



R CODE AND IDE OF CHOICE

Stan Smoltis

About myself

- Data science enthusiast
- Database analyst and software engineer
 - Query tuning and optimising for high performance
- Last 6 years: designing ETL projects and data warehouses
- MCSE: Data platform

Stan

s.smoltis@gmail.com

Agenda

- What IDE is
- Some interesting stats
- Prerequisites
- Demo

IDE

An application that provides facilities for software development

- Source editor
- Build Automation tools
- Debugger
- Intelligent code completion
- Compiler/Interpreter
- Version Control System integration

Top IDE index

- The more an IDE is searched, the more popular the IDE is assumed to be. (Google Trends)

Worldwide, Jul 2016 compared to a year ago:

Rank	Change	IDE	Share	Trend
1	↑	Visual Studio	23.35 %	+0.6 %
2	↓	Eclipse	23.07 %	-5.8 %
3	↑	Android Studio	10.09 %	+2.7 %
4	↓	Vim	7.99 %	-0.0 %
5		NetBeans	5.62 %	-0.4 %

TOP programming languages

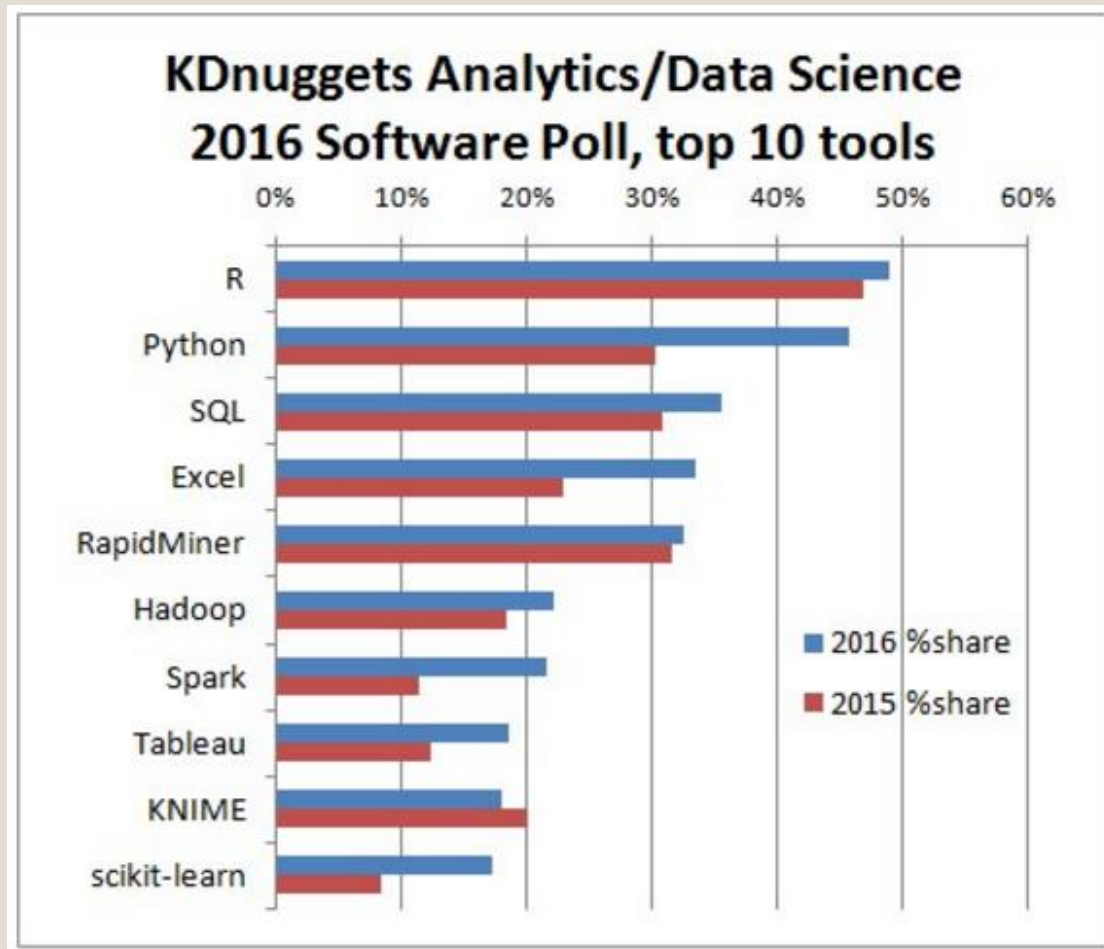
Worldwide, Jul 2016 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Java	23.8 %	-0.7 %
2	↑	Python	13.0 %	+2.3 %
3	↓	PHP	10.5 %	-0.7 %
4		C#	9.0 %	-0.3 %
5	↑↑	Javascript	7.7 %	+0.7 %
6	↓	C++	7.2 %	-0.4 %
7	↓	C	7.0 %	-0.2 %
8		Objective-C	4.5 %	-0.8 %
9	↑	R	3.2 %	+0.6 %
10	↑	Swift	3.0 %	+0.3 %

Top Analytics/Data Science Tools

Tool	2016 % share	% change	% alone
R	49%	+4.5%	1.4%
Python	45.8%	+51%	0.1%
SQL	35.5%	+15%	0%
Excel	33.6%	+47%	0.2%
RapidMiner	32.6%	+3.5%	11.7%
Hadoop	22.1%	+20%	0%
Spark	21.6%	+91%	0.2%
Tableau	18.5%	+49%	0.2%
KNIME	18.0%	-10%	4.4%
scikit-learn	17.2%	+107%	0%

Kdnuggets 2016 Software Poll: top 10 most popular tools in 2016



Getting started with RTVS

- R interpreter
- IDE
- Intel Math Kernel Library & Math Library (for high performance computing)

R development in Visual Studio:

Step 1: Download Visual Studio



Step 2: Download R Tools for VS

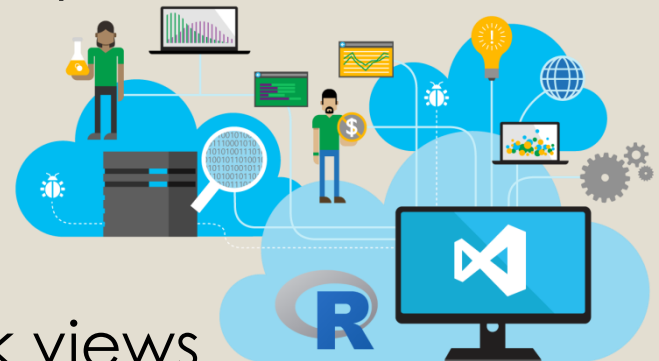


Step 3: Download Microsoft R Open

R tools for VS

<https://aka.ms/rvs-current>

- v0.4 (June, 2016 release)
- Visual Studio 2015 Update 2 (64bit only)
- RevoMath with MKL
- Open Source and Free
- Support for CRAN R, Microsoft R Open, Microsoft R Server
- R aware Editor with IntelliSense
- Plots and Shiny apps
- Debugger with Locals and Stack views



New features in v0.4

- Multiple Data Table Viewer window instances
- Sort columns in Data Table Viewer
- Delete one or all variables with Variable Explorer
- Go-to-definition and peek for library functions
- Collapsible code regions surrounded by --- and ---- comments
- Full support for R editor features in RMarkdown code block
- Internet keyword search with F1
- Auto-formatting improvements
- Additional ways of setting current working directory
- Plot window uses global Window DPI setting, and locator API is now supported
- **R templates in SQL Database project**
- Microsoft R Client integration

Microsoft R: Product Comparison

Features	Microsoft R Open	Microsoft R Client	Microsoft R Server
Big Data	In-memory bound Can only process datasets that fit into the available memory	In-memory bound Can process datasets that fit into the available memory Operates on large volumes when connected to R Server	Disk scalability Operates on bigger volumes & factors
Speed of Analysis	Multi-threaded when MKL is installed for non-ScaleR functions	Multi-threaded with MKL for non-ScaleR functions Up to 2 threads for ScaleR functions with a local compute context	Full parallel threading & processing
Enterprise Readiness	Community support	Community support	Commercial support
Analytic Breadth & Depth	8000+ open source packages	Leverage & optimize open source R packages plus 'Big Data'-ready ScaleR packages	Leverage & optimize open source R packages plus 'Big Data'-ready + Multithreaded ready ScaleR packages
Commercial Viability	Risk of deployment to open source	Free for everyone	Commercial licenses
DeployR Enterprise	Not available	Not available	Included

Demo 1: Look and feel

The image displays the RStudio interface with three overlapping windows illustrating the look and feel of the software.

Variable Explorer (Top Left): This window shows the structure of the data frame. The left pane lists variables: `con`, `df_TM`, `@.Data`, `@.names`, `@.row.names`, `@.S3Class`, `CustomerKey`, `MaritalStatus`, and `Gender`. The right pane shows the value for each variable, such as `Class 'RODBC' 4` for `con` and `chr [1:12] "CustomerKey" ...` for `df_TM`.

Environment (Top Right): This window shows the global environment. The left pane lists the objects: `con`, `df_TM`, `@.Data`, `@.names`, `@.row.names`, `@.S3Class`, `CustomerKey`, `MaritalStatus`, and `Gender`. The right pane shows the class and type of each object, such as `RODBC` (integer) for `con` and `data.frame` (list) for `df_TM`.

Data (Bottom): This window shows the data frame `df_TM`. The left pane shows the variable names: `CustomerKey`, `MaritalStatus`, `Gender`, `TotalChildren`, and `NumberChildren`. The right pane shows the data values, such as `CustomerKey : int 11000 11001 11002 11003 11004 11005 11006 11007 11008` and `MaritalStatus : Factor w/ 2 levels "M", "S": 2 2 2 1 1...`.

Demo 2: Matrix Calc

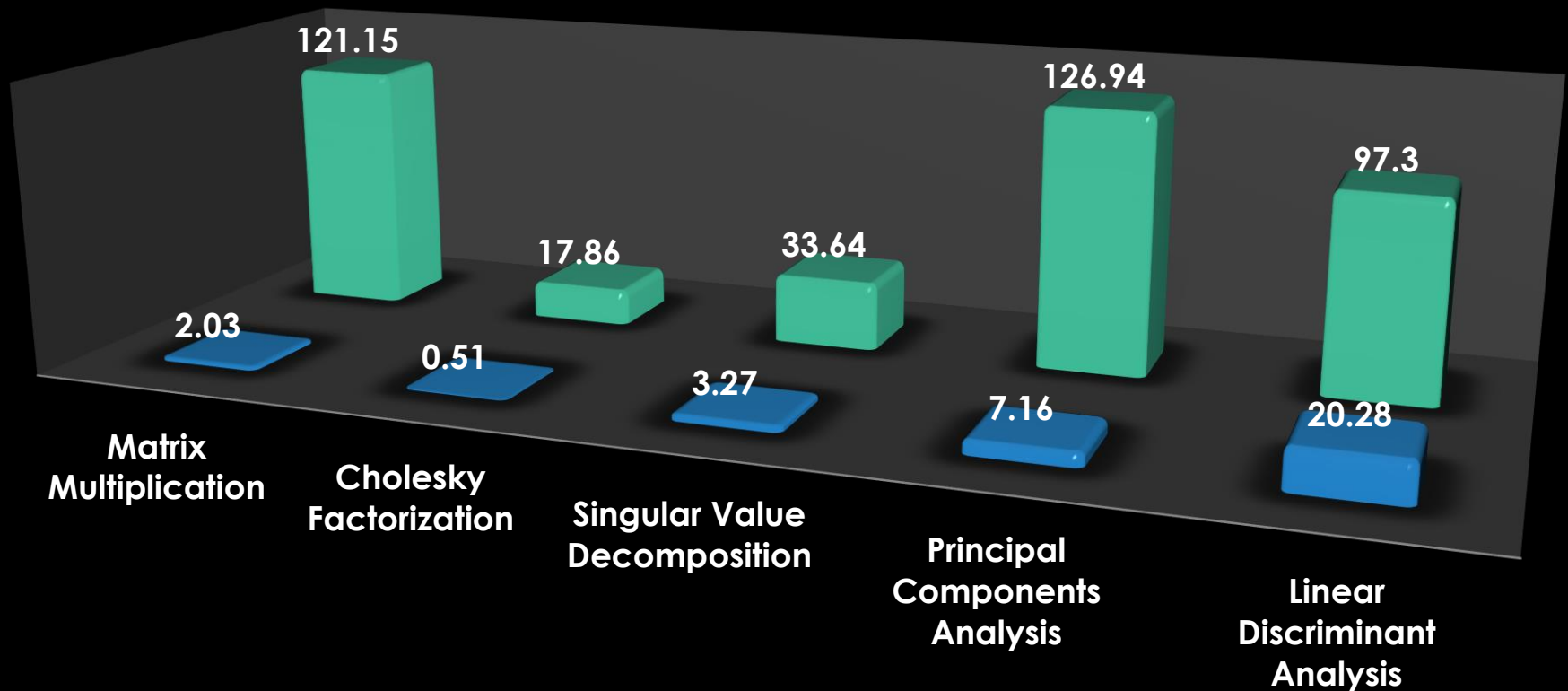
Dear Algebra,
Stop asking me to
find your x .
She's never coming
back, and don't ask y .

Computing performance

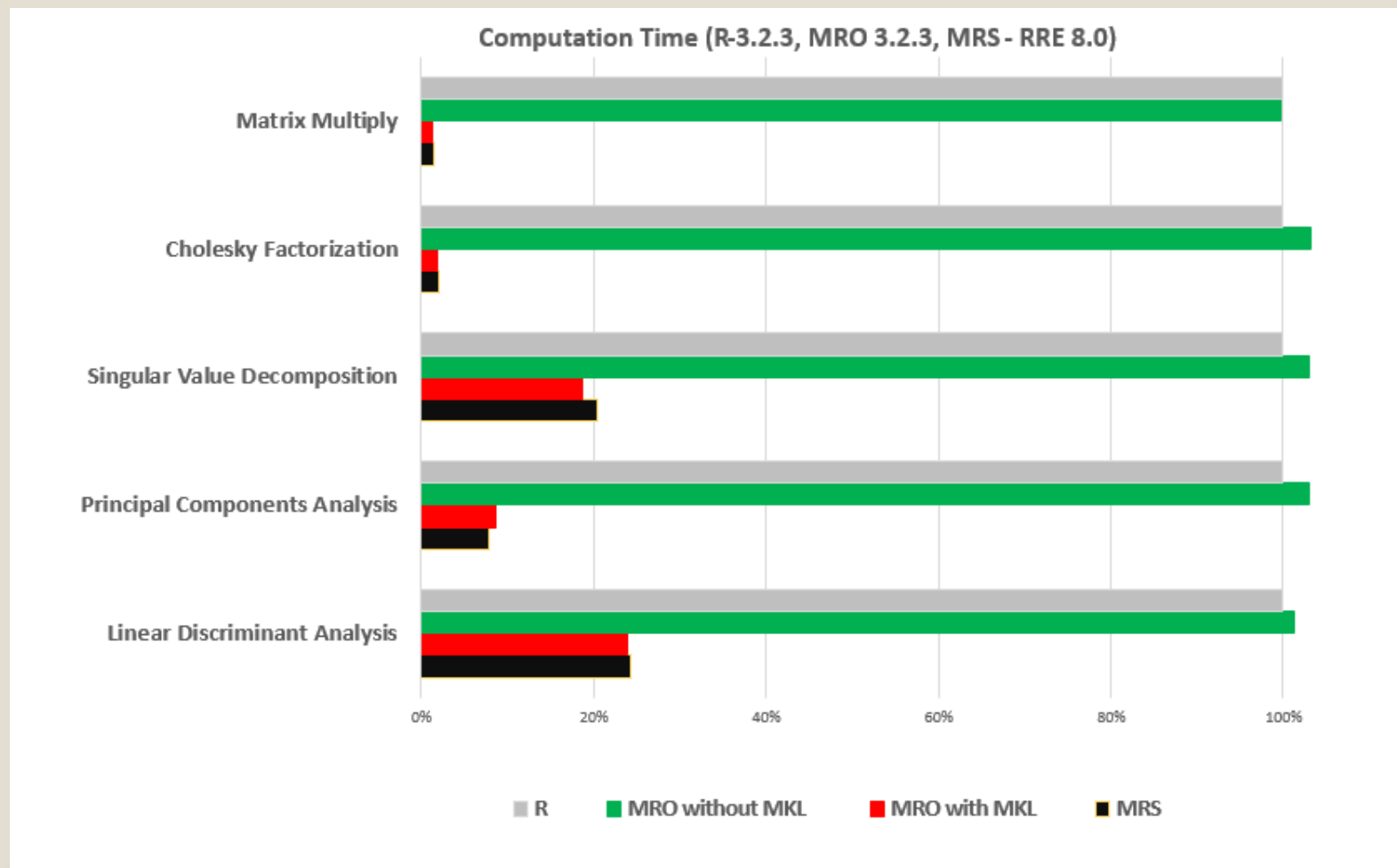
Duration, sec

■ Open R 3.2.5 w/ RevoUtilsMath (4 threads)

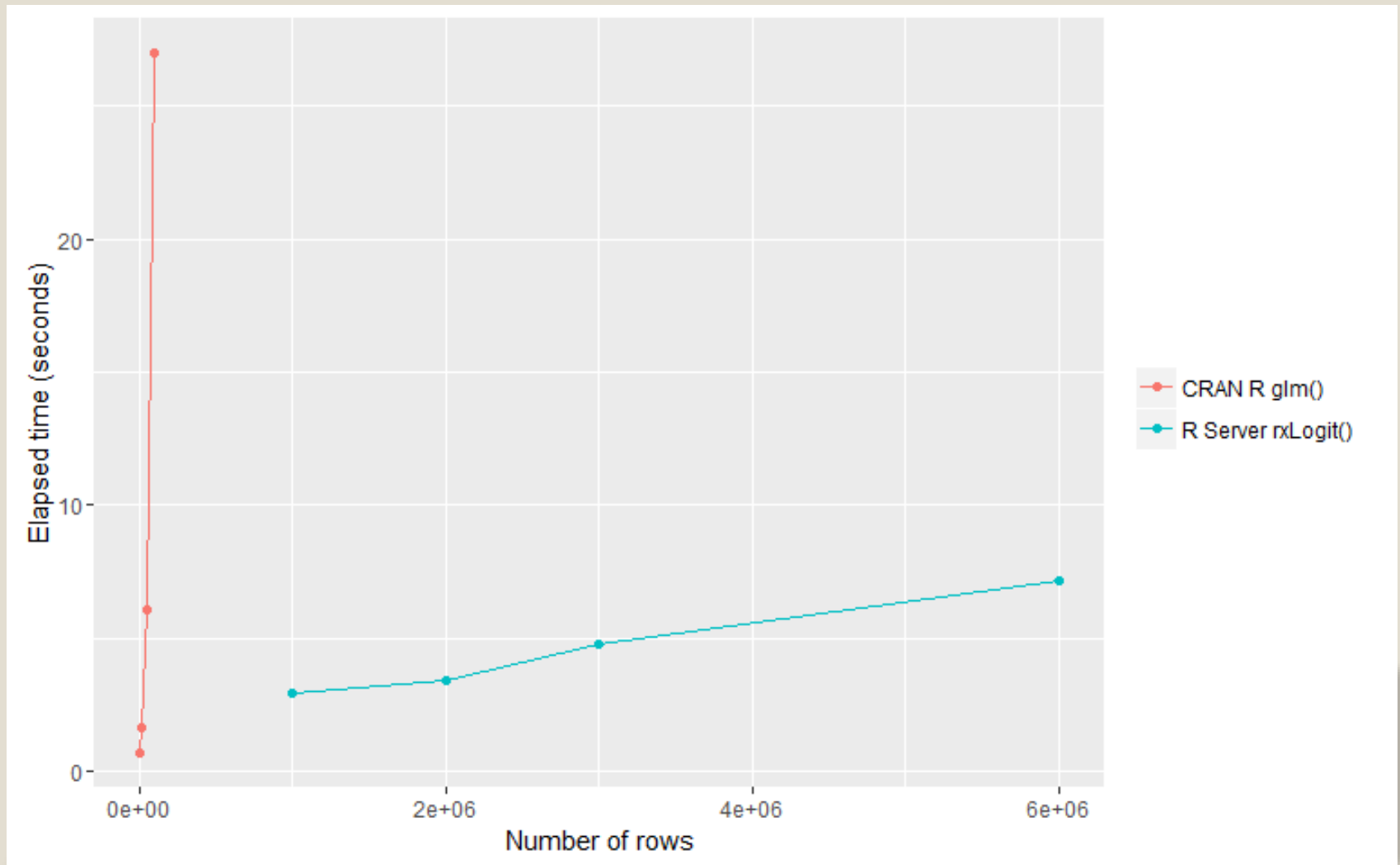
■ CRAN R 3.3.1



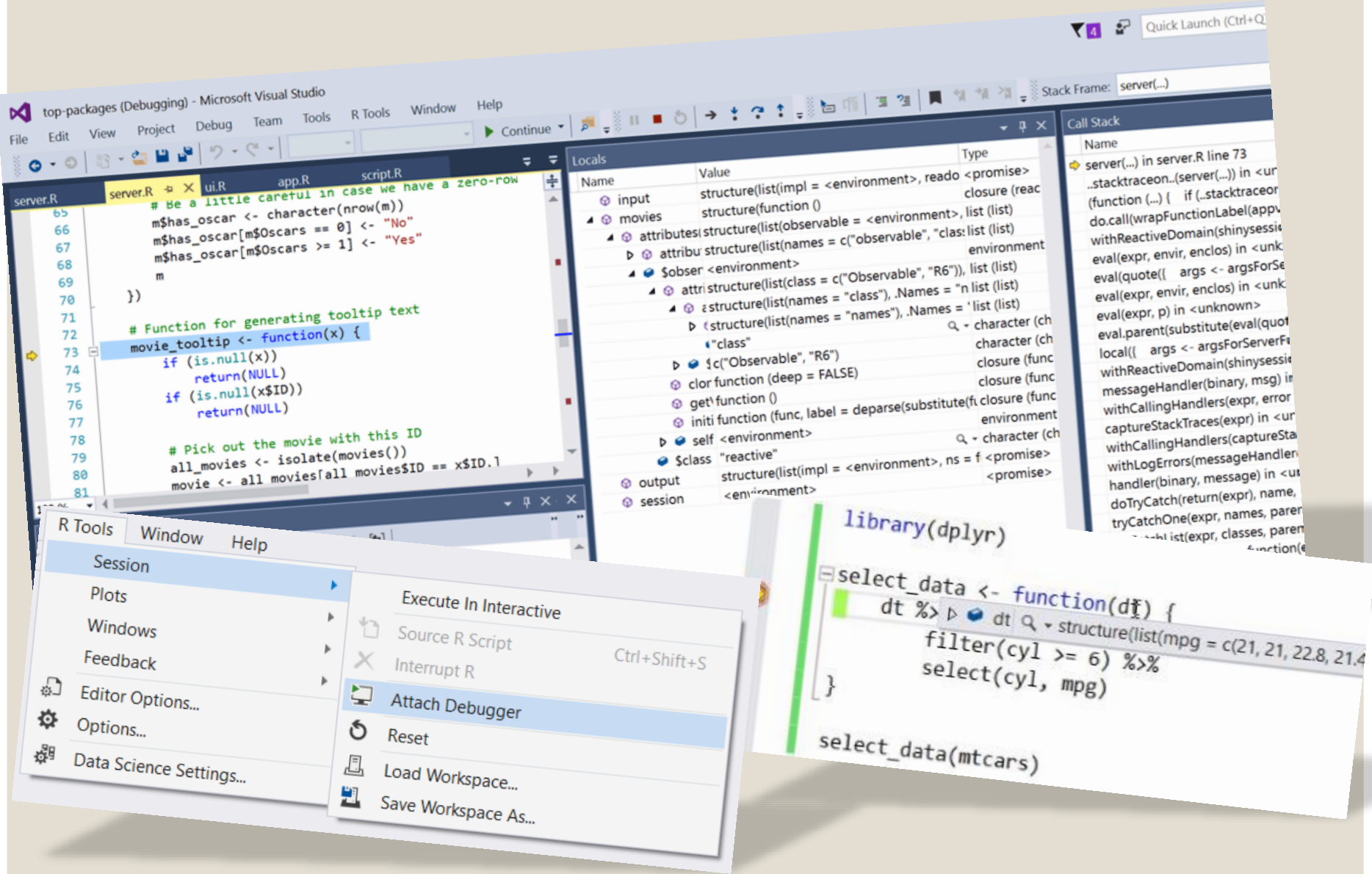
Better computation performance



Logistic Regression scalability



Demo 3: Debugging

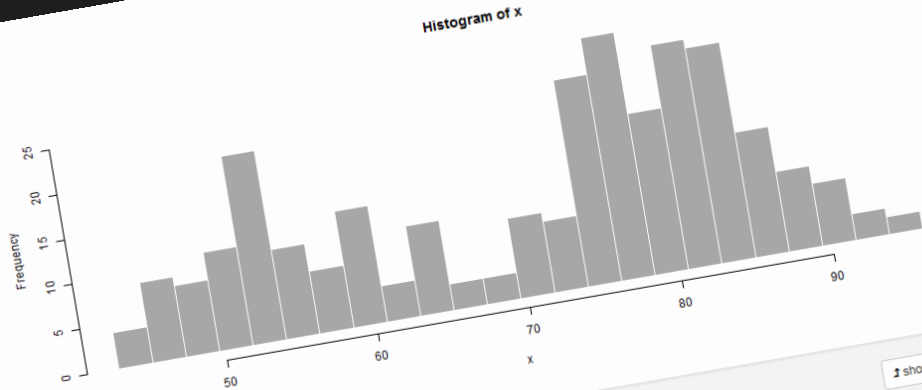
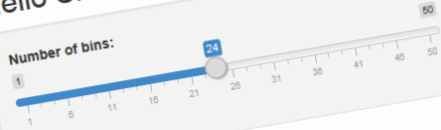


Demo 4: Shiny app

```
library(shiny)
ui <- fluidPage()
server <- function(input, output, session) {}
shinyApp(ui = ui, server = server)
```

Hello Shiny!

Number of bins:



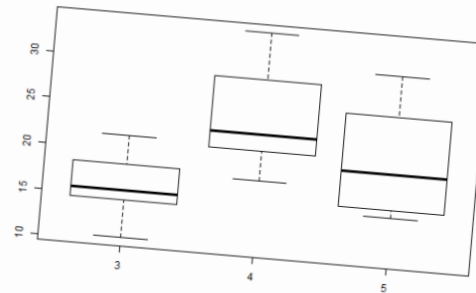
Miles Per Gallon

Variable:

Gears

☒ Show outliers

mpg ~ gear



Hello Shiny!
by RStudio, Inc.

This small Shiny application demonstrates how to use the Number of bins slider. Move the Number of bins slider and notice how the histogram is automatically re-evaluated when its dependency is changed, causing a histogram with a new number of bins.

```
server.R  ui.R
library(shiny)
library(datasets)

# We tweak the "am" field to have nicer factor labels. Since
# this doesn't rely on any user inputs we can do this once at
# startup and then use the value throughout the lifetime of the
# application
mpgData <- mtcars
mpgData$am <- factor(mpgData$am, labels = c("Automatic", "Manual"))

# Define server logic required to plot various variables against
# mpg
shinyServer(function(input, output) {
  # Compute the formula text in a reactive expression since it
  # is shared by the output$caption and output$mpgPlot functions
  formulaText <- reactive({
    paste("mpg ~", input$variable)
  })

  # Return the formula text for printing as a caption
  output$caption <- renderText({
    formulaText()
  })

  # Generate a plot of the requested variable against mpg and
  # only include outliers if requested
  output$mpgPlot <- renderPlot({
    boxplot(as.formula(formulaText()),
            data = mpgData,
            outline = input$outliers)
  })
})
```

Thanks for watching

Stan

s.smoltis@gmail.com