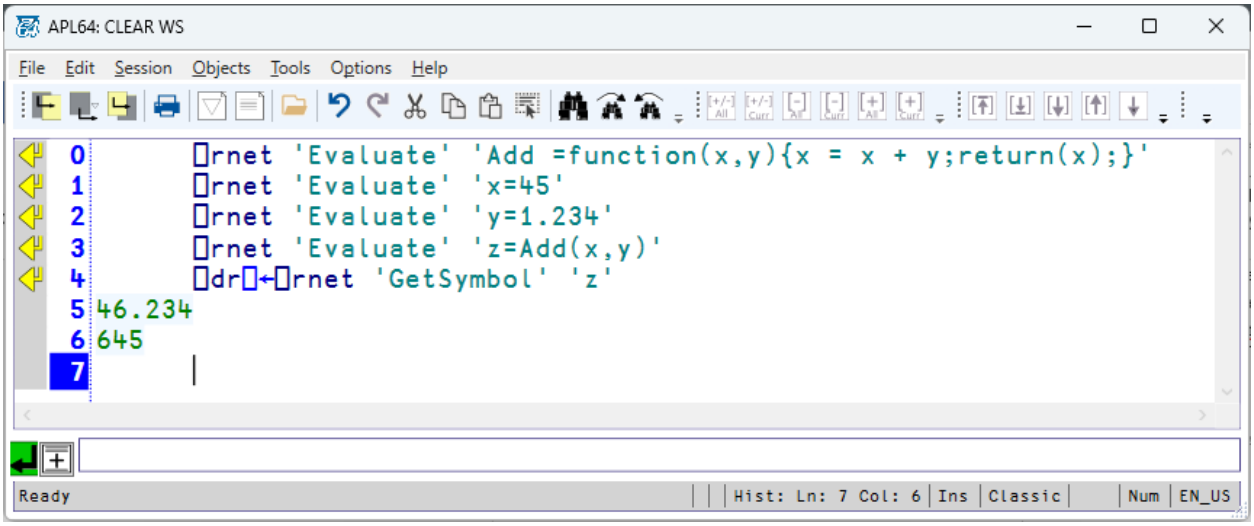
Sometimes it is convenient to create a user-defined function in R so that it can be executed several times during the APL64 session.

```
]rnet 'Evaluate' 'Add_10 =function(x){x = x + 10;return(x);}'
]rnet 'Evaluate' 'x=1.234'
]rNet 'Evaluate' 'y=Add_10(x)'
]dr] ← ]rnet 'GetSymbol' 'y'
```



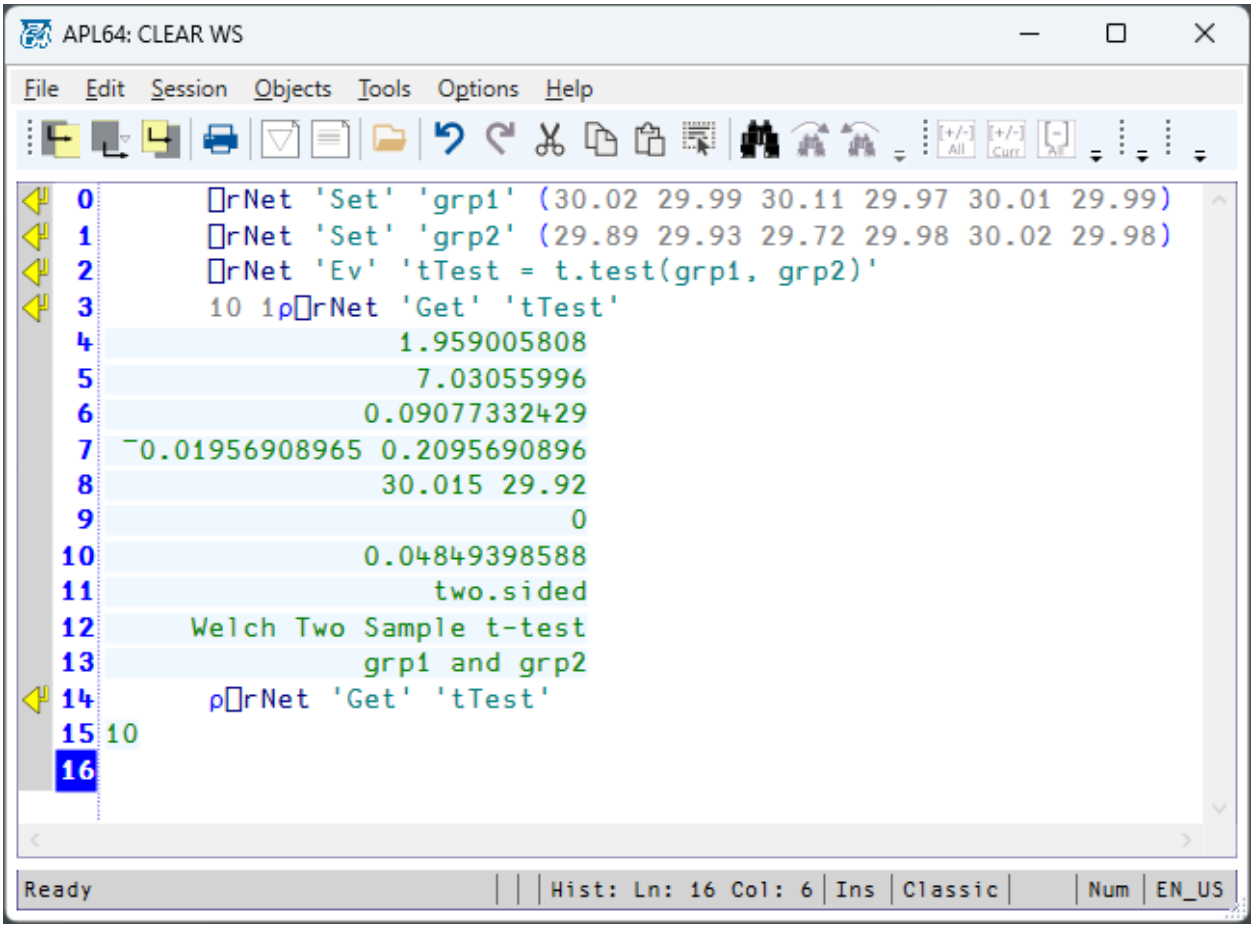
Creating an Executing an R-Function with Multiple Arguments

```
]rnet 'Evaluate' 'Add =function(x,y){x = x + y;return(x);}'
]rnet 'Evaluate' 'x=45'
]rnet 'Evaluate' 'y=1.234'
]rnet 'Evaluate' 'z=Add(x,y)'
]dr] ← ]rnet 'GetSymbol' 'z'
```



T-Test

```
⎕rNet 'Set' 'grp1' (30.02 29.99 30.11 29.97 30.01 29.99)
⎕rNet 'Set' 'grp2' (29.89 29.93 29.72 29.98 30.02 29.98)
⎕rNet 'Ev' 'tTest = t.test(grp1, grp2)'
10 1⍴⎕rNet 'Get' 'tTest'
⍴⎕rNet 'Get' 'tTest'
```



Plot

R-Plot of Random Data

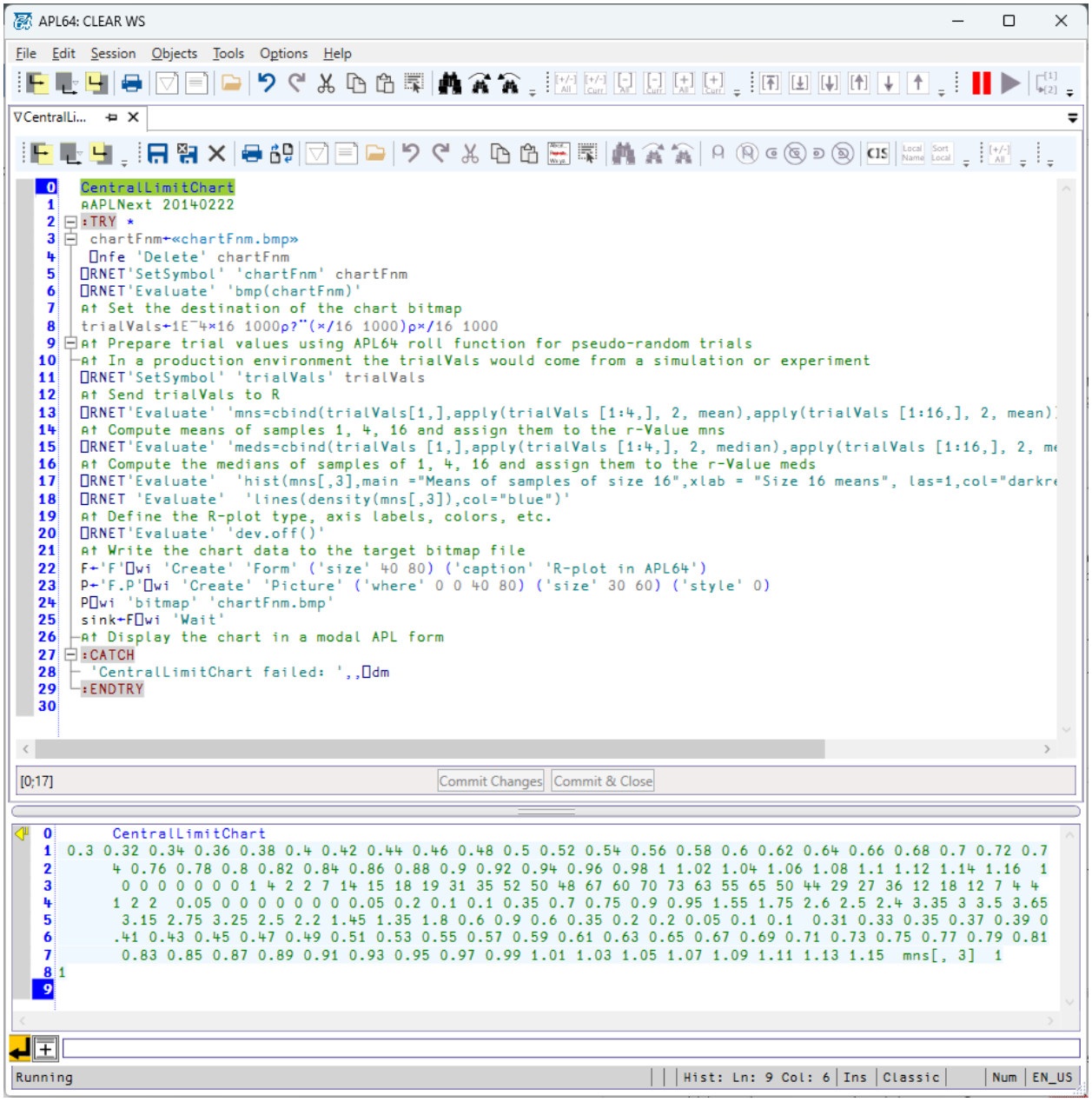
In this example two R-plots are overlaid in the same chart. The first is a histogram of sample 'average' values and the second is a line chart of the same 'average' values illustrating the anticipated observation of the Central Limit Theorem.

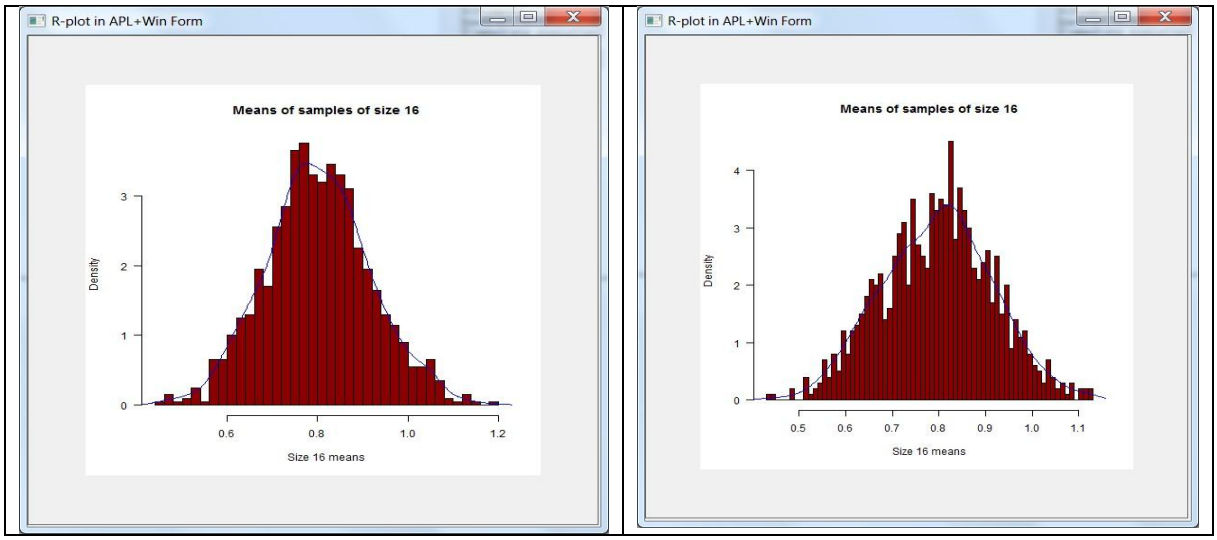
Create this APL function and run it:

```
CentralLimitChart
⍝APLNext 20140222
:TRY *
 chartFnm←«chartFnm.bmp»
 ⎕nfe 'Delete' chartFnm
⎕RNET'SetSymbol' 'chartFnm' chartFnm
⎕RNET'Evaluate' 'bmp(chartFnm)'
⍝↑ Set the destination of the chart bitmap
```

```
trialVals←1E¯4×16 1000ρ?¨(×/16 1000)ρ×/16 1000
⍝↑ Prepare trial values using APL64 roll function for pseudo-random trials
⍝↑ In a production environment the trialVals would come from a simulation or experiment
⎕RNET'SetSymbol' 'trialVals' trialVals
⍝↑ Send trialVals to R
⎕RNET'Evaluate' 'mns=cbind(trialVals[1,],apply(trialVals [1:4,], 2, mean),apply(trialVals [1:16,], 2, mean))'
⍝↑ Compute means of samples 1, 4, 16 and assign them to the r-Value mns
⎕RNET'Evaluate' 'meds=cbind(trialVals [1,],apply(trialVals [1:4,], 2, median),apply(trialVals [1:16,], 2, median)'
⍝↑ Compute the medians of samples of 1, 4, 16 and assign them to the r-Value meds
⎕RNET'Evaluate' 'hist(mns[,3],main ="Means of samples of size 16",xlab = "Size 16 means", las=1,col="darkred", breaks=50, prob = TRUE)'
⎕RNET 'Evaluate' 'lines(density(mns[,3]),col="blue")'
⍝↑ Define the R-plot type, axis labels, colors, etc.
⎕RNET'Evaluate' 'dev.off()'
⍝↑ Write the chart data to the target bitmap file
F←'F'⎕wi 'Create' 'Form' ('size' 40 80) ('caption' 'R-plot in APL64')
P←'F.P'⎕wi 'Create' 'Picture' ('where' 0 0 40 80) ('size' 30 60) ('style' 0)
P⎕wi 'bitmap' 'chartFnm.bmp'
sink←F⎕wi 'Wait'
⍝↑ Display the chart in a modal APL form
:CATCH
 'CentralLimitChart failed: ',,⎕dm
:ENDTRY
```

When the APL64 ‘CentralLimitChart’ function is run, an APL64 form is displayed containing the JPEG-format image containing the R-plot. Since the ‘trial’ data used in this example is based on the APL64 ‘deal’ function, running this function, a few times gives a graphic representation of the APL64 random number generator results.



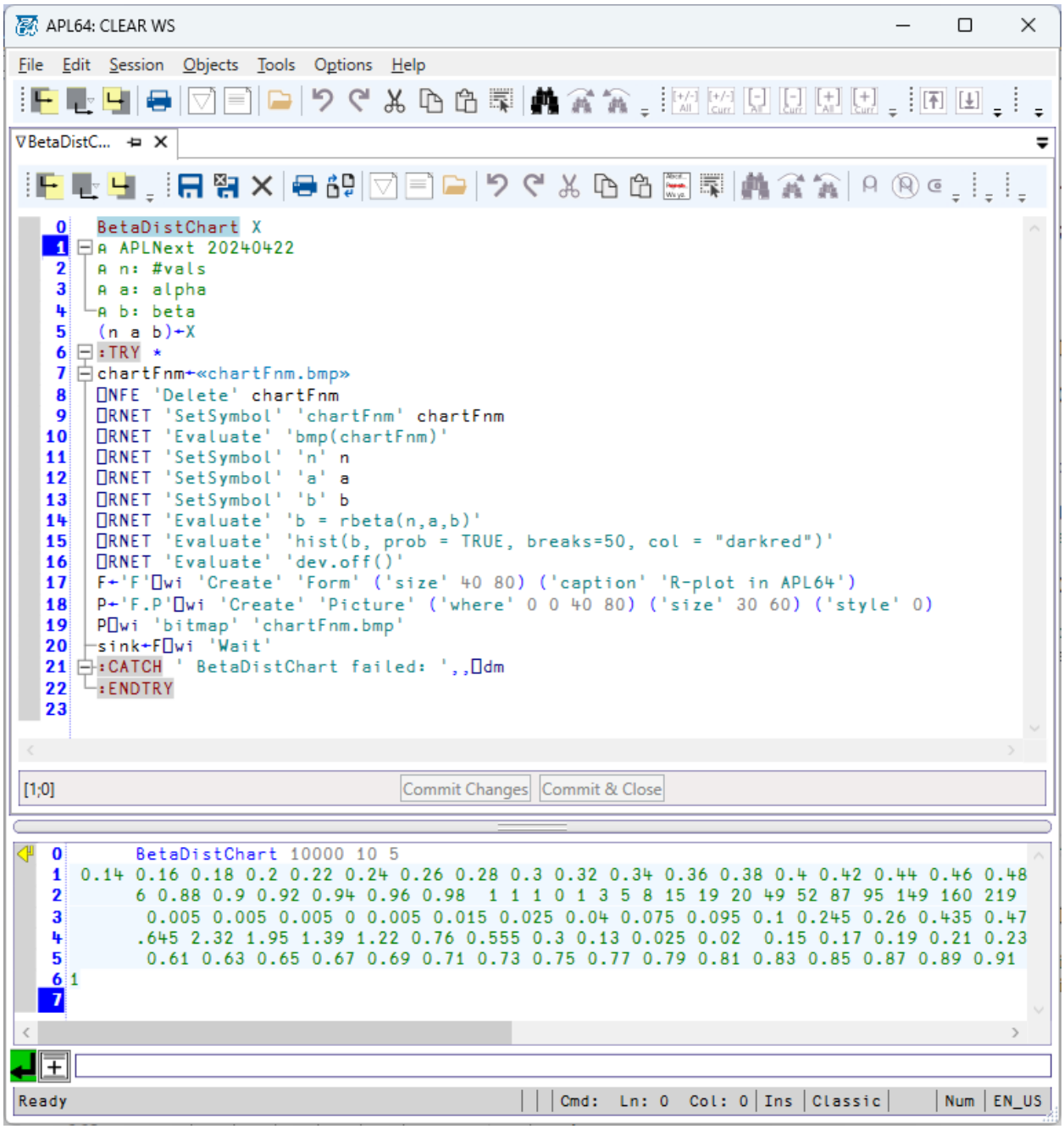


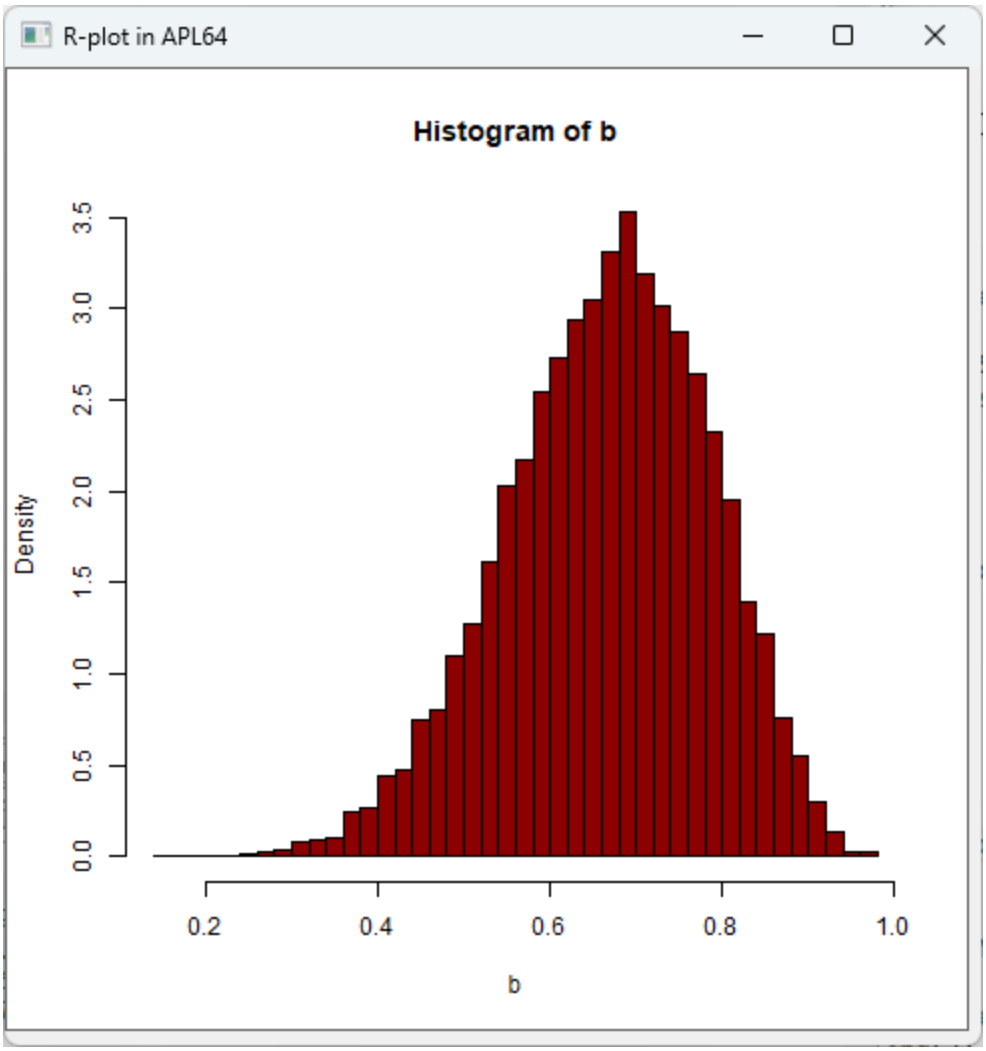
R-Plot of Beta Distribution

The beta distribution has two parameters α and β which provide flexibility to specify an asymmetrical probability distribution with mean 1+1÷β/α in APL operation order. The 'BetaDistChart' rbeta(n, α, β) function to generate n sample values of a beta-distributed random variable and displays the plot in APL64.

```
BetaDistChart X
⍝ APLNext 20240422
⍝ n: #vals
⍝ a: alpha
⍝ b: beta
(n a b)←X
:TRY *
chartFnm←«chartFnm.bmp»
⎕NFE 'Delete' chartFnm
⎕RNET 'SetSymbol' 'chartFnm' chartFnm
⎕RNET 'Evaluate' 'bmp(chartFnm)'
⎕RNET 'SetSymbol' 'n' n
⎕RNET 'SetSymbol' 'a' a
⎕RNET 'SetSymbol' 'b' b
⎕RNET 'Evaluate' 'b = rbeta(n,a,b)'
⎕RNET 'Evaluate' 'hist(b, prob = TRUE, breaks=50, col = "darkred")'
⎕RNET 'Evaluate' 'dev.off()'
F←'F'⎕wi 'Create' 'Form' ('size' 40 80) ('caption' 'R-plot in APL64')
P←'F.P'⎕wi 'Create' 'Picture' ('where' 0 0 40 80) ('size' 30 60) ('style' 0)
P⎕wi  'bitmap' 'chartFnm.bmp'
sink←F⎕wi 'Wait'
```

```
:CATCH ' BetaDistChart failed: ',,⎕dm
:ENDTRY
```

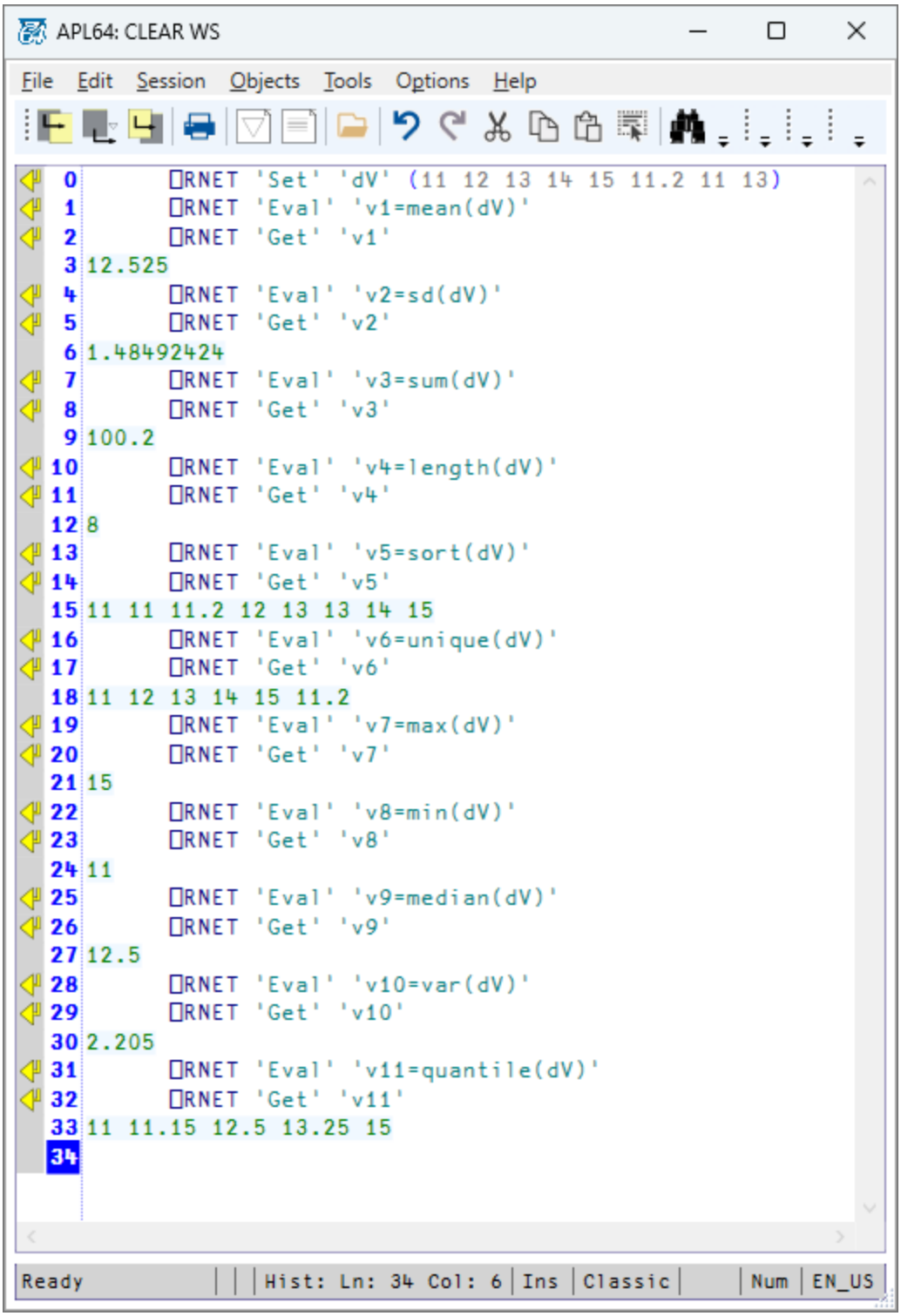




Statistical Values

- [ ] RNET 'Set' 'dV' (11 12 13 14 15 11.2 11 13)
- [ ] RNET 'Eval' 'v1=mean(dV)'
- [ ] RNET 'Get' 'v1'
- [ ] RNET 'Eval' 'v2=sd(dV)'
- [ ] RNET 'Get' 'v2'
- [ ] RNET 'Eval' 'v3=sum(dV)'
- [ ] RNET 'Get' 'v3'
- [ ] RNET 'Eval' 'v4=length(dV)'
- [ ] RNET 'Get' 'v4'
- [ ] RNET 'Eval' 'v5=sort(dV)'
- [ ] RNET 'Get' 'v5'
- [ ] RNET 'Eval' 'v6=unique(dV)'
- [ ] RNET 'Get' 'v6'
- [ ] RNET 'Eval' 'v7=max(dV)'

```
RNET 'Get' 'v7'
RNET 'Eval' 'v8=min(dV)'
RNET 'Get' 'v8'
RNET 'Eval' 'v9=median(dV)'
RNET 'Get' 'v9'
RNET 'Eval' 'v10=var(dV)'
RNET 'Get' 'v10'
RNET 'Eval' 'v11=quantile(dV)'
RNET 'Get' 'v11'
```

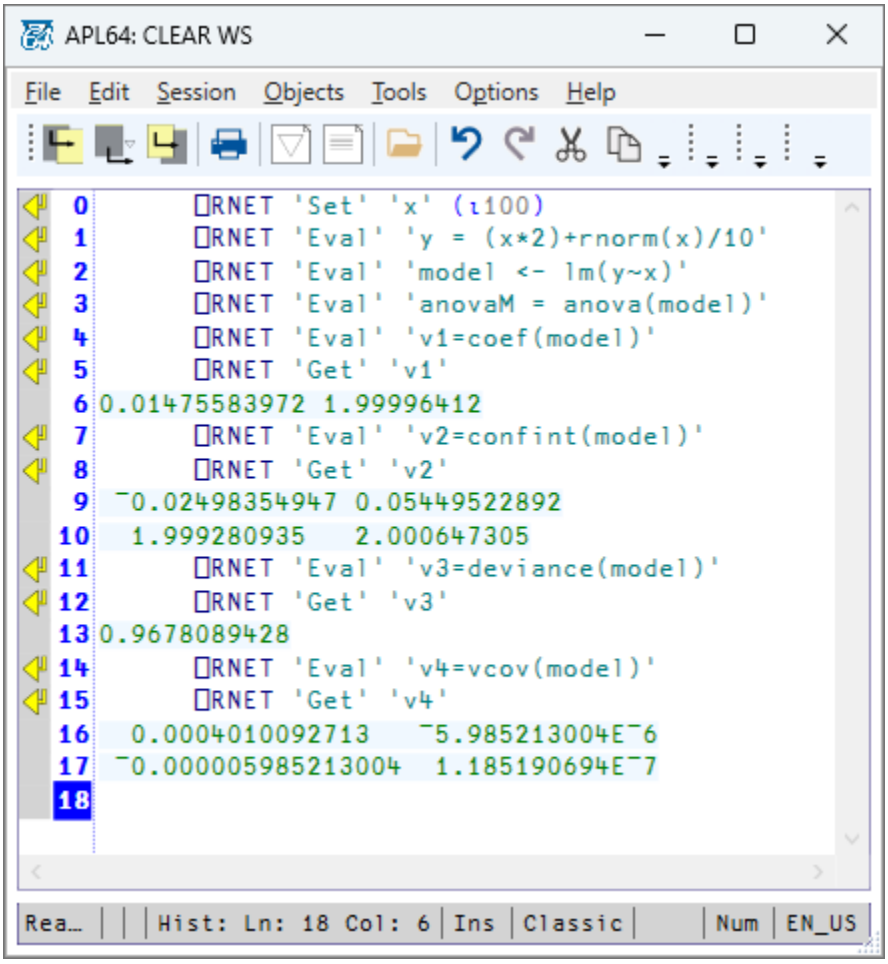


Anova Model

```
⎕RNET 'Set' 'x' (⍳100)
⎕RNET 'Eval' 'y = (x*2)+rnorm(x)/10'
⎕RNET 'Eval' 'model <- lm(y~x)'
```

```
RNET 'Eval' 'anovaM = anova(model)'
RNET 'Eval' 'v1=coef(model)'
RNET 'Get' 'v1'
RNET 'Eval' 'v2=confint(model)'
RNET 'Get' 'v2'
RNET 'Eval' 'v3=deviance(model)'
RNET 'Get' 'v3'
RNET 'Eval' 'v4=vcov(model)'
RNET 'Get' 'v4'
```

Results may vary because the example uses normal distribution samples.

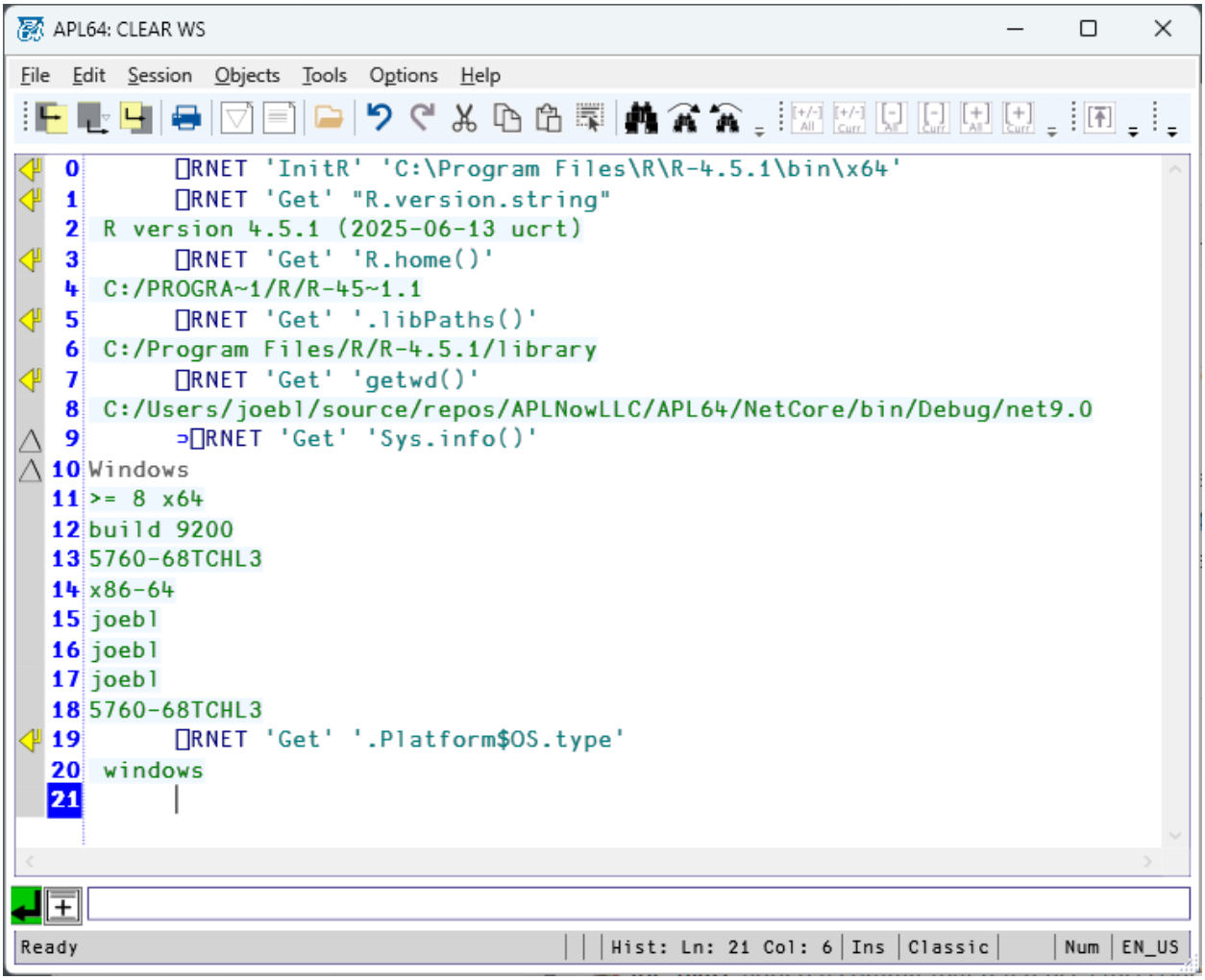


Getting Information about the R Installation under Program Control

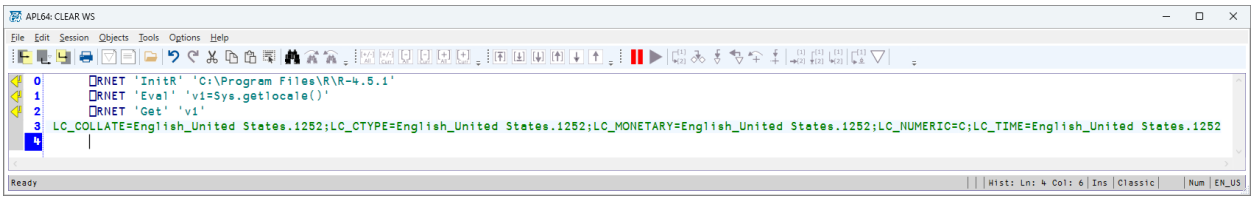
Check the R version being used by RNET.

```
RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
RNET 'Get' "R.version.string"
```

```
⎕RNET 'Get' 'R.home()'
⎕RNET 'Get' '.libPaths()'
⎕RNET 'Get' 'getwd()'
⊃⎕RNET 'Get' 'Sys.info()'
```

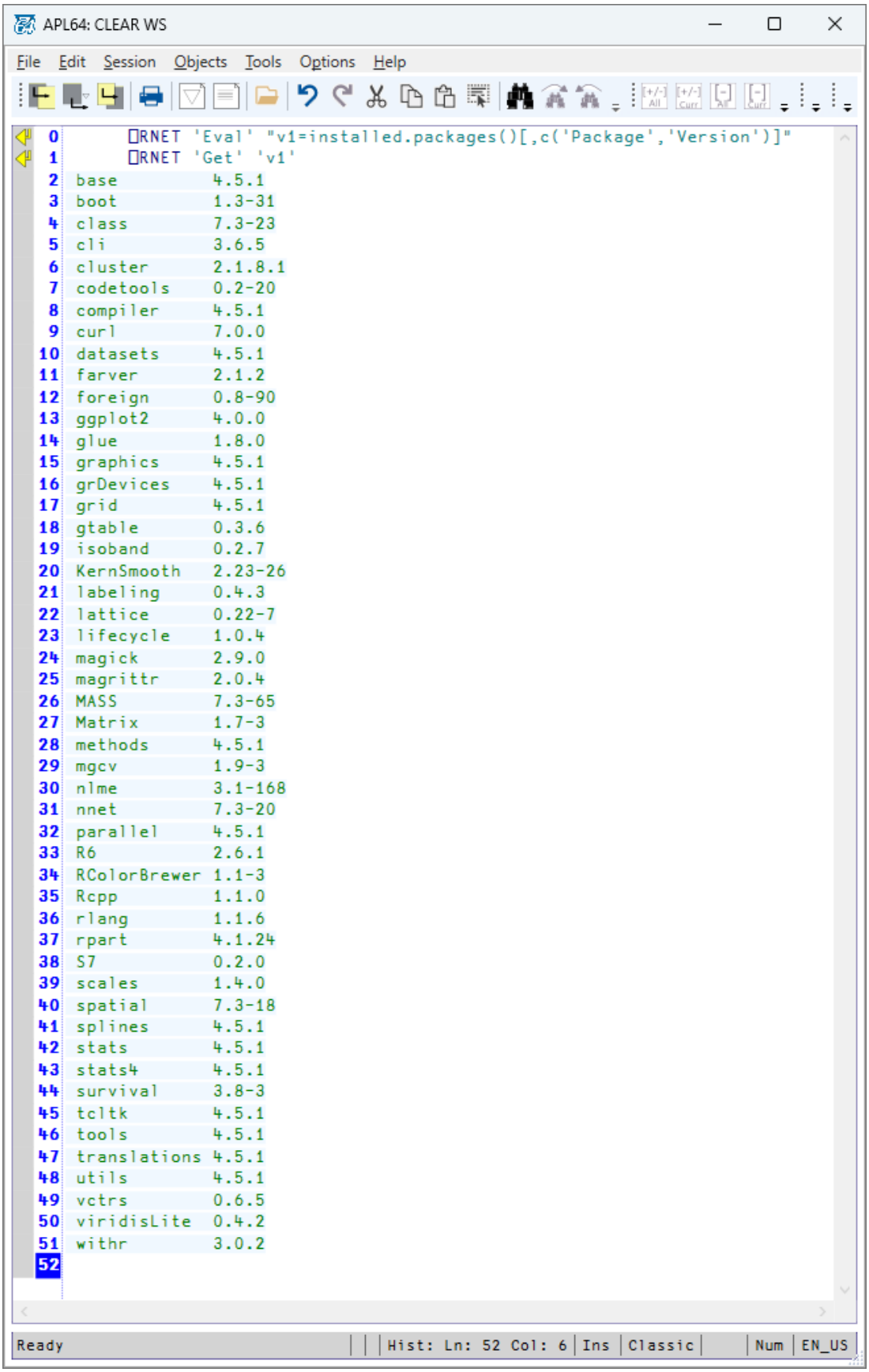



```
⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1'
⎕RNET 'Eval' 'v1=Sys.getlocale()'
⎕RNET 'Get' 'v1'
```



R Packages Installed on the Workstation

```
      RNET 'Eval' "v1=installed.packages()[,c('Package','Version')]"
      RNET 'Get' 'v1'
```

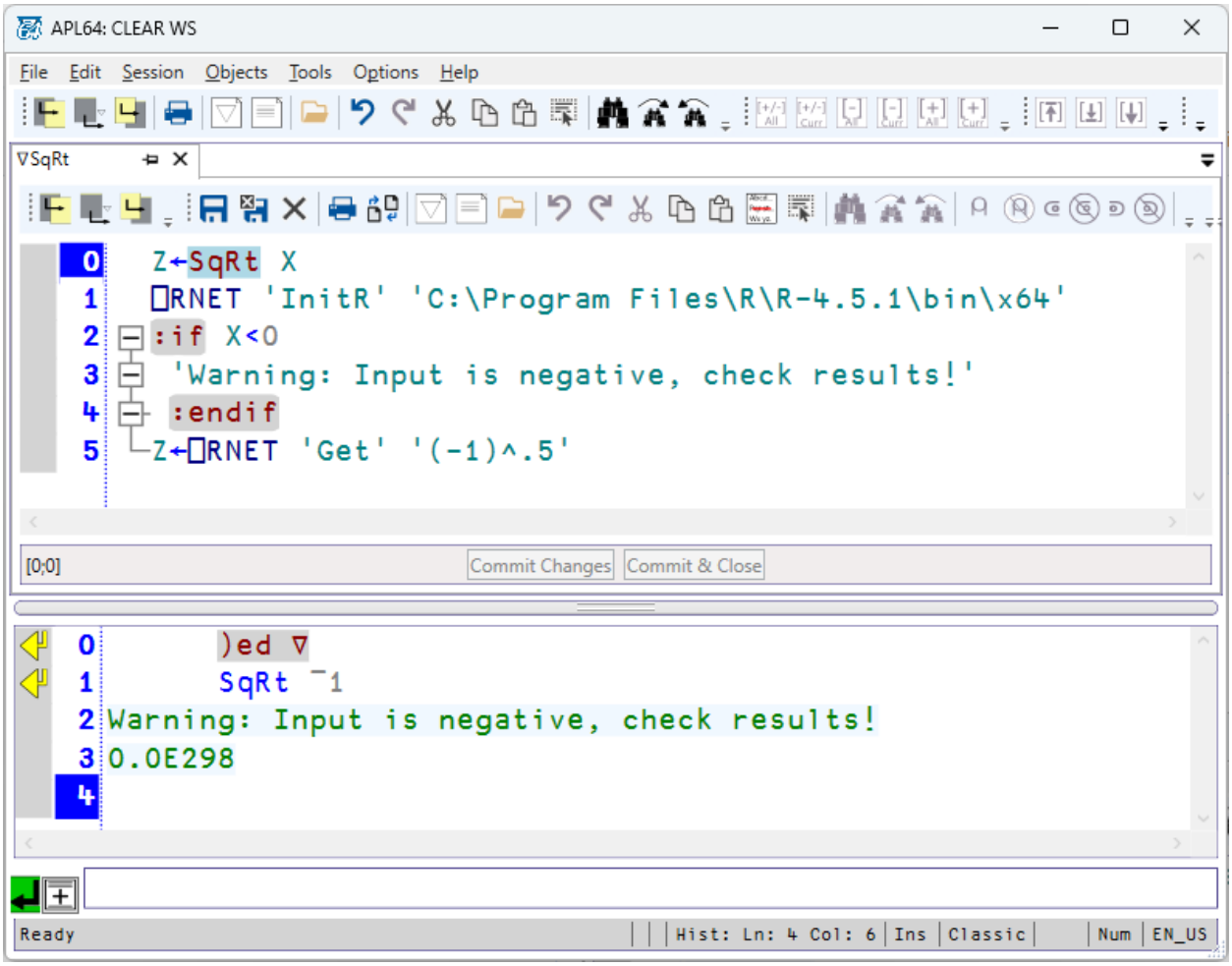


R may not throw an Exception

R may not throw an exception for results which are apparently inappropriate. It is up to the APL64 programmer to validate APL64 input sent to the R environment for processing.

Example: Create this function and run it:

```
Z←SqRt X
 ⎕RNET 'InitR' 'C:\Program Files\R\R-4.5.1\bin\x64'
:if X<0
 'Warning: Input is negative, check results!'
 :endif
Z←⎕RNET 'Get' '(-1)^.5'
```



Resources

What is R?

R is a statistics and graphics toolkit. Its origin is the S language developed at Bell Laboratories. R provides a wide variety of statistics and graphics tools. R is available as no-cost, open-source software under the terms of the Free Software Foundation's GNU General Public License. R is in wide use by many enterprises both public and private for scientific, research and commercial purposes on a worldwide basis, so it incorporates tested tools for statistics and graphics. R is not implemented in .Net, so it is 'unmanaged' code.

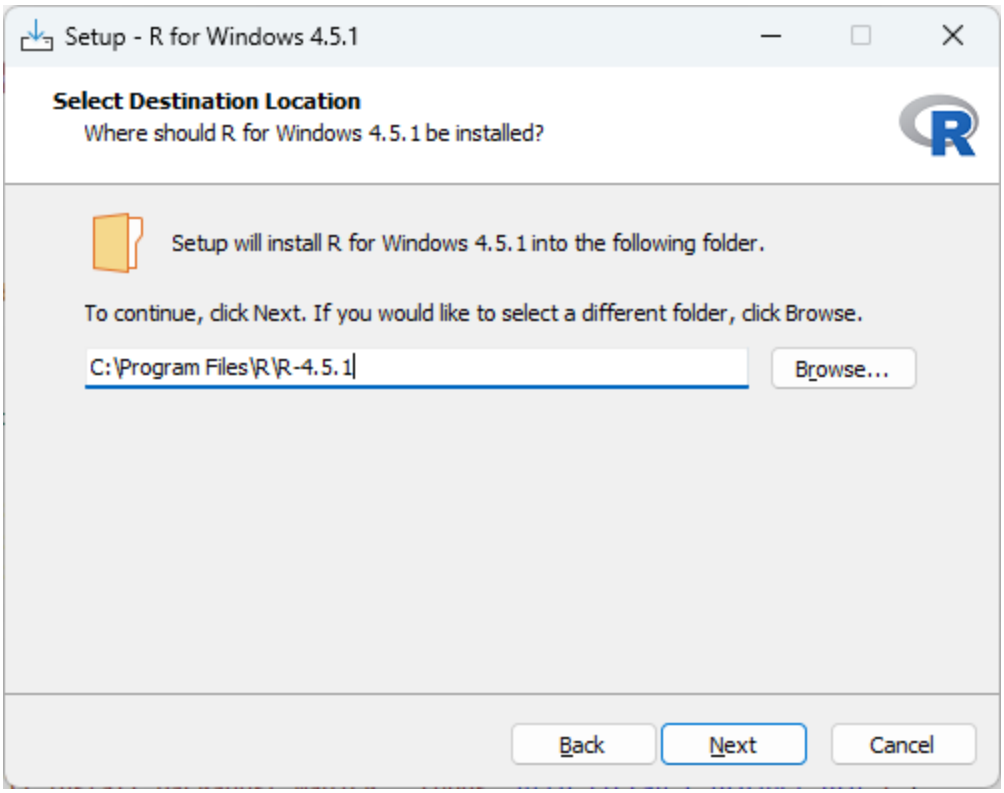
The R-Project

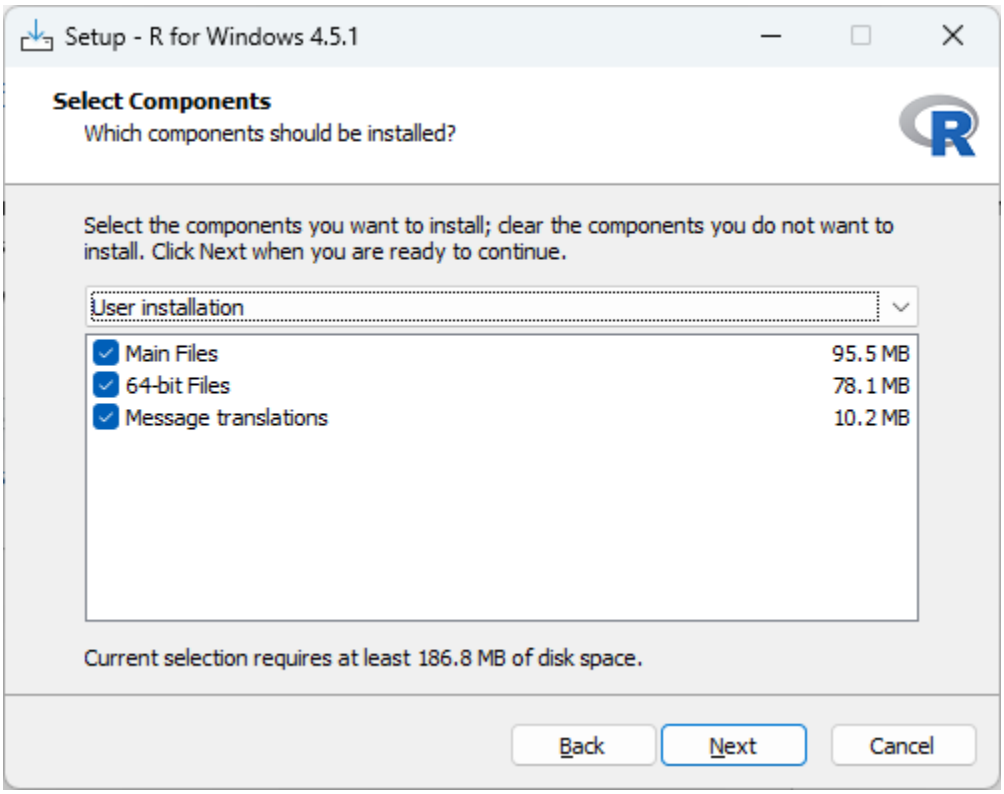
The R-project provides many useful resources.

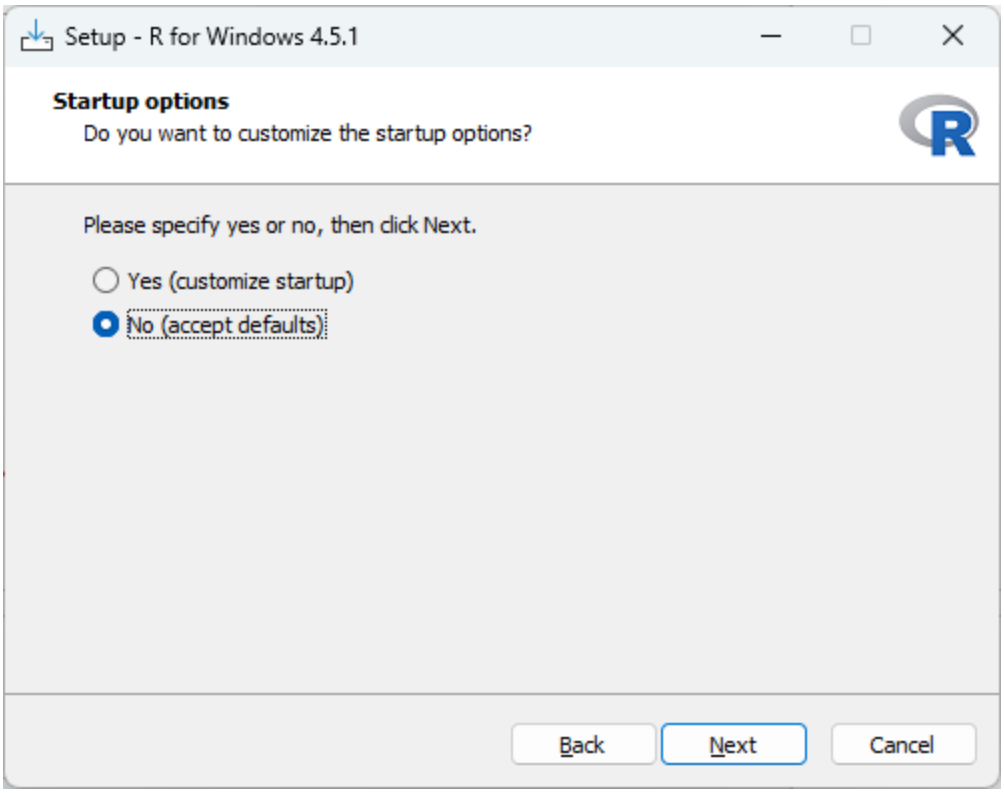
Installing R

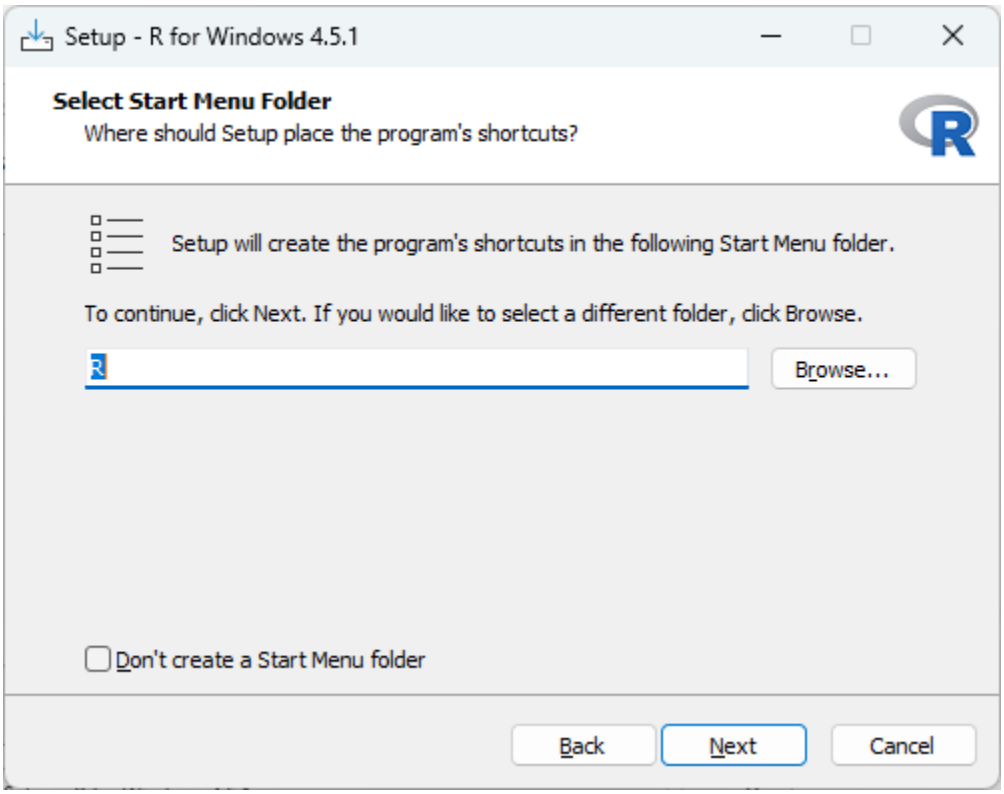
The R license requires that end users install the product, so it is not included in APL64. The R toolkit should be installed to the target workstation. The R toolkit is cross-platform, however the appropriate version of R for the target operating system must be installed. The R installer defaults are sufficient. It may be necessary to run R-installer with elevated credentials, for example, in Windows 'run as administrator'.

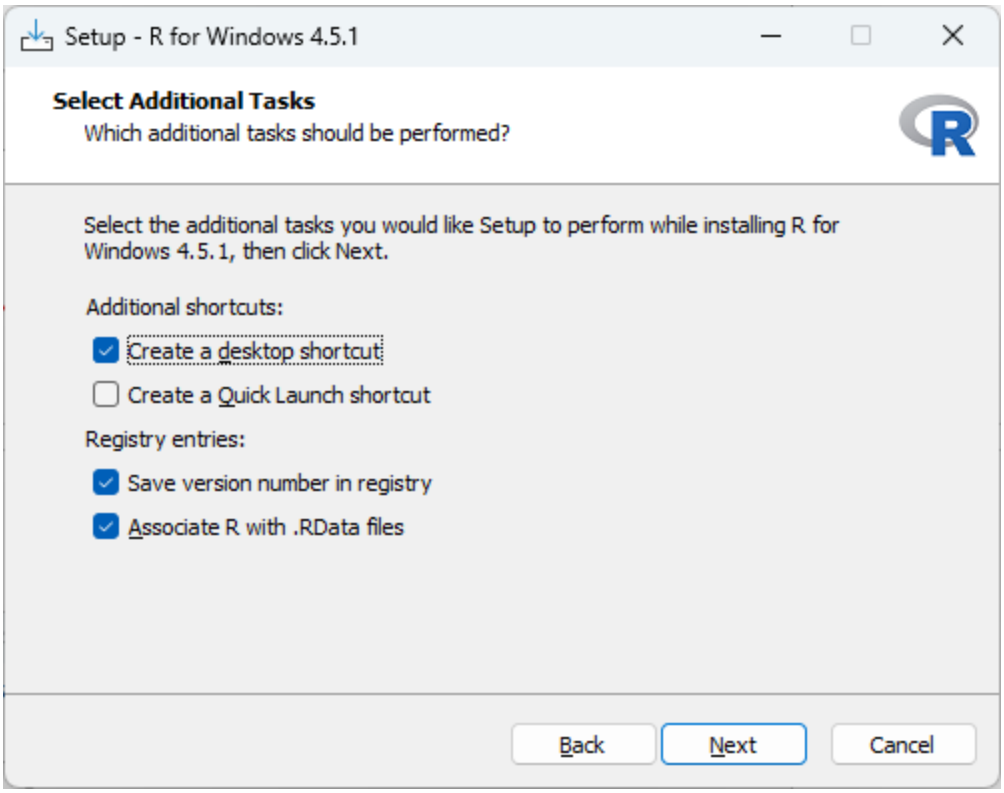
For Windows get the installer here.

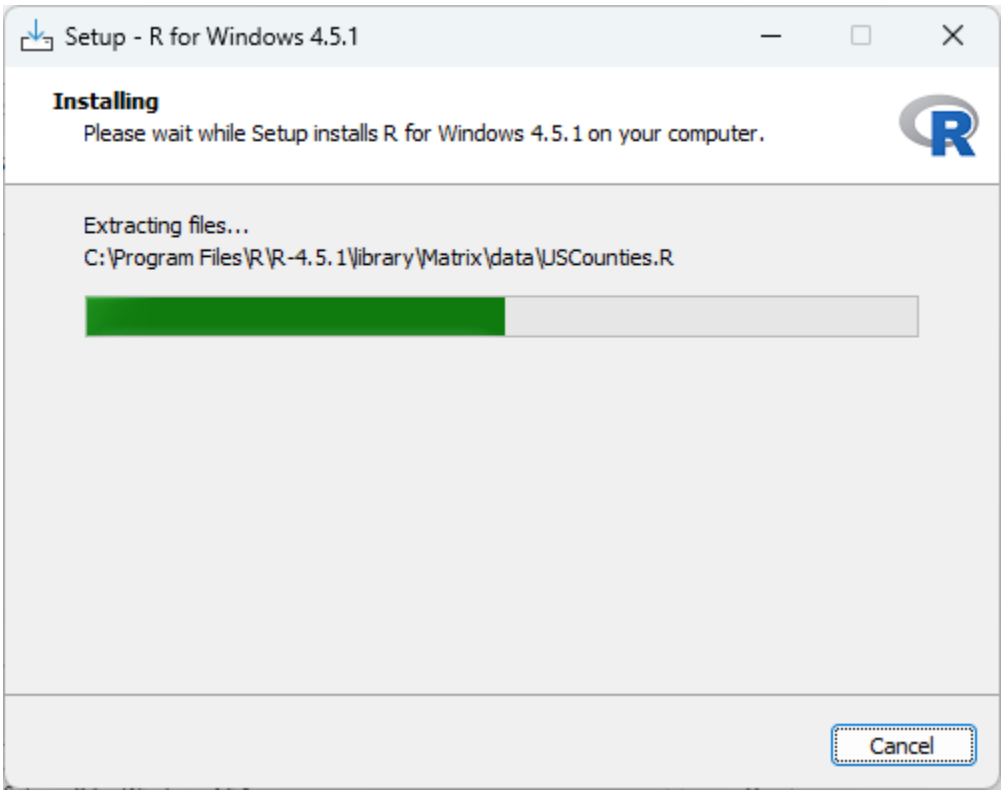




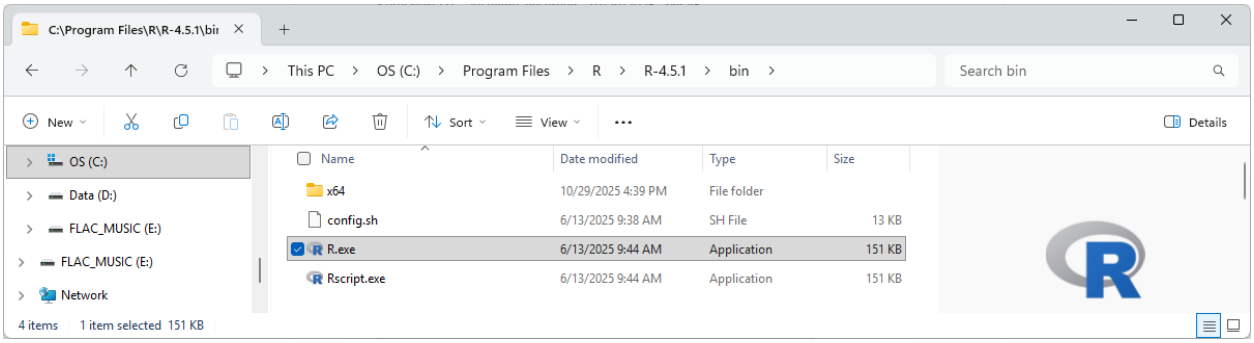




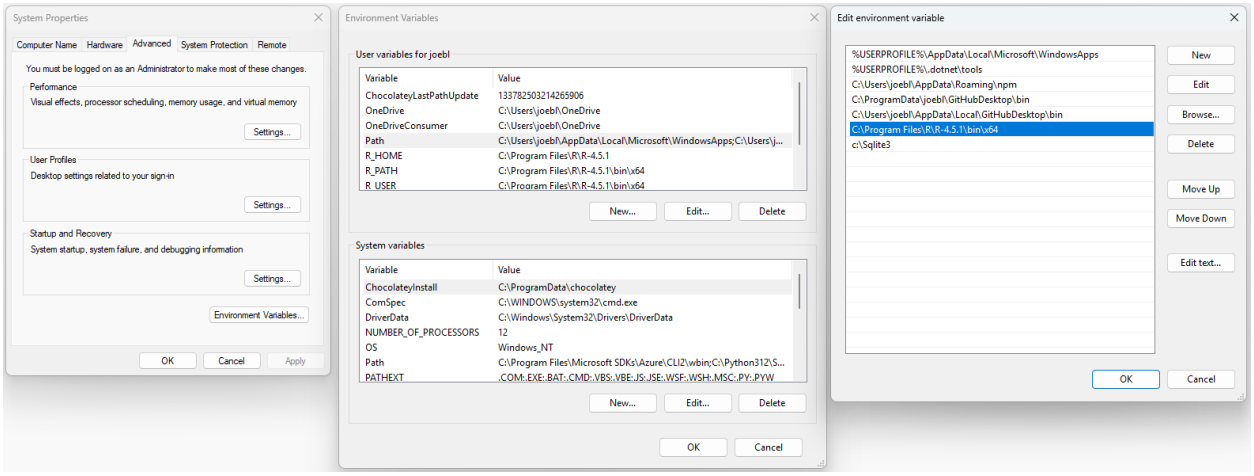




Using Windows Explorer, find the path of R.exe on the target workstation. This path will be needed when using the ☐RNET 'InitR' installationPathOfR action:



In Windows it may be necessary to create or modify the environment variables to provide the path to the R components. The Windows Path environment variable should contain the path to the R components:



Compatibility with APL+Win

The ⎕RNET implementation in APL64 is simplified, is cross-platform, has improved performance, and has new features not available in APL+Win. ⎕RNET does not require the use of the APL64 ⎕CSE system function, or deprecated ActiveX technology. Because ⎕RNET is designed to be cross-platform, images created by ⎕RNET are necessarily file-based as there is no universal platform to display images in an application. R-images, such as those create by Plot or GGPlot2 can be included within a ⎕WI form.

R-Tutorials

This document is not an R-tutorial. Here is a good R-tutorial as well as this one.

R.Net

R.NET is a .Net interface to R used to implement ⎕RNET.