What does the title mean?

1. part: R on Different Platforms

- What is R?
- What is a platform?
  - ‘Platform – the computer hardware and operating system software that runs application software.’ (http://nces.ed.gov/pubs98/tech/glossary.asp)
  - R is available and widely used on various platforms, most notably:
    - OS: Linux, Mac OS, Windows etc.
    - CPU: PowerPC, Sparc, x86, x86-64, Sparc etc.

2. part: The useR’s Point of View

- Who is the useR?
  - I am a useR!
  - You are a useR!
  - A developer is a useR!
  - useRs become developers!
  - Summary: We all are useRs!
  - OK, I will stop being too religious ...

- What is the useR’s view?
  - R is great! (no question)
  - Students, statisticians in research and business
  - Non-statisticians
  - Simon Urbanek
  - Uwe Ligges (and that’s what I will talk about)
Time for an introduction

- Work on a desktop with **Windows**, number crunching under **Linux** and do it with the **Mac** at home!
- We try to answer some useRs’ questions on their way to become developers.
  - What’s common among platforms?
    - Everything – at least almost
  - What’s different between platforms?
    - Not that much

What is the “right” editor to use?

- platform independent (one environment for all platforms!?)
- editor we are most used to (don’t switch!?)
- some choices:
  - Emacs and ESS (Rossini et al., 2004)
  - JGR http://rosuda.org/JGR/
  - vi
  - ...
- some more Windows only choices:
  - notepad
  - SciViews (Grosjean, 2003)
  - Tinn-R (Tinn-R Development Team, 2006)
  - WinEdt with RWinEdt (Ligges, 2003)
  - ...

Where is the GUI?

**GUI**: Graphical User Interface

- JGR http://rosuda.org/JGR/
- Rcmdr (Fox, 2005)
- R-Excel (Baier and Neuwirth, 2003)
- SciViews (Grosjean, 2003)
- ...

- session on Teaching and User Interfaces

Packages

- **Standardized** mechanism to distribute data, code, documentation, ...
- No OS dependency, except:
  - OS dependent functions
  - Linking to OS depending external code
  - Relying on OS depending external software
  - (compiled html in installed Windows packages)
- Easy to build, install, check on **all platforms**
How to install and manage packages?

- Install package **PackageName** into library **LibDir**: 
  R CMD INSTALL -l LibDir PackageName_ver.tar.gz
- ... and again:
  install.packages("PackageName", lib = "LibDir")
- Various further functions such as:
  update.packages(checkBuilt = TRUE)

Installing source packages under Windows

So what? It’s easy!

- Manual: **R Installation and Administration**
  (R Development Core Team, 2006)
- Example: **R Help Desk: Make ‘R CMD’ Work under Windows - an Example** (Ligges and Murdoch, 2005)

Required Software

- R tools:
  http://www.murdoch-sutherland.com/Rtools
- Perl: http://www.activestate.com/Products/ActivePerl/Download.html
- MinGW ‘candidate' compilers (http://www.mingw.org):
  gcc-core-VER.tar.gz, gcc-g++-VER.tar.gz,
  gcc-g77-VER.tar.gz, binutils-VER.tar.gz,
  mingw-runtime-VER.tar.gz and w32api-VER.tar.gz
- \LaTeX{} (e.g. MikTeX):
  http://www.miktex.org/setup.html

Required Settings

- Environment variable ‘PATH’:
  
  PATH=.;c:\devel\tools\bin;c:\devel\MinGW\bin;
  c:\devel\R-2.3.1\bin;c:\devel\HtmlHelp;
  c:\devel\Perl\bin;c:\devel\texmf\miktex\bin;
  %PATH%
- Environment variable ‘TMPDIR’
- Edit file c:\devel\R-2.3.1\src\gnuwin32\MkRules:
  HHWDIR=c:/devel/HtmlHelp
Installing binary packages under Windows

- install a **binary package** (given the repository supports Windows binaries):
  
  ```
  R> install.packages("PackageName",
  
  R+       lib = "LibDir")
  ```

- canonical form of the repository (using R-2.3.1):
  
  ```
  R> contrib.url("http://foo.org/")
  [1] "http://foo.org/bin/windows/contrib/2.3"
  ```

- (at least) for CRAN, it is worth looking into the **check summary**

---

CRAN Windows Binaries’ Package Check

Last updated on 2006-06-13 17:51:39  
(simplified)

<table>
<thead>
<tr>
<th>No</th>
<th>Package</th>
<th>Version</th>
<th>R-2.3.1</th>
<th>Inst. time</th>
<th>Check time</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>741</td>
<td>violplot</td>
<td>0.2</td>
<td>OK</td>
<td>51</td>
<td>46</td>
</tr>
<tr>
<td>742</td>
<td>wavelets</td>
<td>0.2-1</td>
<td>OK</td>
<td>26</td>
<td>88</td>
</tr>
<tr>
<td>743</td>
<td>waveslim</td>
<td>1.5</td>
<td>OK</td>
<td>58</td>
<td>109</td>
</tr>
<tr>
<td>744</td>
<td>wavethresh</td>
<td>2.2-8</td>
<td>OK</td>
<td>30</td>
<td>75</td>
</tr>
<tr>
<td>745</td>
<td>wcscsom</td>
<td>1.1.0</td>
<td>OK</td>
<td>18</td>
<td>87</td>
</tr>
<tr>
<td>746</td>
<td>wle</td>
<td>0.9-2</td>
<td>OK</td>
<td>24</td>
<td>365</td>
</tr>
<tr>
<td>747</td>
<td>xgobi</td>
<td>1.2-13</td>
<td>ReadMe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>748</td>
<td>xtable</td>
<td>1.3-2</td>
<td>OK</td>
<td>22</td>
<td>52</td>
</tr>
<tr>
<td>749</td>
<td>zicounts</td>
<td>1.1.4</td>
<td>WARNING</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>750</td>
<td>zoo</td>
<td>1.1-0</td>
<td>OK</td>
<td>23</td>
<td>60</td>
</tr>
</tbody>
</table>

**SUM (in hours) on a Xeon 3.06 GHz:** 6.34 19.77

---

Which features are platform specific?

- Some packages (binary, and those depending on other platform specific stuff)
- Some memory management
- Some graphics devices
- GUI
- Performance?

---

Platform specific packages

- Windows only CRAN packages (< 10 of 750):
  - BRugs – OpenBUGS
  - mimR – mim interface
  - rcom – R COM interface
  - RWInEdt – WinEdt “interface”
  - spectrino – spectral analysis software
  - tcltk2 – SciViews GUI API
  - ... ?

- non- Windows CRAN packages (< 20 of 750):
  - snow – Simple Network Of Workstations
  - nice – handling UNIX niceness
  - ...
Platform specific functions

- **Graphics devices**, e.g. `windows()`, `win.print()`, `win.metafile()`, `png()`, `bmp()`, `jpeg()`, `savePlot()`, ...
- **GUI**, e.g. `winDialog()`, ...
- **Memory**, e.g. `memory.limit()`, `memory.size()`, ...
- **Package management**, e.g. `install.packages()`, ...
- **Shell**, e.g. `shell()`, `shell.exec()`, `(system())`, ...
- **Others**, e.g. `read.clipboard()`, ...

---

### Memory

- 32 bit OS: limited address space
- Windows (R only supported on 32 bit versions!):
  - address space for a single process: 2Gb (or up to 3Gb)
  - command line flag `--max-mem-size`: default is the smaller of the amount of physical RAM and 1024Mb
  - `memory.size()`: ‘reports the current or maximum memory allocation’
  - `memory.limit()`: ‘reports or increases the limit in force on the total allocation’
  - *R for Windows FAQ* (Ripley and Murdoch, 2006)
- 64 bit OS: good reason for > 1 platforms, if you do not want to discard Windows

---

### Speed

- Linux wins vs. Windows (∼10%)
- in the old days, gcc for 64 bit resulted in slow code
- **optimized (B)LAS**:
  - standard
  - ACML (AMD Core Math Library)
  - ATLAS (Automatically Tuned Linear Algebra Software)
  - Goto’s BLAS
  - benchmark example:
    ```
    set.seed(123)
    X <- matrix(rnorm(1e6), 1000)
    Z <- X + 0i
    ```

---

### Benchmark under Windows

**Athlon 2000+ (1.67 GHz), Windows NT4**

<table>
<thead>
<tr>
<th>Function</th>
<th>base</th>
<th>ACML</th>
<th>ATLAS</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>crossprod(X)</code></td>
<td>6.0</td>
<td>1.0</td>
<td>0.6</td>
</tr>
<tr>
<td><code>X %*% X</code></td>
<td>12.0</td>
<td>1.0</td>
<td>0.8</td>
</tr>
<tr>
<td><code>Z %*% Z</code></td>
<td>26.7</td>
<td>3.4</td>
<td>3.2</td>
</tr>
<tr>
<td><code>solve(X, LINPACK=T)</code></td>
<td>17.5</td>
<td>17.4</td>
<td>15.5</td>
</tr>
<tr>
<td><code>solve(X)</code></td>
<td>12.4</td>
<td>2.2</td>
<td>2.0</td>
</tr>
<tr>
<td><code>solve(Z)</code></td>
<td>26.7</td>
<td>6.2</td>
<td>5.7</td>
</tr>
<tr>
<td><code>svd(X)</code></td>
<td>46.9</td>
<td>13.7</td>
<td>11.6</td>
</tr>
<tr>
<td><code>svd(Z)</code></td>
<td>170.4</td>
<td>126.8</td>
<td>126.8</td>
</tr>
</tbody>
</table>
Benchmark on “server systems”

(2x) Xeon 3.06GHz, Windows 2003 Server vs. SuSE Linux 9.0

<table>
<thead>
<tr>
<th></th>
<th>Windows</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>base</td>
<td>ATLAS</td>
</tr>
<tr>
<td>crossprod(X)</td>
<td>2.5</td>
<td>0.4</td>
</tr>
<tr>
<td>X %*% X</td>
<td>5.2</td>
<td>0.7</td>
</tr>
<tr>
<td>Z %*% Z</td>
<td>11.9</td>
<td>2.4</td>
</tr>
<tr>
<td>solve(X, LINPACK=T)</td>
<td>11.5</td>
<td>12.6</td>
</tr>
<tr>
<td>solve(X)</td>
<td>6.1</td>
<td>1.8</td>
</tr>
<tr>
<td>solve(Z)</td>
<td>13.3</td>
<td>4.8</td>
</tr>
<tr>
<td>svd(X)</td>
<td>22.6</td>
<td>9.0</td>
</tr>
<tr>
<td>svd(Z)</td>
<td>110.0</td>
<td>92.1</td>
</tr>
</tbody>
</table>

Simulations under Unix

- long lasting number crunching, simulations
- R jobs with low priority in the background on Unix-like systems:
  
  ```bash
  nohup nice -n 14 R CMD BATCH myRfile.R &
  ```

- nohup: do not hang up on logout
- nice: be nice to other processes

References


Mac OS duality

- **Mac OS**
  - user ≠ developer
  - focus on usability, friendly user interface (UI guidelines)
  - attention to detail

- **unix**
  - user = developer
  - focus on versatility, extensibility

**Mac OS X = Mac + unix**

R for “Mac” users

- **Genuine Mac look and feel**
  - Aqua look
  - drag & drop

- **Leverage common OS technologies**
  - seamless PDF display and data exchange
  - localization
  - AppleScript

- **Friendly user interface (do not require “user = developer”)**

R for “unix” users

- **BSD-based unix (Darwin) + GNU tools**
  - command line R (ESS, shell-scriptable etc.)
  - X11 devices (jpeg, png, ...)
  - unix libraries (Tcl/Tk, Gtk, ..)

- **“flavors”**
  - unix style - e.g. /usr/local/lib/R/

- Mac OS X framework (includes classic unix structure in the Resources subdirectory)
R for developers: R Framework anatomy

- `/Library/Frameworks/R.framework`
  - Versions
    - Current → 2.3
  - Resources (R_HOME)
    - ...
  - Headers → Versions/Current/Resources/include
  - Resources → Versions/Current/Resources

Universal binary

- Universal = Intel + PowerPC
- “Fat” files
  - one file, multiple architectures (ppc, ppc64, i386, ...)
    - gcc -arch ppc -arch i386 -o foo.o foo.c
      (Mac OS X only)
- sub-architectures in R
  - separate binary directories for each architecture
  - e.g. package/libs/ppc vs package/libs/i386
  - any unix-based system (e.g. x86, x86_64)

Universal R

- Share common files (documentation, R code, ...)
- Separate binaries and configuration (Makeconf, ...)
- “Fat” framework (libR.dylib)
- Universal binary packages
  - contain “libs” for both architectures (ppc+x86)
  - can be cross-compiled on Intel Macs for PowerPC Macs
  - available from CRAN and Bioconductor
    (see http://R.research.att.com/ for nightly builds and results)

Compiling source packages

- Requirements
  - R
  - Apple Xcode Tools (included in Mac OS X)
  - gcc 4.0.3 (included in CRAN R)
  - teTeX (optional, available e.g. via i-Installer)
- Installation
  - install.packages(“foo”, type=“source”)
  - R CMD INSTALL foo_0.1.tar.gz
  - universal build with configure script needs 2nd step:
    R_ARCH=/ppc R CMD --libs-only INSTALL foo_0.1.tar.gz
Writing portable binary packages

- Use Makevars whenever possible
  - multi-arch binaries are then built automatically
- Use Makefiles only in conjunction with autoconf
- Don’t use common symbols?
  - no variable definitions in headers files
    (test with -fno-common in gcc)
- Don’t assume the library extension is .so

Summary

- R runs on many platforms - official binaries for Windows and Mac OS X - contributed binaries for Linux distributions
- R attempts to accommodate different user types on different platforms
- Package installation consistent on all platforms
- Work on a desktop with Windows, number crunching under Linux and do it with the Mac at home

Contact

- **Stefano Iacus**
  - Universita degli Studi di Milano, Italy
- **Uwe Ligges**
  - Fachbereich Statistik, Universität Dortmund, Germany
- **Simon Urbanek**
  - AT&T Labs - Research, Florham Park, NJ