Oracle® R Enterprise

User's Guide

Release 11.2 for Linux and Microsoft Windows

E26499-04

March 2012

ORACLE®

Oracle R Enterprise User's Guide, Release 11.2 for Linux and Microsoft Windows

E26499-04

Primary Author: Margaret Taft

Contributing Author:

Contributor:

Contents

3 Using Oracle R Enterprise

4 Oracle R Enterprise Statistical Functions

A Third-Party Licenses

Index

Preface

This book describes how to install and use Oracle R Enterprise.

Audience

This document is intended for anyone responsible for installing Oracle R Enterprise and for anyone who uses Oracle R Enterprise. Installation and use of Oracle R Enterprise requires knowledge of R and the Oracle Database.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following document:

- *Oracle R Enterprise Release Notes*

For information about Oracle Database, see the O*racle Database Documentation Library 11g Release 2 (11.2)* at `http://www.oracle.com/technetwork/indexes/documentation/index.html?ssSourceSiteId=ocomen`.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.

Convention	Meaning
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in Oracle R Enterprise 11.2?

This section describes new features in releases of Oracle R Enterprise 11.2. It includes the following sections:

- New Features for Release 1.1

New Features for Release 1.1

Release 1.1 includes these new features:

- **Improved mathematics libraries in R**

  You can now use the improved Oracle R Distribution with support for dynamically picking up either the Intel Math Kernel Library (MKL) or the AMD Core Math Library (ACML) with Oracle R Enterprise.

- **Server runs on Windows**

  The Oracle R Enterprise server now runs on 64-bit and 32-bit Windows operating systems.

- **Support for Oracle Wallet**

  R scripts no longer need to have database authentication credentials in clear text. Oracle R Enterprise is integrated with Oracle Wallet for that purpose.

- **Improved installation**

  The installation scripts have been improved with more prerequisite checks and detailed error messages. Error messages provide specific instructions on remedial actions.

1

Overview of Oracle R Enterprise

R is an open source statistical programming language and environment. For information about R, see the R Project for Statistical Computing at http://www.r-project.org.

R provides an environment for statistical computing, including:

- An easy-to-use language
- A powerful graphical environment for visualization
- Many out-of-the-box statistical techniques
- R packages (An R package is a set of related functions, help files, and data files; currently, there are more than 3340 R packages.)
- The R Console graphical user interface for analyzing data interactively

R's rapid adoption has earned it a reputation as a new statistical software standard.

Oracle R Enterprise is a component of the Oracle Advanced Analytics Option of Oracle Database Enterprise Edition. For detailed information about Oracle R Enterprise, including links to software downloads, go to **Oracle R Enterprise** at http://www.oracle.com/technetwork/database/options/advanced-analytics/r-enterprise/index.html.

Oracle R Enterprise allows users to perform statistical analysis on data stored in tables in an Oracle Database. Oracle R Enterprise has these components:

- The Oracle R Enterprise R **transparency layer**. The transparency layer is a collection of packages that support mapping of R data types to Oracle Database objects and generate SQL transparently in response to R expressions on mapped data types. The transparency layer allows an R user to directly interact with database-resident data using R language constructs. This enables R users to work with data too large to fit into the memory of a user's desktop system.
- The Oracle **statistics engine**, a collection of statistical functions and procedures corresponding to commonly-used statistical libraries. The statistics engine packages execute in Oracle Database.
- **SQL extensions** supporting R engine execution through the database on the database server. These SQL extensions enable productizing R scripts, that is, running R scripts in a lights-out mode.
- **Oracle R Connector for Hadoop** is an R package executing MapReduce jobs that enables R users to directly work with an Oracle Hadoop cluster executing computations written in the R language and working on data resident in HDFS, Oracle database or local files.

The components of Oracle R Enterprise are described in Chapter 3.

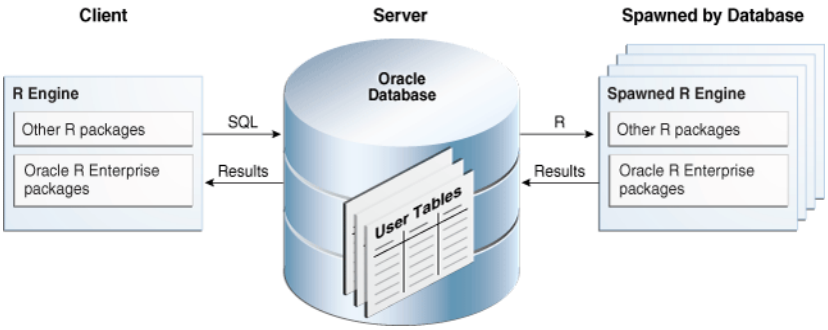
Oracle R Connector for Hadoop is a related product that is part of the Big Data Appliance.

Oracle R Enterprise also includes functions that perform most common or base statistical procedures; see Chapter 4 for more information.

The rest of this chapter describes Oracle R Enterprise Architecture, Oracle R Enterprise Data Types, and Oracle R Enterprise Supported Configurations.

Oracle R Enterprise Architecture

Oracle R Enterprise has these three components including the connector for Hadoop:



1. The **Client R Engine** is a collection of R packages that allows you to connect to an Oracle Database and to interact with data in that database. The ORCH package allows you to connect to an Oracle Hadoop cluster and interact with data in HDFS files; the package also allows the execution of MapReduce jobs.

   You can use any R commands from the client. In addition, the client supplies these functions:

   - The R SQL Transparency framework intercepts R functions for scalable in-database execution
   - Functions intercept data transforms, statistical functions, and Oracle R Enterprise-specific functions
   - Interactive display of graphical results and flow control as in open source R
   - Submission of R closures (functions) for execution in the Oracle Database

2. The **Server** is a collection of PL/SQL procedures and libraries that augment Oracle Database with the capabilities required to support an Oracle R Enterprise client. The R engine is also installed on Oracle Database to supported embedded R execution. Oracle Database spawns R engines, which can provide data parallelism.

   The Oracle R Enterprise Database engine provides this functionality:

   - Scale to large datasets
   - Access to tables, views, and external tables in the database, as well as those accessible through database links
   - Use SQL query parallel execution
   - Use in-database statistical and data mining functionality

3. **R Engines spawned by Oracle Database** are spawned to support database-managed parallelism; provide lights-out scheduled execution of R scripts, that is, scheduling or triggering R scripts packaged inside a PL/SQL or SQL query. Oracle R Enterprise provides efficient transfer to and from the spawned engines. Embedded R execution can be used to emulate MapReduce style programming.

There are several data types specific to Oracle R Enterprise; see Oracle R Enterprise Data Types for details.

Oracle R Connector for Hadoop

The Oracle R Connector for Hadoop (ORCH) is an R package that allows an ORE client to interact with and execute MapReduce jobs on the Oracle Hadoop cluster.

For information about ORCH, see the *Oracle Big Data Connectors User's Guide* (`http://docs.oracle.com/cd/E27101_01/doc.10/e27365/toc.htm`), part of the *Oracle Big Data Documentation* library (`http://docs.oracle.com/cd/E27101_01/index.htm`).

Oracle R Enterprise Data Types

Oracle R Enterprise introduces a variant to many R data types. The name of the Oracle R Enterprise data type is the name of the corresponding R data type prefixed by *ore*. These data types establish a mapping between an R object and a database table or view. The mapping tracks metadata of the Oracle object which in turn aids in SQL query generation. These data types form the foundation of the Oracle R Enterprise transparency layer.

The following R data types have been overloaded for transparent in-database execution:

- Character, Integer, Numeric and Logical vectors
- Factors
- Data Frame
- Matrix is overloaded in two situations:
  - Linear algebra cross-products
  - Creating input matrices for advanced analytics

For more information and examples, see Oracle R Enterprise Transparency Framework on page 3-4.

Oracle R Enterprise Supported Configurations

Oracle R Enterprise consists of a client and a server. The client runs on Microsoft Windows, Oracle Linux, or Red Hat Linux; the server runs on Microsoft Windows, Oracle Linux, or Red Hat Linux. The server includes an Oracle Database, to which the client connects. Oracle R Enterprise also runs on Oracle Exadata machines with the Linux operating system. For details, see Prerequisites on page 2-1.

Installation of Oracle R Enterprise is described in Chapter 2.

2 Installation

Follow these steps to install Oracle R Enterprise on your Windows PC and Linux system:

1. Make sure that the Prerequisites are satisfied
2. Install Oracle R Enterprise
   a. Install Client on Microsoft Windows and/or Install Client on Linux
   b. Install Server on Microsoft Windows or Install Server on Linux
3. If you have Oracle R Enterprise 1.0 installed, you can Upgrade Oracle R Enterprise.

   You can upgrade 1.0 release version of Oracle R Enterprise only. You cannot upgrade Beta versions.
4. If necessary, you can Uninstall Oracle R Enterprise

Prerequisites

First decide which of the Oracle R Enterprise Supported Configurations that you will use.

Then install the required software before you install Oracle R Enterprise:

- Linux and/or Windows
- R
- Oracle Database

Linux and/or Windows

Verify that one or both of these operating systems is installed:

- Oracle R Enterprise server is supported on Linux x86-64 and Microsoft Windows 32-bit:
  - Microsoft Windows XP, Vista, or Windows 7, 32-bit
  - Oracle Linux Release 5 Update 6 or higher
  - Red Hat Linux 5 Update 6 or higher

  Oracle R Enterprise server is also supported on Oracle Exadata running Oracle Linux.
- Oracle R Enterprise client is supported on Linux x86-64 and Microsoft Windows 32-bit or 64-bit:

- Microsoft Windows XP, Vista, or Windows 7, 32-bit or 64-bit
- Oracle Linux Release 5 Update 6 or higher
- Red Hat Linux 5 Update 6 or higher

To download Oracle Linux Release 5 Update 6 Media Pack for x86_64 (64 bit), go to http://www.oracle.com/us/technologies/linux/index.html).

R

On your client system, download and install R 2.13.2. You can download R from http://www.r-project.org or any source that provides R.

Note: Oracle R Enterprise is certified with R 2.13.2 only.

You must install R on both the client and the server.

Installation of R depends on the platform:

- Install R on Windows
- Install the Oracle R Distribution on Linux
- Install the Oracle R Distribution on Oracle Exadata

Install R on Windows

Follow these steps to install R 2.13.2 on Windows:

1. Go to the CRAN Mirror of your choice.
2. Click **Download R for Windows**.
3. Click **base**.
4. Under the heading **Other builds**, click **Previous releases** in the third bullet.
5. Click **R 2.13.2 (September, 2011)** to start the download.

When the download completes, double click `R-2.13.2-win.exe` to launch the Windows installer for R. Follow the instructions to complete the installation.

Install the Oracle R Distribution on Linux

This section describes how to install Oracle's distribution of Open Source R on Oracle Linux or Red Hat Linux. To install the Oracle distribution on Oracle Exadata Machine, see Install the Oracle R Distribution on Oracle Exadata

Go to http://public-yum.oracle.com/ and follow these steps to install R:

1. Install the yum repos as follows:

   ```
   cd /etc/yum.repos.d
   wget http://public-yum.oracle.com/public-yum-el5.repo
   ```

2. Using the text editor of your choice, set "enabled=1" for `[el5_addons]` and `[ol5_u6_base]` in the file `/etc/yum.repos.d/public-yum-el5.repo`.
3. This step is optional.

   Type the following commands in a shell to check that your yum repository is configured correctly:

   ```
   sudo yum repolist
   ```

The output should look like this:

```
el5_addons                                | 951 B     00:00
ol5_u6_base                                | 1.1 kB     00:00
repo id              repo name                                status
base                 Red Hat Linux - Base                     enabled: 3,024
el5_addons           Enterprise Linux 5 - x86_64 - addons     enabled:    93
ol5_u6_base          Oracle Linux 5 - U6 - x86_64 - base      enabled: 4,551
```

Make sure that both the `el5_addons` and `ol5_u6_base` repos are listed. The list of repos can be different depending on the Linux version and current user's configuration of yum.

4. To install R, use the following command:

```
sudo yum install R.x86_64
```

If the installation is successful, you should see the following messages. The list of required packages may be longer (if you install ORE on Oracle Exadata) or shorter depending on your specific Linux environment and packages installed prior to this R installation.

Make sure to look for the keywords `Dependencies Resolved`.

```
Dependencies Resolved

================================================================================
Package        Arch       Version            Repository      Size
================================================================================
Installing:
R              x86_64     2.13.2-4.el5       el5_addons      15 k
Installing for dependencies:
R-core         x86_64     2.13.2-4.el5       el5_addons      28 M
R-devel        x86_64     2.13.2-4.el5       el5_addons      89 k
dialog         x86_64     1.0.20051107-1.2.2 ol5_u6_base     165 k
libRmath       x86_64     2.13.2-4.el5       el5_addons      112 k
libRmath-devel x86_64     2.13.2-4.el5       el5_addons      19 k
pcre-devel     x86_64     6.6-6.el5          ol5_u6_base     184 k
tcl-devel      x86_64     8.4.13-4.el5       ol5_u6_base     1.0 M
tetex          x86_64     3.0-33.8.el5_5.6   ol5_u6_base     8.8 M
tetex-dvips    x86_64     3.0-33.8.el5_5.6   ol5_u6_base     574 k
tetex-fonts    x86_64     3.0-33.8.el5_5.6   ol5_u6_base     29 M
tetex-latex    x86_64     3.0-33.8.el5_5.6   ol5_u6_base     4.2 M
tk-devel       x86_64     8.4.13-5.el5_1.1   ol5_u6_base     808 k
Updating for dependencies:
pcre           x86_64     6.6-6.el5          ol5_u6_base     118 k

Transaction Summary
================================================================================
Install    13 Package(s)
Upgrade     1 Package(s)

Total download size: 73 M
Is this ok [y/N]:
```

Install the Oracle R Distribution on Oracle Exadata

Using `http://public-yum.oracle.com/` is the recommended way to install R on Oracle Exadata. However, it may not be possible to use `http://public-yum.oracle.com/`.

If you cannot use `http://public-yum.oracle.com/`, you can follow these steps to install R using RPMs.

> **Note:** These directions work for Oracle Linux Release 5 only.

The required RPMs are in one of these locations

- http://public-yum.oracle.com/repo/EnterpriseLinux/EL5/5/base/x86_64/
- The three libMath RPMs (`libSM-1.0.1-3.1.x86_64.rpm`, `libRmath-static-2.13.2-4.el5.x86_64.rpm`, and `libRmath-2.13.2-4.el5.x86_64.rpm` are in http://public-yum.oracle.com/repo/EnterpriseLinux/EL5/addons/x86_64/

Follow these steps to install the Oracle R Distribution on Oracle Exadata:

1. Make sure that the following RPMs are installed. If any are missing, download and install them:

```
rpm -Uvh libsmi-devel-0.4.5-2.el5.x86_64.rpm
rpm -Uvh libsmi-0.4.5-2.el5.x86_64.rpm
rpm -Uvh libSM-devel-1.0.1-3.1.x86_64.rpm
rpm -Uvh libsmbclient-devel-3.0.33-3.28.el5.x86_64.rpm
rpm -Uvh libsmbclient-3.0.33-3.28.el5.x86_64.rpm
rpm -Uvh libSM-1.0.1-3.1.x86_64.rpm
rpm -Uvh libRmath-static-2.13.2-4.el5.x86_64.rpm
rpm -Uvh libRmath-devel-2.13.2-4.el5.x86_64.rpm
rpm -Uvh libRmath-2.13.2-4.el5.x86_64.rpm
rpm -Uvh libpng-1.2.10-7.1.el5_3.2.x86_64.rpm
rpm -Uvh libjpeg-6b-37.x86_64.rpm
rpm -Uvh libICE-devel-1.0.1-2.1.x86_64.rpm
rpm -Uvh libICE-1.0.1-2.1.x86_64.rpm
rpm -Uvh libgssapi-0.10-2.x86_64.rpm
rpm -Uvh libgsf-1.14.1-6.1.x86_64.rpm
rpm -Uvh libFS-1.0.0-3.1.x86_64.rpm
rpm -Uvh libfontenc-1.0.2-2.2.el5.x86_64.rpm
```

2. Download and install these RPMs in the order in which they are listed:

```
rpm -Uvh cairo-1.2.4-5.el5.x86_64.rpm
rpm -Uvh libtiff-3.8.2-7.el5_3.4.x86_64.rpm
rpm -Uvh bitstream-vera-fonts-1.10-7.noarch.rpm
rpm -Uvh pango-1.14.9-6.el5.x86_64.rpm
rpm -Uvh cups-libs-1.3.7-18.el5.x86_64.rpm
rpm -Uvh paps-0.6.6-19.el5.x86_64.rpm
rpm -Uvh atk-1.12.2-1.fc6.x86_64.rpm
rpm -Uvh hicolor-icon-theme-0.9-2.1.noarch.rpm
rpm -Uvh gtk2-2.10.4-20.el5.x86_64.rpm
rpm -Uvh poppler-0.5.4-4.4.el5_4.11.x86_64.rpm
rpm -Uvh poppler-utils-0.5.4-4.4.el5_4.11.x86_64.rpm
rpm -Uvh dbus-python-0.70-9.el5_4.x86_64.rpm
rpm -Uvh avahi-0.6.16-7.el5.x86_64.rpm
rpm -Uvh avahi-compat-libdns_sd-0.6.16-7.el5.x86_64.rpm
rpm -Uvh cups-1.3.7-18.el5.x86_64.rpm
rpm -Uvh netpbm-10.35.58-8.el5.x86_64.rpm
rpm -Uvh desktop-file-utils-0.10-7.x86_64.rpm
rpm -Uvh dialog-1.0.20051107-1.2.2.x86_64.rpm
rpm -Uvh ed-0.2-39.el5_2.x86_64.rpm
rpm -Uvh tetex-fonts-3.0-33.8.el5.x86_64.rpm
rpm -Uvh tetex-3.0-33.8.el5.x86_64.rpm
rpm -Uvh tetex-dvips-3.0-33.8.el5.x86_64.rpm
rpm -Uvh libFS-1.0.0-3.1.x86_64.rpm
rpm -Uvh xorg-x11-xfs-utils-1.0.2-4.x86_64.rpm
rpm -Uvh xorg-x11-font-utils-7.1-2.x86_64.rpm
rpm -Uvh ttmkfdir-3.0.9-23.el5.x86_64.rpm
rpm -Uvh chkfontpath-1.10.1-1.1.x86_64.rpm xorg-x11-xfs-1.0.2-4.x86_64.rpm
rpm -Uvh urw-fonts-2.3-6.1.1.noarch.rpm
rpm -Uvh ghostscript-8.15.2-9.11.el5.x86_64.rpm
ghostscript-fonts-5.50-13.1.1.noarch.rpm
```

```
rpm -Uvh netpbm-progs-10.35.58-8.el5.x86_64.rpm
rpm -Uvh tetex-latex-3.0-33.8.el5.x86_64.rpm
```

3. Finally download and install the core RPM for R:

```
rpm -Uvh R-core-2.13.2-4.el5.x86_64.rpm
```

Oracle Database

Oracle R Enterprise requires Oracle 11.2 Enterprise Edition for Microsoft Windows, Oracle Linux or Red Hat Linux; the Oracle Database can be installed on Oracle Linux, Red Hat Linux, or Oracle Exadata running Oracle Linux.

In order for certain Oracle R Enterprise functionality to work properly, the Oracle Database must include the patch that fixes bug number 11678127.

You can use Oracle 11.2.0.1 or 11.2.0.2 if you install the patch that fixes bug number 11678127. For basic information about patching an Oracle Database, see Patching Oracle Databases.

Use one of these solutions to install the required patch on the system where you install the Oracle R Enterprise server:

- Download the 11.2.0.3 patch set for Enterprise Edition from My Oracle Support.
- If you do not want to install the 11.2.0.3 patch set, install one of these patches:
  - For Oracle **11.2.0.1.0**: on Linux: Patch number 12598677 fixes bug number 11678127
  - For Oracle **11.2.0.2.0** on Linux: Patch number 12976544 fixes bug number 11678127
  - For Oracle **11.2.0.2** on Windows, Patch 11 fixes Bug Number 11678127

Patching Oracle Databases

Patches for Oracle products are downloaded from My Oracle Support (`http://support.oracle.com`). Access to My Oracle Support requires a CSI (Customer Support ID).

After you download patches, install them using OPatch, described in *Oracle Universal Installer and OPatch User's Guide 11g Release 2 (11.2) for Windows and UNIX*.

Before you apply a patch, review README.txt. The patch may require other patches. Also make sure that the latest version of OPatch is installed.

Install Oracle R Enterprise

After you verify that the Prerequisites are satisfied, install Oracle R Enterprise.

To download software for Oracle R Enterprise, go to Oracle R Enterprise Downloads at `http://www.oracle.com/technetwork/database/options/advanced-analytics/r-enterprise/ore-downloads-1502823.html`. In order to download files, you must accept the OTN License Agreement.

Oracle R Enterprise has two components:

- Oracle R Enterprise client installation: Installs required packages used with the R engine on user's desktop to enable transparent interaction with data resident in Oracle Database. The client runs on Windows or on Linux:

  - Install Client on Microsoft Windows
  - Install Client on Linux

  Install the client before you install the server.

- Oracle R Enterprise server installation: Installs required libraries and PL/SQL procedures to enable the Oracle Database to support an Oracle R Enterprise client. The server runs on Microsoft Windows, on an Oracle Linux or Red Hat Linux system, or on an Oracle Exadata machine running the Oracle Linux operating system:
  - Install Server on Microsoft Windows
  - Install Server on Linux

  Install the server after you install one or more clients.

After you install the client and the server, you can start Oracle R Enterprise:

- Start the Oracle R Enterprise Client on Microsoft Windows
- Start the Oracle R Enterprise Client on Linux

After you start Oracle R Enterprise, you can use the Oracle R Enterprise Samples to learn about using Oracle R Enterprise.

If startup fails or you encounter problems during installation, see Troubleshoot the Installation.

Install Client on Microsoft Windows

The Oracle R Enterprise client is supported on Microsoft Windows XP or later for 32-bit and 64-bit architectures. The client requires R 2.13.2.

To install the client, you must install two sets of packages:

- The supporting R packages DBI, ROracle, and png, all available from the Comprehensive R Archive Network (CRAN)
- The suite of Oracle R Enterprise packages: OREbase, OREstats, OREgraphics, OREeda, ORExml, and ORE

The downloads for Windows support both 32-bit and 64-bit architectures.

After you have installed R 2.13.2 as described in Install R on Windows, follow these steps to install the two sets of R packages for the Oracle R Enterprise Windows client:

1. The downloads for Oracle R Enterprise are available from `http://www.oracle.com/technetwork/database/options/advanced-analytics/r-enterprise/ore-downloads-1502823.html`. Download these files:
   - `ore-supporting-windows-1.1.zip`, the supporting R packages
   - `ore-client-windows-1.1.zip`, the Oracle R Enterprise packages.
2. Unzip `ore-supporting-windows-1.1.zip` to your local system. This creates a top level `ore-supporting-windows-1.1` directory whose subdirectory structure mimics a CRAN-like repository.
3. Unzip `ore-client-windows-1.1.zip` to your local system. This creates a top level `ore-windows-1.1` directory whose subdirectory structure mimics a CRAN-like repository.
4. Start either 32-bit or 64-bit R 2.13.2 from the **All Programs** group of the Windows **Start** menu. (Since the R Windows binary packages contain bundles for both 32-bit

and 64-bit architectures, either architecture of R 2.13.2 can be used during the installation.)

5. You can install both sets of R packages (CRAN and ORE) from either the R Console or from the R GUI.

   - To install both sets of packages from the R Console, type

     ```
     install.packages(c("ROracle", "png"),
                      repos = "file:///<DEP_PATH>/ore-supporting-windows-1.1",
                      type = "win.binary")
     install.packages("ORE", repos = "file:///<ORE_
     PATH>/ore-client-windows-1.1",
                      type = "win.binary")
     ```

     where <DEP_PATH> and <ORE_PATH> are the unzip directory locations of `ore-supporting-windows-1.1.zip` and `ore-windows-1.1.zip files` respectively. The `install.packages` function calls produce the message "successfully unpacked and MD5 sums checked" for each installed package.

   - To install both sets of packages from the R GUI, follow these steps:

     a. From the main menu, select **Packages** then **Install package(s) from local zip files**

     b. Navigate to

        ```
        <DEP_PATH>\ore-supporting-windows-1.1\bin\windows\contrib\2.13
        ```

        where <DEP_PATH> is the unzip directory you used for the `ore-supporting-windows-1.1.zip` file.

     c. Select `DBI_0.2-5.zip`, `ROracle_1.1.zip`, and `png_0.1-4.zip`.

     d. Click **Open**. Each package will produce the message "successfully unpacked and MD5 sums checked" message in the R Console.

     e. From the main menu, select **Packages** then **Install package(s) from local zip files**.

     f. Navigate to

        ```
        <ORE_PATH>\ore-client-windows-1.1\bin\windows\contrib\2.13
        ```

        where <ORE_PATH> is the unzip directory you used for the `ore-windows-1.1.zip` file.

     g. Select `OREbase_1.1.zip`, `OREstats_1.1.zip`, `OREgraphcs_1.1.zip`, `OREeda_1.1.zip`, `ORExml_1.1.zip`, and `ORE_1.1.zip`.

     h. Click **Open**. Each package will produce the message "successfully unpacked and MD5 sums checked" message in the R Console.

After the installation completes. you can Start the Oracle R Enterprise Client on Microsoft Windows after you have installed Oracle R Enterprise server on Linux.

Install Client on Linux

The Oracle R Enterprise client is supported on Oracle Linux or Red Hat Linux. The client requires R-2.13.2.

To install the client you must install two sets of packages:

- The supporting R packages DBI, ROracle, and png, all available from Comprehensive R Archive Network (CRAN). Depending on the platform, you download:

- The suite of Oracle R Enterprise packages: OREbase, OREstats, OREgraphics, OREeda, ORExml, and ORE

After you have installed R-2.13.2 on Linux as described in Install the Oracle R Distribution on Linux, follow these steps to install the two sets of R packages for the Oracle R Enterprise Linux 64-bit client:

1. Download Oracle Instant Client Basic Package for 64-bit from Linux from **Instant Client Downloads for Linux x86-64** (http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html).

   The Oracle Instant Client includes all files required to run OCI, OCCI, and JDBC-OCI applications. The ROracle R package is an OCI application.

   Either download the zip file `Instantclient-basic-linux.x64-11.2.0.3.0.zip` or install from `oracle-instantclient11.2-basic-11.2.0.3.0-1.x86_64.rpm`.

2. Add the path where you unzipped or installed the Oracle Instant Client libraries to your LD_LIBRARY_PATH.

3. Download the supporting R packages `ore-supporting-linux-x86-64-1.1.tar.gz` from http://www.oracle.com/technetwork/database/options/advanced-analytics/r-enterprise/ore-downloads-1502823.html.

4. Unzip and untar `ore-linux-x86-64-1.1.tar.gz` to your local system. This creates the directory `ore-supporting-linux-x86-64-1.1` containing these three files:

   ```
   DBI_0.2-5_R_x86_64-unknown-linux-gnu.tar.gz
   ROracle_1.1-1_R_x86_64-unknown-linux-gnu.tar.gz
   png_0.1-4_R_x86_64-unknown-linux-gnu.tar.gz
   ```

5. Download the Oracle R Enterprise client packages `ore-client-linux-x86-64-1.1.tar.gz` from http://www.oracle.com/technetwork/database/options/advanced-analytics/r-enterprise/ore-downloads-1502823.html.

6. Unzip and untar `ore-client-linux-x86-64-1.1.tar.gz` to your local system. This creates the directory `ore-client-linux-x86-64-1.1` containing these six files:

   ```
   ORE_1.1_R_x86_64-unknown-linux-gnu.tar.gz
   OREbase_1.1_R_x86_64-unknown-linux-gnu.tar.gz
   OREeda_1.1_R_x86_64-unknown-linux-gnu.tar.gz
   OREgraphics_1.1_R_x86_64-unknown-linux-gnu.tar.gz
   OREstats_1.1_R_x86_64-unknown-linux-gnu.tar.gz
   ORExml_1.1_R_x86_64-unknown-linux-gnu.tar.gz
   ```

7. Go to the directory `ore-supporting-linux-x86-64-1.1`. Type the following commands to install the supporting R packages:

   ```
   R CMD INSTALL DBI_0.2-5_R_x86_64-unknown-linux-gnu.tar.gz
   R CMD INSTALL ROracle_1.1-1_R_x86_64-unknown-linux-gnu.tar.gz
   R CMD INSTALL png_0.1-4_R_x86_64-unknown-linux-gnu.tar.gz
   ```

   The commands generate the following messages to confirm successful installation of the packages:

   ```
   * installing to library '<Your $R_HOME directory> /library'
   * installing *binary* package 'DBI' ...
   * DONE (DBI)
   * installing to library '<Your $R_HOME directory> /library'
   * installing *binary* package 'ROracle' ...
   * DONE (ROracle)
   ```

```
* installing to library '<Your $R_HOME directory> /library'
* installing *binary* package 'png' ...
* DONE (png)
```

8. Go to the directory `ore-linux-x86-64-1.1`. Type the following commands to install the ORE packages:

```
R CMD INSTALL ORE_1.1_R_x86_64-unknown-linux-gnu.tar.gz
R CMD INSTALL OREbase_1.1_R_x86_64-unknown-linux-gnu.tar.gz
R CMD INSTALL OREeda_1.1_R_x86_64-unknown-linux-gnu.tar.gz
R CMD INSTALL OREgraphics_1.1_R_x86_64-unknown-linux-gnu.tar.gz
R CMD INSTALL OREstats_1.1_R_x86_64-unknown-linux-gnu.tar.gz
R CMD INSTALL ORExml_1.1_R_x86_64-unknown-linux-gnu.tar.gz
```

   Each command generates messages like the following ones to confirm successful installation of the packages:

```
* installing to library '<Your $R_HOME directory> /library'
* installing *binary* package '<>' ...
* DONE (<>)
```

After the installation completes. you can Start the Oracle R Enterprise Client on Linux after you have installed Oracle R Enterprise server on Linux.

Install the Server

This section describes how to install the Oracle R Enterprise server on Windows and Linux.

Install the server as follows:

1. Make sure that the Prerequisites for Windows or Linux are satisfied.
2. Install the client *before* you install the server. You can either Install Client on Microsoft Windows or Install Client on Linux.
3. Make sure that Oracle Database is installed on the system where you plan to install the server. Make sure that any required patches are installed properly.
4. Make sure that all of the requirements in Before You Install the Server are satisfied.
5. Follow the directions in Install Server on Microsoft Windows or Install Server on Linux to install the server.

The install script creates Administrative Roles that you may need to grant to users who perform certain tasks.

After the install completes, you can Create Users.

After the server installation successfully completes, you can Start the Oracle R Enterprise Client on Microsoft Windows or Start the Oracle R Enterprise Client on Linux. Once the client has started, you can start Using Oracle R Enterprise.

Before You Install the Server

The install scripts for the Oracle R Enterprise server require that certain environment variables are properly set.

Important: Before you start installation, make sure that:

- You have DBA privileges (that is, you can run as oracle). On Linux, you must be a member of the DBA group and on Windows, you must be a member of the ORA_DBA group.
- You have write privileges at the operating system level to the file `$ORACLE_HOME/lib`.
- You can run R. This usually means that the R executable in your PATH environment variable.

  On Windows you may not have a PATH system variable defined; if the PATH variable does not exist, create it and set it to the directory where the executable for the R GUI resides. If you installed R in the default location, the GUI executable resides in `C:\Program Files\R\R-2.13.2\bin\i386`.

The following steps explain how the installation scripts work:

1. Before the Oracle R Enterprise server install starts, it checks for the presence of an R installation in the form of environment variable R_HOME. Make sure that R_HOME exists and is set properly.
2. Once the script verifies that R is known to be installed and its location known via PATH, the install checks for the presence of three libraries in `$R_HOME/lib/`. In particular, the script checks for
   - `$R_HOME/lib/libR.so`
   - `$R_HOME/lib/libRblas.so`
   - `$R_HOME/lib/libRlapack.so`
3. Next the script checks the location of the database installation by checking for the presence of environment variable ORACLE_HOME and ORACLE_SID.

   If ORACLE_HOME is set, then the install expects that the `$ORACLE_HOME/lib` directory is present.

   Before you start the script check that ORACLE_HOME and ORACLE_SID are present and properly set.
4. Next the script checks the Oracle database instance information The check includes looking for environment variable ORACLE_SID and then connecting to the instance by starting

   ```
   sqlplus  as sysdba
   ```

   Logging into the database as sysdba is critical for the install script to proceed.

   If `sqlplus` fails to connect to the database instance, the install process aborts.

   Before you start the installation script, check that you can connect to the database using this `sqlplus` command.
5. If Oracle R Enterprise has been installed on the database, that is if you installed release 1.0, then the installer expects to find a user name called RQSYS in `dba_users` table and the Oracle R Enterprise version number details in SYS.RQ_CONFIG. The installer uses this information subsequently to install the correct SQL packages.
6. The script prompts you to optionally enter the names of permanent and temporary table spaces for the RQSYS schema; the default schemas are SYSAUX and TEMP.

   At this point, the install script has determined it has found the prerequisites satisfactory and proceeds to do the actual installation.

7. The install script now attempts to copy libraries to `$ORACLE_HOME/lib`.

   If `$ORACLE_HOME/lib` is not writable then the installer errors out.

8. The install script now installs the RQSYS schema. Installing the schema requires logging into the database as SYSDBA.

9. Finally, the ORE packages are installed under the R installation.

Note that the installation creates Administrative Roles that are required for users to performs certain tasks.

You can create Oracle R Enterprise users as described in Create Users.

Install Server on Microsoft Windows

These directions describe how to install Oracle R Enterprise on Microsoft Windows XP or later for the 32-bit architecture only.

Review Before You Install the Server before you run the installation script. Make sure that all environment variables are properly set and that all required directories are present.

> **Note:** You can install the server on a 32-bit PC only. If you are not sure that your PC is 32-bit, see Check PC Architecture.

After the client installation completes, follow these steps to install the server:

1. Download correct file:
   - `ore-server-windows-x86-32-1.1.zip` (for 32-bit **only**)
2. Unzip the download.
3. Open a command window and navigate to the directory where you unzipped the download.
4. Execute `install.bat`.

   After installation completes, you can Create Users. It may be necessary to grant Administrative Roles.

5. Install the R supporting packages `ore-supporting-windows-1.1.zip`, if they are not installed already.

Check PC Architecture Follow these steps to find out if your computer is running a 32-bit or 64-bit version of Windows:

- If the operating system is Windows 7 or Windows Vista:
  1. Open **System**: Click **Start** button, right-click **Computer**, and then click **Properties**.
  2. **System** shows the system type.
- If the operating system is Windows XP:
  1. Click **Start**.
  2. Right-click **My Computer**, and then click **Properties**.
  3. If you don't see "x64 Edition" listed, then you're running the 32-bit version of Windows XP.

If "x64 Edition" is listed under System, you're running the 64-bit version of Windows XP.

Install Server on Linux

These directions describe how to install Oracle R Enterprise on Oracle Linux, Red Hat Linux, or Oracle Exadata Database Machine on the Linux x86-64 platform.

Review Before You Install the Server before you run the installation script. Make sure that all environment variables are properly set and that all required directories are present.

After the client installation completes, follow these steps to install the server:

1. Download `ore-server-linux-x86-64-1.1.tar.gz` from `http://www.oracle.com/technetwork/database/options/advanced-analytics/r-enterprise/ore-downloads-1502823.html`.
2. Unzip and untar `ore-server-linux-x86-64-1.1.tar.gz` into an empty directory on your local system. This creates a directory containing these library files, SQL scripts, and the `install` shell script `install.sh`.
3. Run `install.sh`. as Oracle User or any account that has DBA privileges to create objects in the SYS and RQSYS schemas.

   This script copies several libraries to `$ORACLE_HOME/lib`.

   The script executes `rqinst.sql`with SYSAUX and TEMP as the default and temporary tablespaces.

   The script creates all SQL objects required by Oracle R Enterprise in the RQSYS user schema. The RQSYS schema is created as a locked account with expired password and no connect privileges.

   See Create Users to create users. It may be necessary to grant Administrative Roles to users who perform certain tasks.
4. Install the R supporting packages `ore-supporting-linux-x86-64-1.1.zip`, if they are not installed already.

Administrative Roles

The installation creates an administrative role RQADMIN and a user role RQROLE. The roles are used as follows:

- Oracle R Enterprise users who are allowed to create R scripts that execute using the database embedded R engine must be granted the RQADMIN role.
- Oracle R Enterprise users who are allowed to execute R code via SQL queries must be granted the RQROLE role.

Create Users

`demo_user.sh` contains an example of how to create an Oracle R Enterprise user.

For each Oracle R Enterprise user, these steps are required to fully enable the user on the database (as demonstrated in the `demo_user.sh` script).

Start `SQL*Plus` as `sysdba` and the:

1. grant rqrole to `<Each ORE user>`
2. `grant execute on rqsys.rqGroupEvalImpl` for each user.
3. Create all the synonyms listed in `rquser.sql` for each user.

Grant Administrative Roles as necessary.

Start the Oracle R Enterprise Client on Microsoft Windows

After the server is installed, you can launch the client.

To launch the Oracle R Enterprise client in a running session of R 2.13.2, execute the following R code from the R Console. Before you execute the code, modify the connection information (user, sid, host, password, and port) for the database where the R Sever is installed:

```
# Load ORE packages and dependencies
# DBI, ROracle, OREbase, MASS, OREstats,
# OREgraphics, OREeda, ORExml, ORE
library(ORE)

# Connect to Oracle RDBMS
# Change the connection information below
ore.connect(user = "<USERNAME>",
            sid = "<SID>",
            host = "<HOST>",
            password = "<PASSWORD>",
            port = PORTNUMBER
            all = TRUE)
```

Also `ore.connect` can now use Oracle Wallet. In this case the connect is as follows: like this:

```
ore.connect(conn_string = "ore_wallet", all = TRUE)
```

`ore_wallet` is a connect string that has been registered with the Wallet.

For more information about database connectivity, see Tables in the Oracle Database.

As with all R commands, this code can be used during the initialization of an R session. For example, you can add the following lines of code, with modified connection information, to the `Rprofile.site` file located below the top-level R 2.13.2 installation directory `<R_HOME>\etc\Rprofile.site` (the default location where R is installed on Windows is `C:\Program Files\R\R-2.13.2)`:

```
library(ORE)
# Change the following connection settings:
user     <- "scott"
password <- "tiger"
sid      <- "orcl"
host     <- "myhost"
port     <- 1521
all      <- TRUE

cat("Connecting to ORE\n")
cat("    User:", user, "\n")
cat("    SID: ", sid, "\n")
cat("    Host:", host, "\n")
ore.connect(user, sid, host, password, port, all)
cat("Connected.\n")
```

For more information on the initialization sequence of R on startup, type `help(Startup)` in the R Console.

For more information about database connectivity, see Connect to an Oracle Database.

Start the Oracle R Enterprise Client on Linux

After the server is installed, you can launch the client.

Before you launch Oracle R Enterprise client, add these paths to the LD_LIBRARY_PATH environment variable:

1. The path where Oracle Instant Client libraries are installed. Otherwise loading of ROracle package will fail.
2. The path for the shared libraries `libR.so`, `libRblas.so`, and `libRlapack.so` from the installation of R-2.13.2

Start R-2.13.2 from your favorite Linux shell. Next use `ore.connect` to connect to the Oracle Database where the server resides.

Launch the Oracle R Enterprise client by executing, after modifying the connection information (user, sid, host, password, and port), the following R code from the R Console:

```
# Load ORE packages and dependencies
# DBI, ROracle, OREbase, MASS, OREstats,
# OREgraphics, OREeda, ORExml, ORE
library(ORE)

# Connect to Oracle RDBMS
# Change the connection information below
ore.connect(user = "<USERNAME>",
            sid = "<SID>",
            host = "<HOST>",
            password = "<PASSWORD>",
            port = PORTNUMBER
            all = TRUE)
```

Your Oracle Database Administrator can provide you with suitable values for USERNAME, SID, HOST, PASSWORD, and PORT. These values provide connection information for the database.

As with all R commands, this code can be used during the initialization of an R session. For example, you can add the following lines of code, with modified connection information, to the Rprofile.site file located below the top-level R-2.13.2 installation directory `<R_HOME>\etc\Rprofile.site`:

```
library(ORE)

# Change the following connection settings:
user     <- "scott"
password <- "tiger"
sid      <- "orcl"
host     <- "myhost"
port     <- 1521
all      <- TRUE

cat("Connecting to ORE\n")
cat("    User:", user, "\n")
cat("    SID: ", sid, "\n")
cat("    Host:", host, "\n")
ore.connect(user, sid, host, password, port, all)
```

For more information on the initialization sequence of R on startup, type `help(Startup)` in the R Console.

For more information about database connectivity, see Connect to an Oracle Database.

Connect to an Oracle Database

Oracle R Enterprise includes the following R functions that enable transparent access to Oracle Database tables and views:

- `ore.attach(USER, SID, host, password)` establishes a database connection using the schema or user name, the database SID, machine hostname, and password, and creates an environment that maps database table names to R objects (ore.frame) from the schema referenced in the database connection.

  At this time, views are not mapped.

  If you use the `all` parameter of `ore.connect` when you attach to a database, `ore.attach` is executed automatically.

- `ore.sync()` synchronizes with your schema (account) in the Oracle Database. `ore.connect` can perform this command.

  If you use the `all` parameter of `ore.connect` when you attach to a database, `ore.sync` is executed automatically.

- `ore.detach("SCHEMA_NAME")` detaches from the schema.
- `ore.ls()` lists all objects in the schema you are currently connected to.

Objects created by Oracle R Enterprise are identified with the `ore` prefix. Pick any object returned by `ore.ls()` and type either `class(OBJECTNAME)` or `class(OBJECTNAME$COLUMN_NAME)`.

For example,

```
R> class(NARROW)
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
```

The prefix `ore` is applied to the class names. This indicates that the object is an Oracle R Enterprise created object that holds metadata (instead of contents) of the corresponding object in Oracle Database.

Oracle R Enterprise Samples

Oracle R Enterprise is shipped with a collection of examples that illustrate how to use Oracle R Enterprise. The examples are shipped as demos included in the ORE package. For more information about the examples, see List of Examples on page 3-14.

Troubleshoot the Installation

The installation script creates a log file in each folder (client, data and server). Make sure that you look at each of the log files even if the installation reports success. Search the log file for ERROR.

If you cannot resolve the problems, request help from Oracle Support or from the Oracle R Enterprise discussion forum.

Upgrade Oracle R Enterprise

If you installed the first release of Oracle R Enterprise, you can upgrade to release 1.1 as follows:

- To upgrade the Client on Windows, re-install the packages. See Install Client on Microsoft Windows.

- To upgrade the Client on Linux, reinstall the packages. See Install Client on Linux.
- To upgrade the Server, reinstall the Server. See Install the Server.

Uninstall Oracle R Enterprise

Follow these steps to uninstall Oracle R Enterprise client:

1. To remove the Oracle R Enterprise packages, start R and type these commands:

```
remove.packages("ORE")
remove.packages("ORExml")
remove.packages("OREeda")
remove.packages("OREgraphics")
remove.packages("OREstats")
remove.packages("OREbase")
remove.packages("ROracle")
remove.packages("DBI")
```

2. Unset the environment variable R_PROFILE_USER.

To uninstall Oracle R Enterprise server, execute `uninstall.sh`. `uninstall.sh` removes libraries installed in `$ORACLE_HOME/lib` and removes all installed SQL objects.

3

Using Oracle R Enterprise

This chapter explains how to use Oracle R Enterprise to analyze data stored in tables or views in an Oracle Database. Before you analyze data in tables, you must connect to a database, as described in Tables in the Oracle Database.

This chapter discusses these topics:

- View Oracle R Enterprise Documentation
- Oracle R Enterprise Data
- Oracle R Enterprise Transparency Framework
- Oracle R Enterprise Database-Embedded R Engine
- Oracle R Enterprise Additional R Functions
- Oracle R Enterprise SQL Functions
- Oracle R Enterprise Examples

We assume familiarity with R in the remainder of this section.

These examples were all created using R Console, the default graphical user interface for Open Source R.

For information about `ore.connect`, `ore.attach`, `ore.sync`, and `ore.ls`, see Start the Oracle R Enterprise Client on Microsoft Windows, Start the Oracle R Enterprise Client on Linux, and Connect to an Oracle Database.

Oracle R Enterprise also includes the Oracle R Enterprise Statistical Functions, described in Chapter 4.

Tables in the Oracle Database

The first step to using Oracle R Enterprise to analyze data stored in database tables is to Start the Oracle R Enterprise Client on Microsoft Windowsen or Start the Oracle R Enterprise Client on Linux.

Objects created by Oracle R Enterprise are identified with the `ore` prefix. Pick any object returned by `ore.ls()` and type either `class(OBJECTNAME)` or `class(OBJECTNAME$COLUMN_NAME)`.

For example,

```
R> class(NARROW)
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
```

The prefix `ore` is applied to the class names. This indicates that the object is an Oracle R Enterprise created object that holds metadata (instead of contents) of the corresponding object in Oracle Database.

View Oracle R Enterprise Documentation

Use this command to view the Oracle R Enterprise documentation library:

```
R> OREShowDoc()
```

Oracle R Enterprise Data

When you install Oracle R Enterprise, two tables NARROW and ONTIME_S are loaded into the `rquser` schema:

```
R> ore.ls()
[1] "NARROW"   "ONTIME_S"
```

Oracle R Enterprise includes these functions:

- Load Data into the Database
- Drop a Database Table
- Pull a Database Table to an R Frame

Load Data into the Database

Follow these steps to load data from files on your system to the Oracle Database:

1. Load contents of the file to an R data frame using `read.table()` or `read.csv()` functions documented in the R manuals.
2. Then use `ore.create()` to load a data frame to a table:

   ```
   ore.create(data_frame, table="TABLE_NAME")
   ```

loads data_frame into the database table TABLE_NAME.

This example creates an R data frame `df` consisting of pairs of numbers and letters and then loads the data frame into the Oracle table DF_TABLE. The example shows that the data frame and the table have the same dimensions and the same first few elements, but different values for class. The class for DF_TABLE is ore.frame.

```
R> df <- data.frame(A=1:26, B=letters[1:26])
R> dim(df)
[1] 26  2
R> class(df)
[1] "data.frame"
R> head(df)
  A B
1 1 a
2 2 b
3 3 c
4 4 d
5 5 e
6 6 f
R> ore.create(df, table="DF_TABLE")
R> ore.ls()
[1] "DF_TABLE" "NARROW"   "ONTIME_S"
R> class(DF_TABLE)
```

```
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
R> dim(DF_TABLE)
[1] 26  2
R> head(DF_TABLE)
  A B
0 1 a
1 2 b
2 3 c
3 4 d
4 5 e
5 6 f
R>
```

Materialize R Data

`ore.push(data-frame)` stores an R object in the database as a temporary object, and returns a handle to that object. It converts data frame, matrix, and vector to a table, and list, model, and others to a serialized object.

This example pushes the numerical vector created by the R command `c(1,2,3,4,5)` to *v*, an Oracle R Enterprise object:

```
v <- ore.push(c(1,2,3,4,5))
R> class(v)
[1] "ore.numeric"
attr(,"package")
[1] "OREbase"
R> head(v)
[1] 1 2 3 4 5
```

Drop a Database Table

To drop a table in the database use

```
ore.drop(table="NAMEOFTABLE")
```

For example, this command drops DF_TABLE:

```
ore.drop(table="DF_TABLE")
```

Pull a Database Table to an R Frame

To pull the contents of an Oracle Database table or view into an in-memory R data frame use `ore.pull(OBJECT_NAME)` for the name of an object returned by `ore.ls()`.

> **Note:** You can pull a table or view to an R frame only if the.if the size of the data can fit into R's memory.

For example, use `ore.pull()` to create the data frame `df_narrow` from the table NARROW and then verify that `df_narrow` is a data frame:

```
R> df_narrow <- ore.pull(NARROW)
R> class(df_narrow)
[1] "data.frame"
```

Oracle R Enterprise Transparency Framework

The Oracle R Enterprise transparency framework allows R users to continue to use R syntax to work directly with database-resident objects without having to pull data from Oracle into R's memory on the user's desktop.

R language constructs and syntax are supported for objects mapped to Oracle Database objects. The following R data types have been overloaded so that they are mapped to database objects and hence enabled for in-database execution:

- Character, Integer, Numeric and Logical vectors
- Factors
- Data Frame
- Matrix is overloaded in two situations:
  - Linear algebra cross-products
  - Creating input matrices for advanced analytics

`class(object)` reports the data type of such mapped objects. For example,

```
R> class(NARROW$AGE)
[1] "ore.numeric"
attr(,"package")
[1] "OREbase"
```

The following operators and functions are supported. See R documentation for syntax and semantics of these operators and functions. Syntax and semantics for these items remain unchanged when used on a corresponding database-mapped data type (also known as an Oracle R Enterprise data type):

- **Mathematical transformations**: abs, sign, sqrt, ceiling, floor, trunc, cummax, cummin, cumprod, cumsum, log, log10, log2, log1p, acos, acosh, asin, asinh, atan, atanh, exp, expm1, cos, cosh, sin, sinh, tan, tanh, gamma, lgamma, digamma, trigamma, round, signif, pmin, pmax, zapsmall
- **Basic statistics**: mean, summary, min, max, sum, any, all, median, range, IQR, fivenum, mad, quantile, sd, var, table, rowSums, colSums, rowMeans, colMeans
- **Arithmetic operators**:+, -, *, /, ^, %%, %/%
- **Comparison operators**: ==, >, <, !=, <=, >=
- **Logical operators**: &, |, xor
- **Set operations**: unique, %in%
- **Assignment**: <-, =, ->
- **String operations**: tolower, toupper, casefold, toString, chatr, sub, gsub, substr, substring, paste, nchar
- **Combine Data Frame**: cbind, rbind, merge
- **Combine vectors**: append
- **Vector creation**: ifelse
- **Subset**: [, [[, $, head, tail, window, subset, Filter, na.omit, na.exclude, complete.cases
- **Data reshaping**: split, unlist
- **Data processing**: eval, with, within, transform

- **Apply variants**: tapply, aggregate, by
- **Regression**: `ore.lm()` - a variant of `lm()`
- **Special value checks**: is.na, is.finite, is.infinite, is.nan
- **Metadata functions**: attributes, nrow, NROW, ncol, NCOL, nlevels, names, row, col, dimnames, dim, length, row.names, col.names, levels, reorder
- **Graphics**: hist, boxplot, plot, smoothScatter
- **Garbage collection**: gc (removal of implicitly created temporary tables after errors and explicitly created temporary tables)
- **Conversion functions**: `as.ore.{character, factor, integer, logical, numeric, vector}`
- **Test functions**: `is.ore.{character, factor, integer, logical, numeric, vector}`
- **Save**: ore.push (table is automatically refreshed in R memory)

The following additional categories of functions exist to accomplish conversions to/from and checks on Oracle R Enterprise data types:

- **Hypothesis testing**: wilcox.test, ks.test, var.test, binom.test, chisq.test, t.test, bartlett.test
- **Bessel Functions**: Bessel(I,J,K,Y)
- **Gamma Functions**: gamma, lgamma, digamma, trigamma (part of mathematical functions group)
- **Various Distributions**: Density, cumulative distribution, and quantile functions for standard distributions
- **Matrix Operations**: %*% (matrix multiplication), crossprod (matrix cross-product), tcrossprod (matrix cross-product A times transpose of B)

The Oracle R Enterprise sample programs, described in Oracle R Enterprise Examples include several examples using each category of the functions listed above with Oracle R Enterprise data types.

Our design principle has been to support data pre-processing functionality extensively so all data preparation and analysis can take place directly in the database.If you need to use a statistical technique that is not available in Oracle R Enterprise, having used Oracle R Enterprise to preprocess and filter the data, a much smaller amount of data can be pulled into R.

If a specific function that you need is not in the list above, you must explicitly pull data from the database into the R engine memory using `ore.pull()` to create an in-memory R object first.

Using R with Oracle R Enterprise Data Types

The following examples illustrate using R with Oracle R Enterprise data types:

- **Simple column and row selection in R**:

```
# Push built-in R data set iris to database
ore.create(iris, table="IRIS")
head(iris)
iris_projected = IRIS[, c("PETAL_LENGTH", "SPECIES")]
R> head (iris_projected)
  PETAL_LENGTH SPECIES
```

```
0          1.4  setosa
1          1.4  setosa
2          1.3  setosa
3          1.5  setosa
4          1.4  setosa
5          1.7  setosa
```

- **Database JOIN using R**:

```
df1 <- data.frame(x1=1:5, y1=letters[1:5])
df2 <- data.frame(x2=5:1, y2=letters[11:15])
merge (df1, df2, by.x="x1", by.y="x2")
 x1 y1 y2
1  1  a  o
2  2  b  n
3  3  c  m
4  4  d  l
5  5  e  k
# Create database objects to correspond to in-memory R objects df1 and df2
ore.df1 <- ore.create(df1, table="DF1")
ore.df2 <- ore.create(df2, table="DF2")
# Compare results
R> merge (DF1, DF2, by.x="X1", by.y="X2")
  X1 Y1 Y2
0  1  a  o
1  2  b  n
2  3  c  m
3  4  d  l
4  5  e  k
```

- **Database aggregation using R**:

```
# Push built-in data set iris to database
ore.create(iris, table="IRIS")
aggdata <- aggregate(IRIS, by = list(IRIS$SPECIES), FUN = summary)
class(aggdata)
head(aggdata)
```

- **Data formatting and creating derived columns in R**

  Note that adding derived columns does not change the database table. See Derived Columns in Oracle R Enterprise.

```
diverted_fmt <- function (x) {
ifelse(x==0, 'Not Diverted',
ifelse(x==1, 'Diverted',''))
}
cancellationCode_fmt <- function(x) {
ifelse(x=='A', 'A CODE',
ifelse(x=='B', 'B CODE',
ifelse(x=='C', 'C CODE',
ifelse(x=='D', 'D CODE', 'NOT CANCELLED'))))
}
delayCategory_fmt <- function(x) {
ifelse(x>200,'LARGE',
ifelse(x>=30,'MEDIUM','SMALL'))
}
zscore <- function(x) {
(x-mean(x,na.rm=TRUE))/sd(x,na.rm=TRUE)
# ONTIME_S is a database table
ONTIME_S$DIVERTED <- diverted_fmt(DIVERTED)
ONTIME_S$CANCELLATIONCODE <- cancellationCode_fmt(CANCELLATIONCODE)
```

```
ONTIME_S$ARRDELAY <- delayCategory_fmt(ARRDELAY)
ONTIME_S$DEPDELAY <- delayCategory_fmt(DEPDELAY)
ONTIME_S$DISTANCE_ZSCORE <- zscore(DISTANCE)
```

Derived Columns in Oracle R Enterprise

When you add derived columns using Oracle R Enterprise, the derived do not affect the underlying table in the database. All that is generated is a SQL query that has the additional derived columns in the select list.

Oracle R Enterprise Database-Embedded R Engine

The embedded R engine in the Oracle Database allows R users to off load desktop calculations that may require either more resources such as those available to Oracle Database or database-driven data parallelism. The embedded R engine also executes R scripts embedded in SQL or PL/SQL programs (lights-out processing).

These examples illustrate using Oracle R Enterprise embedded R engine with standard R packages downloaded from CRAN:

- Build a Regression Model
- Perform R Computation in the Oracle Database
- Build a Series of Regression Models Using Data Parallelism

Build a Regression Model

This example illustrates **building a regression model** using a CRAN package. Prepare the data used for training in the database (filtering out observations that are not of interest, selecting attributes, imputing missing values, etc.) to create the table ONTIME_S_PREPROCESSED_SUBSET. Pull the prepared training set (which is usually small enough to fit in desktop R memory) into R client to execute the model build. The resulting model is then used to score vast numbers of rows, in parallel, in the Oracle Database.

Note that scoring is a trivially parallelizable operation because one row can be scored independent of and in parallel with another row. The model built on the desktop is shipped to the database to perform scoring on vast number of rows in the database.

The computations are divided into these steps:

1. **Build a model in the desktop**:

```
dat <- ore.pull(ONTIME_S_PREPROCESSED_SUBSET)
mod <- glm(ARRDELAY ~ DISTANCE + DEPDELAY, dat)
mod
summary(mod)
```

2. **Score in-parallel in the database using embedded R**:

```
prd <- predict(mod, newdata=ONTIME_S_FINAL_DATA_TO_BE_SCORED)
class(prd)
# Add predictions as a new column
res <- cbind(newdat, PRED = prd)
head(res)
```

Perform R Computation in the Oracle Database

This example illustrates **off-loading R computation** to execute in the embedded R engine. The R user simply includes his code within a closure (that is, `function() {}`) and invokes `ore.doEval()`. `ore.doEval()` schedules execution of the R code with the database-embedded R engine and returns the results back to the desktop for continued analysis:

```
mod <- ore.doEval(
   function() {
      library(biglm)
      dat <- ore.pull(ONTIME_S)
      mod <- biglm(ARRDELAY ~ DISTANCE + DEPDELAY, dat)
      mod
    });
print(mod)
mod=ore.pull(mod)
print(mod)
```

Build a Series of Regression Models Using Data Parallelism

This example illustrates **database-driven data parallelism** at work in building a series of regression models using a CRAN package. One model is built per unique value of a factor. The database orchestrates parallel and concurrent building of the models, one per factor and bringing the list of all models built to the user desktop for further analysis:

```
modList <- ore.groupApply(
  # Organize input to the R script - This is always an Oracle R Enterprise
  # data frame
   X=ONTIME_S,
  # Specify the grouping column. Here we request one model per unique value of
  # ONTIME_S$DEST
   INDEX=ONTIME_S$DEST,
  # Model building code goes inside the closure. Input and grouping
  # conditions can be referenced as parameters to the function
    function(x, param) {
     library(biglm)
     biglm(ARRDELAY ~ DISTANCE + DEPDELAY, x)
     });

   modList_local <- ore.pull(modList)
# Print the model for just one destination - BOSTON
summary(modList_local$BOS)
```

Oracle R Enterprise Additional R Functions

These functions are available to enable richer statistical analysis. See the Oracle R Enterprise Sample Library described in Oracle R Enterprise Examples for usage examples of each function. The functions all operate on an Oracle R Enterprise data frame:

- `ore.summary`: Enables powerful multiple aggregations of columns
- `ore.rank`: Enables flexible ranking across multiple columns
- `ore.sort`: Enables flexible sorting along one or more columns
- `ore.corr`: Enables correlation analysis of numeric columns

- `ore.crosstab`: Enables cross-column analysis
- `ore.freq`: Enables cross tabulation analysis of numeric columns

Oracle R Enterprise SQL Functions

Oracle R Enterprise users who are allowed to execute R code via SQL queries must be granted the RQROLE role.

To enable execution of an R script in the database (lights-out processing), Oracle R Enterprise provides variants of `ore.doEval()`, `ore.groupApply()`, and `ore.indexApply()` in SQL. (`ore.doEval()`, `ore.groupApply()`, and `ore.indexApply()` are described in Oracle R Enterprise Database-Embedded R Engine.)

The SQL functions are

- `rqTableEval()`
- `rqEval()`
- `rqRowEval()`

You can also code an rqGroupEval() Function function.

The `rq*:Eval()` functions have the same syntax:

```
rq*Eval(
     cursor(select * from table-1,
     cursor(select * from table-2,
     'select <column list> from table-3 t',
     <grouping col-name from table-1 or num_rows>,
      'function(x,param) {
        registered-R-code
     ] ')
```

where

- The first cursor is the input cursor: Input is passed as a whole table, group, or one row as a time to the R closure described in the fourth parameter.
- The second cursor is the parameters cursor: One value can be passed (that is, collection of the models to be implemented.
- The query specifies the output table definition; if this parameter is NULL, output is a BLOB; output can also be XML.
- `grouping col-name` is optional; it provides the name of the grouping column
- `num_rows` is optional; it provides then number of rows to provide to the functions at one time.
- `registeredR-code` is a registered version of the R function to execute. See Registering R Scripts for details.

The following examples illustrate using these functions:

- This example uses all rows from the table `fish` as input to the R function that takes no other parameters and produces `output` that contains all input data plus the ROWSUM of values.

  Note that both `input(x)` and parameters (`param`) to the R function is optional.

  ```
  select * from table(rqTableEval(
    cursor(select * from fish),
    NULL,
  ```

```
  'select t.*, 1 rowsum from fish t',
  'function(x, param) {
    dat <- data.frame(x, stringsAsFactors=F)
    cbind(dat, ROWSUM = apply(dat,1,sum))
  }'));
```

- This example illustrates passing n=1 (4th parameter) row at a time from the table `fish` to the R function. No parameters are required by the function. The function generates ROWSUM which is added as an extra column to `fish` in the output.

```
select * from table(rqRowEval(
  cursor(select * from fish),
  NULL,
 'select t.*, 1 rowsum from fish t',
  1,
  'function(x, param) {
    dat <- data.frame(x, stringsAsFactors=F)
    cbind(dat, ROWSUM = apply(dat,1,sum)+10)
  }'));
```

rqGroupEval() Function

There is no `rqGroupEval()` function as such. You must define a private version of `rqGroupEval()` based on the data and grouping column. This is the limitation of the table function infrastructure.

Here is an example based on the ONTIME_S sample data. The data cursor uses all data, but you could also define cursors that use some columns using PL/SQL records. Then you must define as many table functions as the number of grouping columns that you are interested in using for a particular data cursor:

```
CREATE PACKAGE ontimePkg AS
  TYPE cur IS REF CURSOR RETURN ontime_s%ROWTYPE;
END ontimePkg;
/

CREATE FUNCTION ontimeGroupEval(
  inp_cur  ontimePkg.cur,
  par_cur  SYS_REFCURSOR,
  out_qry  VARCHAR2,
  grp_col  VARCHAR2,
  exp_txt  CLOB)
RETURN SYS.AnyDataSet
PIPELINED PARALLEL_ENABLE (PARTITION inp_cur BY HASH (month))
CLUSTER inp_cur BY (month)
USING rqGroupEvalImpl;
/
```

At this time, only one grouping column is supported. If you have multiple columns you combine the columns into one column and use the new column as a grouping column. PARALLEL_ENABLE clause is optional but CLUSTER BY is not.

Registering R Scripts

For security purposes, you must first register the R script under some system unique name and use new name instead of the actual script in the call to `rq*Eval` table functions.

There are two administrative functions that create and drop scripts and a view that lists scripts:

- `sys.rqScriptCreate`
- `sys.rqScriptDrop`
- `sys.rq_scripts` view allows you to list and use scripts that were created

The scripts and the view require grants as described in Roles Required to Create and Use Scripts.

Here is an example of registering the scripts and using the registered scripts:

```
begin
  sys.rqScriptCreate('tmrqfun2',
'function() {
ID <- 1:10
res <- data.frame(ID = ID, RES = ID / 100)
res
}');
end;
/

select *
  from table(rqEval(
        NULL,
       'select 1 id, 1 res from dual',
       'tmrqfun2'));

begin
  sys.rqScriptDrop('tmrqfun2');
end;
```

Roles Required to Create and Use Scripts

To execute `sys.rqScriptCreate` and `sys.rqScriptDrop`, you must be granted the administrative role RQADMIN.

Select privilege for the `sys.rq_scripts` view is granted to RQROLE role.

The RQADMIN and RQROLE role are created when you Install Server on Linux

Oracle R Enterprise Examples

Oracle R Enterprise is shipped with a collection of examples that illustrate how to use Oracle R Enterprise. These examples are a collection of self-contained R scripts.

Most of the sample programs use the data frame `iris`, which is included in the R distribution. `iris` is loaded into a table as described in Load Data Frame to a Table.

The rest of this section describes two examples in detail and includes a list of all examples:

- Load Data Frame to a Table
- Handle NULL Values Using airquality
- List of Examples

Load Data Frame to a Table

Follow these steps to load an R data frame to a database table:

1. Starts R, load the ORE packages via library(ORE), and then connect to the database. The latter steps are automatic if Rprofile is in place.
2. Most of these examples use the R data frame `iris`. iris is shipped with R. Use the R command `class` to verify that `iris` is an R data frame:

   ```
   R> class(iris)
   [1] "data.frame"
   ```

   `iris` consist of measurements of parts of iris flowers. Use the R command `head` to see a small sample of the data in `iris`.

   ```
   R> head(iris)
         Sepal.Length Sepal.Width Petal.Length Petal.Width Species
   1          5.1         3.5          1.4         0.2  setosa
   2          4.9         3.0          1.4         0.2  setosa
   3          4.7         3.2          1.3         0.2  setosa
   4          4.6         3.1          1.5         0.2  setosa
   5          5.0         3.6          1.4         0.2  setosa
   6          5.4         3.9          1.7         0.4  setosa
   ```

3. Now load the data frame `iris` into the database that you are connected to.

   In these examples, the database table version of `iris` is named IRIS_TABLE. Drop IRIS_TABLE to make sure that no table of this name exists in the connected schema:

   ```
   ore.drop(table = "IRIS_TABLE")
   ```

   If IRIS_TABLE doesn't exist, you do not get a message.
4. Now create a database table with the data contained in `iris`:

   ```
   ore.create(iris, table = "IRIS_TABLE")
   ```

   Use `ore.ls()` to verify that the table was created:

   ```
   R> ore.ls()
   [1] "IRIS_TABLE" "NARROW"     "ONTIME_S"
   ```

5. IRIS_TABLE is a database-resident table with just metadata on the R side:

   ```
   R> class(IRIS_TABLE)
   [1] "ore.frame"
   attr(,"package")
   [1] "OREbase"
   ```

6. Use `head` to see the column names and the first few values in IRIS_TABLE:

   ```
   R> head(IRIS_TABLE)
     SEPAL_LENGTH SEPAL_WIDTH PETAL_LENGTH PETAL_WIDTH SPECIES
   0          5.1         3.5          1.4         0.2  setosa
   1          4.9         3.0          1.4         0.2  setosa
   2          4.7         3.2          1.3         0.2  setosa
   3          4.6         3.1          1.5         0.2  setosa
   4          5.0         3.6          1.4         0.2  setosa
   5          5.4         3.9          1.7         0.4  setosa
   ```

7. Use mode to see the data type of the column SPECIES.

   ```
   mode(IRIS_TABLE$SPECIES)
   [1] "raw"
   ```

8. Some algorithms only work if all of the data is numerical. Follow these steps to create IRIS_TABLE_N that does not contain SPECIES, the nonnumeric column:

```
IRIS_TABLE_N=IRIS_TABLE[,c("SEPAL_LENGTH", "SEPAL_WIDTH",  "PETAL_LENGTH",
"PETAL_WIDTH")]
```

You can use R functions to analyze the data in the table. Here are some simple examples taken from the example `basic.R`:

- Use `unique` to get a list of the unique entries in a column. This example find the unique SPECIES:

```
R> unique(IRIS_TABLE$SPECIES)
[1] setosa     versicolor virginica
Levels: setosa versicolor virginica
```

- Find the minimum, maximum, and mean of PETAL_LENGTH:

```
R> min(IRIS_TABLE$PETAL_LENGTH)
[1] 1
R> max(IRIS_TABLE$PETAL_LENGTH)
[1] 6.9
R> mean(IRIS_TABLE$PETAL_LENGTH)
[1] 3.758
```

If you need information about an R function, use the command `help(function-name)`.

Handle NULL Values Using airquality

The sample `null.R` is the only sample that does not use `iris` as data. `null.R` compares the handling of NULLs in SQL with the handling of NAs in R.

In R, NA is a logical constant of length 1 which contains a missing value indicator. In the database, null refers to the absence of a value in a column of a row. Nulls indicate missing, unknown, or inapplicable data.

Follow these steps to execute the sample:

1. This example uses the data frame `airquality`. Verify that the data set is a data frame and look at the few rows of the data frame:

```
R> class(airquality)
[1] "data.frame"
R> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

2. Load `airquality` into the database as "AIRQUALITY":

```
ore.drop(table = "AIRQUALITY")
ore.create(airquality, table = "AIRQUALITY")
```

   Use `ore.ls()` to verify that the table was created. If you wish, use `class(AIRQUALITY)` to verify that `AIRQUALITY` is a database-resident table with just metadata on the R side.

3. Examine how R handles NAs. Return all observations where ozone < 30:

```
R> nrow(airquality[airquality$Ozone < 30,])
[1] 92
```

Compare this with the results when NAs are explicitly excluded:

```
R> nrow(airquality[airquality$Ozone < 30 & !is.na(airquality$Ozone),])
[1] 55
```

4. The default behavior for SQL tables is to exclude NULLS in output:

```
nrow(AIRQUALITY[AIRQUALITY$OZONE < 30,])
[1] 55
```

To handle NULLs the same way that R handles NA, request the behavior explicitly:

```
options(ore.na.extract = TRUE)
nrow(AIRQUALITY[AIRQUALITY$OZONE < 30,])
[1] 92
```

List of Examples

These scripts have been added as demos to the ORE package.

To access a complete listing of them type

```
R> demo(package = "ORE")
```

To run one of these scripts, specify the name of the demo in a `demo` function call. For example, to run `aggregate.R`, type

```
R> demo("aggregate", package = "ORE")
```

These examples are shipped with Oracle R Enterprise:

```
table_apply.R   Execute R code on all rows of a table passed in at once
aggregate.R     Demonstrates aggregations. See also summary.R
analysis.R      Demonstrates basic analysis and data processing operations
basic.R         Demonstrates basic connectivity to database
binning.R       Demonstrates binning in R
columnfns.R     Demonstrates use of column functions
corr.R          Correlation matrix (Pearson's, Spearman/Kendalls)
crosstab.R      Frequency cross-tabulations. Also see freq.R
derived.R       Handling derived columns
distributions.R Distribution, Density, and Quantile Functions
doEval.R        Demonstrates support for database-enabled parallel simulations
freqanalysis.R  Frequency cross-tabulations. Also see crosstab.R
graphics.R      Demonstrates visual analysis (boxplot, histogram)
group_apply.R   Execute R code for different sets of rows, one set per group
hypothesis.R    Hypothesis Testing Functions(binomial, chi square, T test, etc.)
matrix.R        Matrix operations
nulls.R         Demonstrates handling of nulls in SQL vs. NAs in R
push_pull.R     Demonstrates collaborative processing between database and client
rank.R          Ranking of observations (ranking, handling ties, etc.)
reg.R           Multivariate Regression
row_apply.R     Execute R code on each row
sql_like.R      Demonstrates how R commands map to SQL operations
stepwise.R      Stepwise Multivariate Regression
summary.R       Demonstrates summary functionality
```

4

Oracle R Enterprise Statistical Functions

This chapter describes Oracle R Enterprise functions that perform most common or base statistical procedures. These functions are designed to help users who are converting from commercially available products to Oracle R Enterprise.

Oracle R Enterprise provides these collections of functions:

- ore.corr
- ore.crosstab
- ore.extend
- ore.freq
- ore.rank
- ore.sort
- ore.summary
- ore.univariate

The use of the functions is illustrated with examples. Most of the examples use the same data, described in Data for Examples.

Data for Examples

Most of the examples use the table NARROW, which is installed in your database when you install with Oracle R Enterprise.

NARROW is an ore.frame with 9 columns:

```
R> class(NARROW)
[1] "ore.frame"
attr(,"package")
[1] "OREbase"
R> names(NARROW)
[1] "ID"             "GENDER"         "AGE"            "MARITAL_STATUS"
[5] "COUNTRY"        "EDUCATION"      "OCCUPATION"     "YRS_RESIDENCE"
[9] "CLASS"
```

ore.corr

`ore.corr` performs correlation analysis across numeric columns in an ore.frame.

`ore.corr` supports partial correlations with a control column.

`ore.corr` enables aggregations prior to correlations.

`ore.corr` allows post-processing of results and integration into an R code flow.

The output of `ore.corr` can be made to conform to output of the R `cor()` function; this allows the output of ore.corr to be post-processed by any R function or graphics.

See ore.corr Parameters for syntax and output and ore.corr Examples for examples.

ore.corr Parameters

`ore.corr` has these parameters:

- **data**: The data for which to compute correlation coefficients as an ore.frame.
- **var**: The numeric column(s) of **data** for which to build correlation matrix
- **group.by**: Indicates the correlation matrices to calculate; `ore.corr` calculates as many correlation matrices as unique values in **group.by** columns; default value is NULL
- **weight**: A column of the data whose numeric values provide a multiplicative factor for **var** columns; default value is NULL
- **partial**: columns of data to use as control variables for partial correlation; default value is NULL
- **stats**: The method of calculating correlations; one of pearson (default), spearman, kendall

`ore.corr` returns an ore.frame as output in all cases except when **group.by** is used. If **group.by** is used, returns an Oracle R Enterprise list object.

To convert the output of `ore.corr` into R `cor()`-compatible output format, use:

```
OREeda:::ore.corr.as.matrix()
```

ore.corr Examples

These examples show how to use `ore.corr`:

- Basic Correlation Calculations
- Partial Correlation
- Create Several Correlation Matrices
- Visualization of Correlations

These examples use the NARROW data set; for more information, see Data for Examples.

Basic Correlation Calculations

Before you can use `ore.corr`, you must project out all non-numerical values:

```
R> names(NARROW)
 [1] "ID"             "GENDER"     "AGE"    "MARITAL_STATUS"
"COUNTRY"        "EDUCATION"      "OCCUPATION"
 [8] "YRS_RESIDENCE"  "CLASS"          "AGEBINS"
R> NARROW=NARROW[,c(3,8,9)]
```

Now calculate correlation in several ways:

```
R> x=ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS')
#Calculate using Spearman
R> x=ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='spearman')
```

```
# Calculate using Kendall
R> x=ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='kendall')
# Convert output so that it is compatible with output of R cor
R> cor_compatible_matrix = OREeda:::.ore.corr.as.matrix(x)
R> class(cor_compatible_matrix)
[1] "matrix"
```

Partial Correlation

Use the version of NARROW with non-numeric values that was created in Basic Correlation Calculations.

Calculate partial correlation using Spearman's methods:

```
R> x=ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='spearman',
partial='GENDER')
```

Create Several Correlation Matrices

Use the version of NARROW with non-numeric values that was created in Basic Correlation Calculations.

Create several correlation matrices and then convert the output so that it is compatible with R output:

```
R> x=ore.corr(NARROW,var='AGE,YRS_RESIDENCE,CLASS', stats='pearson',
partial='GENDER', group.by='COUNTRY')
R> class(x)
[1] "list"
R> cor_compatible_matrix = OREeda:::.ore.corr.as.matrix(x[[1]])
```

Visualization of Correlations

If you calculate several matrices, you can use R packages to visualize them.

ore.crosstab

Cross tabulation is a statistical technique that finds an interdependent relationship between two tables of values.

`ore.crosstab` enables cross column analysis of an ore.frame. This function is a sophisticated variant of the R `table()` function.

`ore.crosstab` must be performed before frequency analysis is done using ore.freq.

You can extend the cross tab calculation with various sums as described in ore.extend.

`ore.crosstab` is written in R. The function is mapped to SQL that gets executed at the database server.

See ore.crosstab Parameters for syntax and output and ore.crosstab Examples for examples.

You can use ore.extend to augment crass tabulation.

ore.crosstab Parameters

`ore.crosstab` has these parameters:

- **expr**: the cross tabulation definition

  ```
  [COLUMN_SPEC] ~ COLUMN_SPEC [*<WEIGHTING COLUMN>] [/<GROUPING COLUMN>]
  ```

```
 [^<STRATIFICATION COLUMN>] [|ORDER_SPECIFICATION]
COLUMN_SPEC is <column-name>[+COLUMN_SET][+COLUMN_RANGE]
COLUMN_SET is <column_name>[+COLUMN_SET]
COLUMN_RANGE is <FROM COLUMN>-<TO COLUMN
```

where

```
COLUMN_SPEC is <column>[+COLUMN_SET][+COLUMN_RANGE]
COLUMN_SET is <column>[+COLUMN_SET]
COLUMN_RANGE is (<from column>-<to column>)
ORDER_SPECIFICATION is one of [-]NAME, [-]DATA, [-]FREQ, or INTERNAL
```

The stratification column is used to cluster, or group, data. When used, the values contribute to the ORE$STRATA column of the resulting cross-tabulated table.

- **data**: the ore.frame containing the data to cross tabulate
- **grouping column**: as many cross tabulations as unique values in grouping columns; default value is NULL
- **order**: defines optional sorting of output data. Specify [-]NAME to sort by tabulation columns, [-]FREQ to sort by frequency counts in table. Unspecified order is the most efficient. The optional '-' reverses the order direction.
- **weights**: column of the data that indicates the frequency of the corresponding row; default value is NULL
- **partial**: columns of data to use as control variables for partial correlation; default value is NULL

`ore.crosstab` returns an ore.frame as output in all cases except when multiple tables are created. If multiple tables are created, ore.crosstab returns an Oracle R Enterprise list object.

ore.crosstab Examples

These examples illustrate use of `ore.crosstab`:

- Single-Column Frequency Table
- Analyze Two Columns
- Weighting Rows
- Order Rows in the Cross Tabulated Table
- Analyze Three or More Columns
- Specify a Range of Columns
- Produce One Cross Table for Each Value of Another Column
- Augment Cross Tabulation with Stratification
- Custom Binning Followed by Cross Tabulation
- ore.extend

These examples use the NARROW data set; for more information, see Data for Examples.

Single-Column Frequency Table

The most basic use case is to create a single column frequency table. The following command filters NARROW) grouping by GENDER:

```
R> ct = ore.crosstab(AGE, data=NARROW)
R> ct
```

Analyze Two Columns

This command analyses AGE by GENDER aid AGE by CLASS:

```
R> ct = ore.crosstab(AGE~GENDER+CLASS, data=NARROW)
R> head(ct)
```

Weighting Rows

To weight rows, include count based on another column; this example weights values in AGE and GENDER using values in YRS_RESIDENCE:

```
R> ct = ore.crosstab(AGE~GENDER*YRS_RESIDENCE, data=NARROW)
R> head(ct)
```

Order Rows in the Cross Tabulated Table

There are several possibilities:

- Default or NAME- Order by the columns being analyzed
- FREQ - Order by frequency counts
- -NAME or -FREQ does reverse ordering
- INTERNAL - Bypass ordering

Here are two examples:

```
R> ct = ore.crosstab(AGE~GENDER|FREQ, data=NARROW)
R> head(ct)
  AGE GENDER ORE$FREQ ORE$STRATA ORE$GROUP

R> ct = ore.crosstab(AGE~GENDER|-FREQ, data=NARROW)
R> head(ct)
```

Analyze Three or More Columns

This is similar to what SQL GROUPING SETs accomplish:

```
 ct = ore.crosstab(AGE+COUNTRY~GENDER, NARROW)
```

Specify a Range of Columns

You can specify a range of columns instead of having to type all the column names, as illustrated in this example:

```
R> names(NARROW)
[1] "ID"             "GENDER"         "AGE"            "MARITAL_STATUS"
[5] "COUNTRY"        "EDUCATION"      "OCCUPATION"     "YRS_RESIDENCE"
[9] "CLASS"
```

Since AGE, MARITAL_STATUS and COUNTRY are successive columns, you can simply use

```
ct = ore.crosstab(AGE-COUNTRY~GENDER, NARROW)
```

An equivalent version is

```
ct = ore.crosstab(AGE+MARITAL_STATUS+COUNTRY~GENDER, NARROW)
```

Produce One Cross Table for Each Value of Another Column

This command produces one cross table (AGE, GENDER) for *each* unique value of another column COUNTRY:

```
R> ct=ore.crosstab(~AGE/COUNTRY, data=NARROW)
R> head(ct)
```

You can extend this to more than one column. For example, this command produces one (AGE,EDUCATION) table for each unique combination of (COUNTRY, GENDER):

```
R> ct = ore.crosstab(AGE~EDUCATION/COUNTRY+GENDER, data=NARROW)
```

Augment Cross Tabulation with Stratification

All of the above cross tabs can be augmented with stratification. For example,

```
R> ct = ore.crosstab(AGE~GENDER^CLASS, data=NARROW)
R> head(ct)
```

The command in this example is the same as

```
ct = ore.crosstab(~GENDER, NARROW, strata="CLASS")
```

Custom Binning Followed by Cross Tabulation

First bin AGE, the calculate cross tabulation for GENDER and the bins:

```
R> NARROW$AGEBINS=ifelse(NARROW$AGE<20, 1, ifelse(NARROW$AGE<30,2,
ifelse(NARROW$AGE<40,3,4)))
R> ore.crosstab(GENDER~AGEBINS, NARROW)
```

ore.extend

The cross tabulation produced using ore.crosstab can be further augmented with these three basic statistics:

- Row and Column Sums

  ```
  crosstab = ore.extend.sum(crosstab)
  ```

- Cumulative sums for each cell of the table

  ```
  crosstab = ore.extend.cumsum(crosstab)
  ```

- Total for the entire table

  ```
  crosstab = ore.extend.total(crosstab)
  ```

The following example illustrates `ore.extend`:

```
R> ct = ore.crosstab(~GENDER, NARROW)
R> ct = ore.extend.sum(ct)
R> ct
  GENDER ORE$FREQ ORE$STRATA ORE$GROUP ORE$SUM$GENDER
0      F      421          1         1            421
1      M      880          1         1            880
```

ore.freq

`ore.crosstab` must be performed before frequency analysis is done using ore.freq.

`ore.freq` analyses the output of `ore.crosstab` and automatically determines the techniques that are relevant to an `ore.crosstab` result. The techniques depend on the kind of cross tables:

- 1-way cross tables

  Goodness-of-fit tests for equal proportions or specified null proportions, confidence limits and tests for equivalence.

- 2-way cross tables
  - Various statistics that describe relationships between columns in the cross tabulation
  - Chi-square tests, Cochran-Mantel-Haenzsel statistics, measures of association, strength of association, risk differences, odds ratio and relative risk for 2x2 tables, tests for trend
- N-way cross tables
  - N 2-way cross tables
  - Statistics across and within strata

`ore.freq` uses Oracle Database SQL functions when available.

See ore.freq Parameters for syntax and output and ore.freq Examples for examples.

ore.freq Parameters

`ore.freq` supports these parameters:

- **crosstab**: ore.frame output from `ore.crosstab()`
- **stats**: List of statistics required; these statistics are supported:
  - Chi Square: AJCHI, LRCHI, MHCHI, PCHISQ
  - Kappa: KAPPA, WTKAP
  - Lambda: LAMCR, LAMRC, LAMDAS
  - Correlation: KENTB,PCORR, SCORR
  - Stuart's Tau, Somers: D | C, STUTC, SMDCR,SMDRC
  - Fisher's, Cochran's Q, FISHER, COCHQ
  - Odds Ratio: OR, MHOR, LGOR
  - Relative Risk: RR,MHRR,ALRR
  - Others: MCNEM, PHI, CRAMV, CONTGY, TSYM, TREND, GAMMA

  The default value is NULL.
- **Params**: Control parameters specific to the statistical function specified in **stats**:
  - SCORE: TABLE | RANK | RIDIT | MODRIDIT
  - ALPHA: *number*
  - WEIGHTS: *number*

  The default value is NULL.

- **skip.missing**: Either TRUE or FALSE; skip cells with missing values in the cross table; default value is FALSE
- **skip.failed**: Either TRUE or FALSE; if a statistical test required fails on the cross table because it is found to be in-applicable to the table then return immediately; ; default value is FALSE

`ore.freq` returns an ore.frame in all cases.

ore.freq Examples

These examples use the NARROW data set; for more information, see Data for Examples.

Before you use `ore.freq`, you must calculate cross tabs.

For example:

```
R> ct = ore.crosstab(~GENDER, NARROW)
R> ore.freq(ct)
  METHOD     FREQ DF PVALUE      DESCR GROUP
0   PCHI 161.9377  1      0 Chi-Square     1
```

ore.rank

`ore.rank` analyzes distribution of values in numeric columns of an ore.frame.

`ore.rank` supports useful functionality, including:

- Ranking within groups
- Partitioning rows into groups based on rank tiles
- Calculation of cumulative percentages and percentiles
- Treatment of ties
- Calculation of normal scores from ranks

`ore.rank` syntax is simpler that the corresponding SQL queries.

See ore.rank Parameters for syntax and ore.rank Examples for examples.

ore.rank Parameters

`ore.rank` supports these parameters:

- **data**: The ore.frame containing the data to rank
- **var**: numeric columns in **data** to rank
- **desc**: If `desc=TRUE`, rank in descending order; otherwise, rank in ascending order. (The default is to rank in ascending order.)
- **groups**: Partition rows into #groups based on ranks. For percentiles, `groups=100`, For deciles, `groups=10`, For quartiles, `groups=4`.

  The default value is NULL.
- **group.by**: Rank each group identified by `group.by` columns separately

  The default value is NULL.
- **ties**: Specify how to treat ties. Assign the largest of, or smallest of, or mean of corresponding ranks to tied values

The default value is NULL.

- **fraction**: The rank of a column value divided by the number of non-missing column values; the default value is FALSE.

  Use with `nplus1` to estimate the cumulative distribution function

- nplus1: fraction plus 1, that is, 1 plus the rank of a column value divided by the number of non-missing column values; the default value is FALSE.

  Use with `fraction` to estimate the cumulative distribution function

- **percent**: **fraction** converted to a percent value, that is f**raction** * 100.

`ore.rank` returns anore.frame in all instances.

You can use these R scoring methods with `ore.rank`:

- To compute exponential scores from ranks, use `savage`.
- To compute normal scores, use one of `blom, tukey`, or `vw`(van der Waerden).

ore.rank Examples

These examples illustrate using `ore.rank`:

- Rank Two Columns
- Handle Ties
- Rank Within Groups
- Partition into Deciles
- Estimate Cumulative Distribution Function

These examples use the NARROW data set; for more information, see Data for Examples.

Rank Two Columns

This example ranks the two columns AGE and CLASS and reports the results as derived columns; values are ranked in the default order (ascending):

```
R> x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass')
```

Handle Ties

This example ranks the two columns AGE and CLASS. If there is a tie, the smallest value is signed to all tied values:

```
R> x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass', ties='low')
```

Rank Within Groups

This example ranks the two columns AGE and CLASS and ranks the values according to COUNTRY:

```
R> x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass',
group.by='COUNTRY')
```

Partition into Deciles

This example ranks the two columns AGE and CLASS and partitions the columns into deciles (10 partitions):

```
R> x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass',groups=10)
```

To partition the columns into a different number of partitions, change the value of `groups`. For example, `groups=4` partitions into quartiles.

Estimate Cumulative Distribution Function

This example ranks the two columns AGE and CLASS and estimates the cumulative distribution function for both columns:

```
R> x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass',nplus1=TRUE)
```

Score Ranks

This example ranks the two columns AGE and CLASS and scores the ranks in tow different ways. The first command all partitions the columns into percentiles (100 groups). `savage` calculates exponential scores and `blom` calculates normal scores:

```
R> x <- ore.rank(data=NARROW, var='AGE=RankOfAge,
         CLASS=RankOfClass',score='savage', groups=100, group.by='COUNTRY')
R> x <- ore.rank(data=NARROW, var='AGE=RankOfAge, CLASS=RankOfClass',score='blom')
```

ore.sort

`ore.sort` enables flexible sorting of a data frame along one or more columns specified in a **by** clause.

`ore.sort` can be used with other data pre-processing functions. The results of sorting can provide input to R visualization.

`ore.sort` sorting takes places in the Oracle database. `ore.sort` supports the database `nls.sort` option.

See ore.sort Parameters for syntax and ore.sort Examplesfor examples.

ore.sort Parameters

`ore.sort` supports these parameters:

- **data**: ore.frame containing the data to be sorted; **required**
- **by**: the column(s) in **data** by which to sort the data; **required**
- **stable**: Relative order is maintained within sorted group (TRUE or FALSE); default value is FALSE
- **reverse**: Optional reversal of collation order for character variables (TRUE or FALSE); default value is FALSE
- **unique.keys**: Optional deletion of observations with duplicate values in the columns being sorted, TRUE or FALSE; default value is FALSE
- **unique.data**: Optional deletion of observations duplicate values in all columns, TRUE or FALSE; default value is FALSE

data and **by** are required parameters; all other parameters are optional

`ore.sort` returns an ore.frame.

ore.sort Examples

The following examples illustrate using `ore.sort`:

- Sort Columns in Descending Order
- Sort Different Columns in Different Orders
- Sort and Return One Row per Unique Value
- Remove Duplicate Columns
- Remove Duplicate Columns and Return One Row per Unique Value
- Preserve Relative Order in Output

Most of these examples use the NARROW data set; for more information, see Data for Examples. There are also Examples Using ONTIME_S.

Sort Columns in Descending Order

Sort the columns AGE and GENDER in descending order:

```
R> x=ore.sort(data=NARROW,by='AGE,GENDER', reverse=TRUE)
```

Sort Different Columns in Different Orders

Sort AGE in descending order and GENDER in ascending order:

```
R> x=ore.sort(data=NARROW,by='-AGE,GENDER')
```

Sort and Return One Row per Unique Value

Sort by AGE and keep one row per unique value of AGE:

```
R> x=ore.sort(data=NARROW,by='AGE', unique.key=TRUE)
```

Remove Duplicate Columns

Sort by AGE and remove duplicate rows:

```
R> x=ore.sort(data=NARROW,by='AGE', unique.data=TRUE)
```

Remove Duplicate Columns and Return One Row per Unique Value

Sort by AGE. Also remove duplicate rows, and return one row per unique value of AGE:

```
R> x=ore.sort(data=NARROW,by='AGE', unique.data=TRUE,unique.key = TRUE)
```

Preserve Relative Order in Output

Maintain the relative order in the sorted output:

```
R> x=ore.sort(data=NARROW,by='AGE', stable=TRUE)
```

Examples Using ONTIME_S

These examples use the ONTIME_S airline data that is installed when you install Oracle R Enterprise:

- Sort ONTIME_S by airline name in descending order and departure delay in ascending order:

```
R> sortedOnTime1 <- ore.sort(data=ONTIME_S, by='-UNIQUECARRIER,DEPDELAY')
```

- Sort ONTIME_S by airline name and departure delay and select one of each combination (that is, return a unique key):

```
R> sortedOnTime1 <- ore.sort(data=ONTIME_S, by='-UNIQUECARRIER,DEPDELAY',
unique.key=TRUE)
```

ore.summary

`ore.summary` calculates descriptive statistics and supports extensive analysis of columns in an ore.frame, along with flexible row aggregations.

`ore.summary` supports these statistics:

- Mean, min., max, mode, number of missing values, sum, weighted sum
- Corrected and uncorrected sum of squares, range of values, stddev, stderr, variance
- t-test for testing the hypothesis that the population mean is 0
- Kurtosis, skew, Coefficient of Variation
- Quantiles: p1, p5, p10, p25, p50, p75, p90, p95, p99, qrange
- 1-sided and 2-sided Confidence Limits for the mean: clm, rclm, lclm
- extreme value tagging

`ore.summary` provides a relatively simple syntax compared with SQL queries for the same results.

See ore.summary Parameters for syntax and ore.summary Examples for examples.

ore.summary Parameters

`ore.summary` supports these parameters:

- **data**: the data to aggregate as an ore.frame
- **class**: column(s) of **data** to aggregate (that is, SQL GROUP BY); default value is NULL
- **var**: column(s) of **data** on which to apply statistics functions (SQL SELECT list)
- **stats**: list of statistics functions to be applied on **var** columns

  mean, min, max, cnt, n, nmiss, css, uss, cv, sum, sumwgt, range, stddev, stderr, var, t, kurt, skew,p1, p5, p10, p25, p50, p75, p90, p95, p99, qrange, lclm, rclm, clm, mode that can be requested on **var** columns.

  The default value are n, mean, min, max.
- **weight**: A column of **data** whose numeric values provide a multiplicative factor for **var** columns
- **maxid**, **minid**: for each group optionally list maximum or minimum value from other columns in **data**; default value is NULL
- **ways**: restrict output to only certain grouping levels of the **class** variables; default value is NULL

- **group.by**: column(s) of data to stratify summary results across; default value is NULL
- **order**: defines optional sorting of output data. Specify [-]NAME to sort by tabulation columns, [-]FREQ to sort by frequency counts in table. Unspecified order is the most efficient. The optional '-' reverses the order direction
- **_FREQ**: frequency, number of observations in a group
- **_TYPE**: identifies the grouping, binary code based
- **_LEVEL**: identifies number of variables used in grouping

`ore.summary` returns an ore.frame as output in all cases except when a **group.by** clause is used. If a **group.by** clause is used, `ore.summary` returns a list of ore.frames, one frame per stratum.

ore.summary Examples

These examples illustrate the use of `ore.summary`:

- Calculate Default Statistics
- Skew and t Test
- Weighted Sum
- Two Separate Group By Columns

These examples use the NARROW data set; for more information, see Data for Examples.

Calculate Default Statistics

This example calculates mean, min, max for columns AGE and CLASS and rolls up (aggregates) GENDER:

```
R> ore.summary(NARROW, class='GENDER', var ='AGE,CLASS'. order='freq')
```

Skew and t Test

This example calculates skew for skew of AGE as column A and the t-test for CLASS as column B:

```
R> ore.summary(NARROW, class='GENDER', var='AGE,CLASS',  stats='skew(AGE)=A,
probt(CLASS)=B')
```

Weighted Sum

This example calculates weighted sum for AGE aggregated by GENDER with YRS_ RESIDENCE as weights; in other words, it calculates `sum(var*weight)`:

```
R> ore.summary(NARROW, class='GENDER', var='AGE', stat='sum=X', weight='YRS_
RESIDENCE')
```

Two Separate Group By Columns

Group CLASS by GENDER and MARITAL_STATUS:

```
r> ore.summary(NARROW, class='GENDER, MARITAL_STATUS', var='CLASS', ways=1)
```

All Possible Group By

This example groups CLASS in all possible ways by GENDER and MARTIAL_ STATUS:

```
R> ore.summary(NARROW, class='GENDER, MARITAL_STATUS', var='CLASS', ways='nway')
```

ore.univariate

`ore.univariate` provides distribution analysis of numeric variables in an ore.frame.

`ore.univariate` provides these statistics:

- All statistics reported by ore.summary
- Signed rank test, Student's t-test
- Extreme values reporting

See ore.univariate Parameters for syntax and ore.univariate Examples for examples.

ore.univariate Parameters

`ore.univariate` supports these parameters:

- **data**: The data to aggregate as an ore.frame
- **var**: Numerical column(s) of **data** to analyze
- **weight**: A column of the *data* whose numeric values provide a multiplicative factor for **var** columns; the default value is NULL
- **stats**: Optional specification of a subset of statistics to calculate and display:
  - moments: n, sumwgt, mean, sum, stddev, var, skew, kurt., uss.css.cv, stderr
  - measures: mean, stddev, median, var, mode, range, iqr
  - quantiles: p100, p99, p95, p90, p75, p50, p25, p10, p5, p1, p0
  - location: studentt, studentp, signt, signp, srankt, srankp
  - normality
  - loccount: loc<,loc>,loc!
  - extremes

  The default value is NULL.

`ore.univariate` returns anore.frame as output in all cases.

ore.univariate Examples

These examples illustrate the use of `ore.univariate`:

- Default Univariate Statistics
- Location Statistics
- Complete Quantile Statistics

These examples use the NARROW data set; for more information, see Data for Examples.

Default Univariate Statistics

This example calculates the default univariate statistics for AGE, YRS_RESIDENCE, and CLASS:

```
R> ore.univariate(NARROW, var="AGE,YRS_RESIDENCE,CLASS")
```

Location Statistics

This example calculates location statistics for YRS_RESIDENCE:

```
R> ore.univariate(NARROW, var="YRS_RESIDENCE",stats="location")
```

Complete Quantile Statistics

This example calculates complete quantile statistics for AGE and YRS_RESIDENCE:

```
R> ore.univariate(NARROW, var="AGE,YRS_RESIDENCE",stats="quantiles")
```

A

Third-Party Licenses

This appendix contains licensing information about third-party products installed with Oracle R Enterprise. It contains the following topics:

- R
- ROracle

R

R is an open source language/environment that is governed by GPL2 and not under the terms of the Oracle license agreement.

R was initially written by Robert Gentleman and Ross Ihaka of the Statistics Department of the University of Auckland.

Since mid-1997 there has been a core group with write access to the R source, currently consisting of

Douglas Bates
John Chambers
Peter Dalgaard
Seth Falcon
Robert Gentleman
Kurt Hornik
Stefano Iacus
Ross Ihaka
Friedrich Leisch
Uwe Ligges
Thomas Lumley
Martin Maechler
Duncan Murdoch
Paul Murrell
Martyn Plummer
Brian Ripley
Deepayan Sarkar
Duncan Temple Lang
Luke Tierney
Simon Urbanek
plus Heiner Schwarte up to October 1999 and Guido Masarotto up to June 2003.

For more information go to (`http://www.r-project.org`).

Current R-core members can be contacted via email to R-project.org with name made up by replacing spaces by dots in the name listed above.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License version 2 as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

GNU GENERAL PUBLIC LICENSE Version 2

June 1991

Copyright © 1989, 1991 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it

in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent

licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions

for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any

further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE

COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author

Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.

This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called

something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Code derived from software contributed to Berkeley by Guido van Rossum

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

FIG: Facility for Interactive Generation of figures

Any party obtaining a copy of these files is granted, free of charge, a full and unrestricted irrevocable, world-wide, paid up, royalty-free, nonexclusive right and license to deal in this software and documentation files (the "Software"), including without limitation the rights to use, copy, modify, merge, publish and/or distribute copies of the Software, and to permit persons who receive copies from any such party to do so, with the only requirement being that this copyright notice remain intact.

unzip.h -- IO for uncompress .zip files using zlib

Version 1.01e, February 12th, 2005

Copyright © 1998-2005 Gilles Vollant

This unzip package allow extract file from .ZIP file, compatible with PKZip 2.04g WinZip, InfoZip tools and compatible.

Multi volume ZipFile (span) are not supported.

Encryption compatible with pkzip 2.04g only supported

Old compressions used by old PKZip 1.x are not supported

I WAIT FEEDBACK at mail info@winimage.com

Visit also `http://www.winimage.com/zLibDll/unzip.htm` for evolution

Condition of use and distribution are the same than zlib:

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

ROracle

This product is an open source package for R that allows R users to interact with an Oracle database. It was developed by an individual called David James. Oracle has taken over new development, maintenance and all upgrade activity on this package.

ROracle is licensed under LGPL v.2 or later and not under the terms of your Oracle license agreement. For more information see:

`http://cran.cnr.berkeley.edu/web/packages/ROracle/ROracle.pdf`

GNU Lesser General Public License Version 2.1

February 1999

Copyright © 1991, 1999 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is

not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. The modified work must itself be a software library.

   b. You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

   c. You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

   d. If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

      (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public

License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the library's name and an idea of what it does.

Copyright © year name of author

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

signature of Ty Coon, 1 April 1990

Ty Coon, President of Vice

That's all there is to it!

Index

P

R

S

T

U

V

W