Universidad Rey Juan Carlos

Master Universitario en Ingeniería de Sistemas de Decisión

Curso Académico 2010 / 2011

Proyecto Final: Tesis de Master

An Introduction to R for Quality Control

Autor: Emilio López Cano

Tutores: Javier M. Moguerza

Andrés Redchuk

Junio de 2011

An Introduction to R for Quality Control

Emilio López Cano

Master's Thesis.

Master in Decision Systems Engineering

Thesis Advisors: Javier M. Moguerza and Andrés Redchuk

Rey Juan Carlos University, Madrid

Contents

TABLE OF ILLUSTRATIONS

Abstract

Six Sigma is a breakthrough strategy in quality management, based in the outline DMAIC (Define, Measure, Analyze, Control) and the use of statistical techniques. R is a free statistical software environment that has spread along the academic and research world, even though not yet in enterprise environment. This essay tries to join both systems, performing some Six Sigma tasks with R, and to show the advantages of using R in an enterprise context. Besides, it entails an R Package development that will be shared with the R-project. The scope of the subject exceeds the requirements of a master thesis. Therefore there has been planned further research and development.

Resumen

Seis Sigma es una estrategia de éxito aplicada a la gestión de la calidad. Se basa en el esquema DMAIC (Definir, Medir, Analizar, Mejorar y Controlar) y la utilización de técnicas estadísticas. R es un software estadístico libre, que se ha implantado en la comunidad académica y científica, pero no así en el entorno empresarial. Esta tesis trata de conjugar ambos sistemas, realizando algunas de las tareas de Seis Sigma con R, y mostrar las ventajas de utilizar R en un contexto empresarial. Además, implica el desarrollo de un "paquete" para R, que será compartido en el proyecto R. El alcance de la materia supera los requerimientos de una Tesis de Master, y por lo tanto se ha previsto continuar con su desarrollo más adelante.

Acknowledgements

I would like to thank my thesis advisors Andrés Redchuk and Javier M. Moguerza, foremost for having worth my own proposal of Master Thesis theme. They saw immediately my effort and power of the subject, and have supported and encouraged me to go further.

It has been gratifying to study the Master in Decision Systems Engineering. It is a pleasure to thank the Statistics and Operations Research Department of Rey Juan Carlos University (Madrid) for the training and knowledge acquired.

The thesis is based in the use of R system. So I am really indebted with the R Core Team (http://www.r-project.org/contributors.html) for supporting the R Project.

Last but not least, I acknowledge my wife and daughters for their understanding and bearing the lost sharing time. Thanks Ángela, Lucía and Sonia.

Agradecimientos

Quiero agradecer a los tutores de la tesis, Andrés Redchuk y Javier M. Moguerza, sobre todo por tener haber valorado mi propia propuesta sobre el tema de la Tesis. Valoraron rápidamente mi esfuerzo y el potencial del tema, y me han apoyado y animado a ir más allá.

Ha sido muy gratificante estudiar el Master en Ingeniería de Sistemas de Decisión. Es un placer agradecer al departamento de Estadística e Investigación Operativa de la en la Universidad Rey Juan Carlos de Madrid, por la formación y los conocimientos adquiridos.

La tesis está basada en el uso del sistema R. Por tanto estoy en deuda con el “Core Team” (http://www.r-project.org/contributors.html) de R, por mantener el Proyecto R.

Por último, pero no menos importante, agradezco a mi mujer y mis hijas por su comprensión y por aguantar el tiempo que no he pasado con ellas. Gracias Ángela, Lucía y Sonia.

1. Introduction

1.1. Motivation

Even though Six Sigma methodology sprang up in an enterprise environment, inside Motorola Company, its use spreads every activity area, not only in private companies, but also out of enterprise activities, such as Education, Public Administration, International Cooperation, Health or **Research** and Development.

Focusing in that latter area, and its multidisciplinary skill, **free technologies** are more and more used in everyday work and in new research ways as well.

Thus, for statistical analysis (regardless of complexity level) the use of **`R`**[1] statistical software is spreading all over the research community, for the most part owed to the contributions with **`R-packages`**.

In Six Sigma environment, Minitab[2] is the most used software. This software aims for all-purpose statistical analysis, and it has lots of specific tools for Quality Management, like SPC, Gage R&R, Design of Experiments, Six Sigma, etc. Thus, the Master subject "Quality: Six Sigma" is taught using that important software.

In the other hand, though there is an immense number of specific R-packages available (2.639 at CRAN repositories nov 21, 2010), and growing (2.868 mar 02, 2011) some areas haven't been developed so prolifically, as regarding Quality Control and Improvement tools. The only packages related to those issues are:

- **`AcceptanceSampling`**[3], for some acceptance sampling techniques.
- **`IQCC`**[4], for improved quality control charts
- **`qcc`**[5], for quality control charts
- **`qualityTools`**[6], a package for teaching statistical methods in the field of Quality Science.

[1] (R Development Core Team (2010). R: A language and environment for statistical computing. R Foundation for Statistical Computing)
[2] (Minitab Inc. (2010) website: http://www.mititab.com)
[3] (Kiermeier, 2008)
[4] (Daniela R. Recchia <daniela_recchia@yahoo.com.br>, Emanuel P. Barbosa and Elias de Jesus Goncalves <eliasjg@gmail.com>, 2010)
[5] (Scrucca, 2004)

All in all, this essay tries to both report the way one can perform some of the subjects' samples and activities using R software, and design new functions in order to make easy those tasks.

Those functions will be gathered in an R-package, which can eventually be considered by the community as a contribution as it is finished.

1.2. Outline

This introduction is the **input** for the next process (using Six Sigma Terminology): A Contents Index, the subject contents and activities, and some bibliography.

The Content Index is part (see "further research" in chapter 4) of the Six Sigma steps: **Define, Measure, Analyze, Improve and Control** (DMAIC). That introduction can be considered as a part of Definition at that scope.

This document doesn't aim at explain statistical theory, or Six Sigma Methodology (they will be taking in account in further research) Therefore, these issues won't be explained, though there will be shown some methodological notes when needed. See bibliography references to drill into both disciplines.

Every chapter will have its own **output**, which will compound these items:

- The syntax necessary to perform the statistical analysis with standard R, related to the samples and activities included in the subject material.
- The output (text and graphs) obtained with that syntax.
- When needed, the syntax of the functions developed to be included in the future package, and its outputs.
- As appendix:
  - The complete functions code.
  - The help contents for the functions.
  - The data used.

6 (Roth, 2010)

Besides, there is an output from the whole essay:

- The package with the development, including data sets.
- The Documentation file for the development
- A presentation with the results.

The package and its functions have been defined in the R wiki[7] and the package is now available at CRAN.

As in the Master subject, the famous "Paper Helicopter Manufacturing" will be used in the samples, datasets and results. This example is widely used in Six Sigma literature [(Allen, 2010) …]

1.3. Functions developed and planned

For this essay, there have been developed these functions:

- Define
  - `ss.pMap`.- Process map representation
  - `ss.ceDiag`.- Cause-and-Effect Diagram
- Measure
  - `ss.rr`.- Gage R&R computations and graphics

The functions that are planned to develop as "Further Research" are:

- Define
  - `ss.ad`.- Affinity Diagram
- Anayze
  - `ss.ci`.- Confidence interval for the response
  - `ss.ca`.- Capability Analysis
  - `ssCompare`.- Hypothesis test for factors
  - `ssGraphSum`.- Summarizer graphics: Histogram, pareto, normal, dotplot, time series, scatter, tree, …
  - `ssReg`.- Regression Analysis
  - `ssAnova`.- Analysis of Variance
- Improve
  - `ss.fa`.- Factorial and fractional Analysis

[7] (http://rwiki.sciviews.org/)

    - `ss.sr`.- Surface Response Analysis
    - `ss.pm`.- Priorization Matrix
- Control
    - `ss.cc`.- Control Charts
    - `ss.as`.- Acceptance Sampling
    - `ss.OCCurve`.- Operation Characteristic Curve
    - `ss.tg`.- Tolgate Review
    - `ss.rc`.- Run charts

Further Research will include as well some up-to-date topics in Six Sigma and Lean, such as Simulation or Optimization. It obviously entails new and powerful functions. Thoughts about how **Bayesian Statistics** could improve Six Sigma methodology will be accounted too. Chapter 3 has a piece of that issue.

The prime source used in the next chapters has been the material from the Master subject. Other books have been an inspiration as well, and they are recommended to study and understand both Statistics (Six sigma included) and R: (Chambers, 2008), (Crawley, 2007), (Allen, 2010) and (Bolstad, 2007).

1.4. Getting Started with R

1.4.1. Introduction

It seems to be that Free Software is only for Universities and Researchers. Even though some IT Company managers think that free software is a strong threat for their business. I'll try to demystify some issues about that.

In an industrial environment, one can ask himself: "Do you mean I must rely on Free Software for my critical process management? Who will guarantee the SLA we need? Who will I complain to when something is wrong?

I think the weaknesses are much less than the strengths. R is a project where thousands of people are working every day, improving the system, and sharing their experience and their problems. You can find (almost surely) the answer to your request faster than through a "support ticket" to a software Company, unless you have a very expensive service contract.

Besides, you have the source of the program, and if something is wrong, you can surely find a professional or a company that can fix it.

But let be realistic. You need a little more effort in order to work with R than working with “clicking software”. You need in your company both Statistics and Informatics professionals. This would not have to be a problem, because most Companies have IT stuff. And if you are involved in Six Sigma projects, you’d better have a Black Belt inside.

It is worth making that effort. One example may illustrate the advantages. Let’s think about a single Control Chart. You can find lots of programs and packages to print a control chart with the data of your process. Some of them can be configured with lots of options. But what if your process has a particular issue distinct of all the options configured in those programs? And, did you see that scientific article where there was discover a new formula that make sense for our critical out of control process? How can we implement it in our Sig Sigma project?

The answer is R.

About R

You can find updated information about the R project in the project website: http://www.r-project.org.

Kurt Hornik (Hornik, 2010) has written a short but precise definition of R:

> “R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files.
>
> The design of R has been heavily influenced by two existing languages: Becker, Chambers & Wilks' S (see What is S?) and Sussman's Scheme. Whereas the resulting language is very similar in appearance to S, the underlying implementation and semantics are derived from Scheme. See What are the differences between R and S?, for further details.
>
> The core of R is an interpreted computer language which allows branching and looping as well as modular programming using functions. Most of the user-visible functions in R are written in R. It is possible for the user to interface to procedures written in the C, C++, or FORTRAN languages for efficiency. The R distribution contains functionality for a large number of statistical procedures. Among these are: linear and generalized linear models, nonlinear regression models, time series analysis, classical parametric and nonparametric tests, clustering and

smoothing. There is also a large set of functions which provide a flexible graphical environment for creating various kinds of data presentations. Additional modules ("add-on packages") are available for a variety of specific purposes (see R Add-On Packages).

R was initially written by Ross Ihaka and Robert Gentleman at the Department of Statistics of the University of Auckland in Auckland, New Zealand. In addition, a large group of individuals has contributed to R by sending code and bug reports.

Since mid-1997 there has been a core group (the "R Core Team") who can modify the R source code archive. The group currently consists of Doug Bates, John Chambers, Peter Dalgaard, Robert Gentleman, Kurt Hornik, Stefano Iacus, Ross Ihaka, Friedrich Leisch, Thomas Lumley, Martin Maechler, Duncan Murdoch, Paul Murrell, Martyn Plummer, Brian Ripley, Duncan Temple Lang, Luke Tierney, and Simon Urbanek.

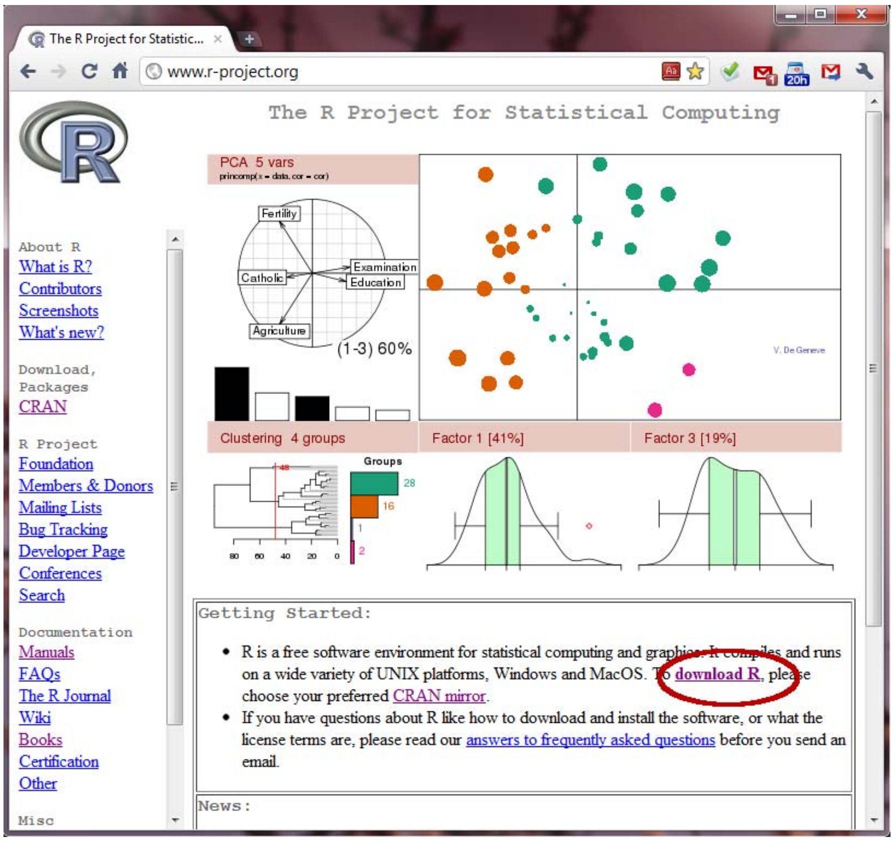
R has a home page at http://www.R-project.org/. It is free software distributed under a GNU-style copyleft, and an official part of the GNU project ("GNU S")."

1.4.2. Installation

The deployment of R on Windows systems[8] could be summarized this way:

1. Download the executable installer. Available the last version at CRAN[9] repositories.

[8] For last versions of Linux, it can be installed through the repositories. Visit the manuals for Linux and Mac installation at http://www.r-project.org
[9] CRAN: The Comprehensive R Archive Network



The R Project for Statistical Computing

About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

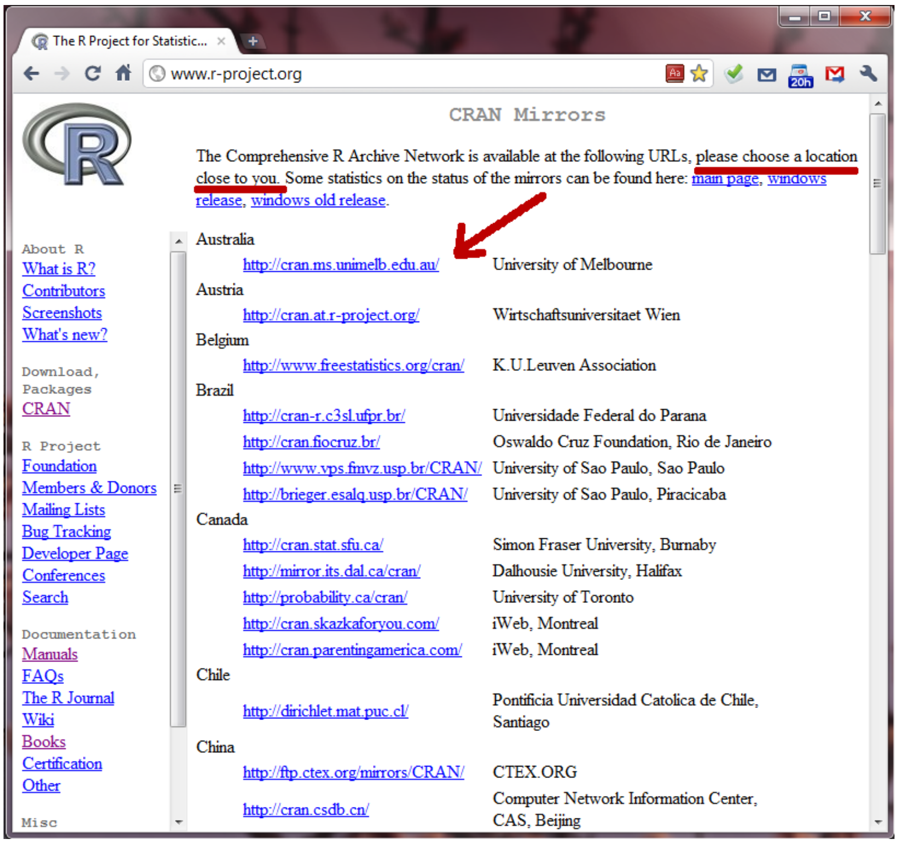
Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

Misc

Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To download R, please choose your preferred CRAN mirror.
- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News:

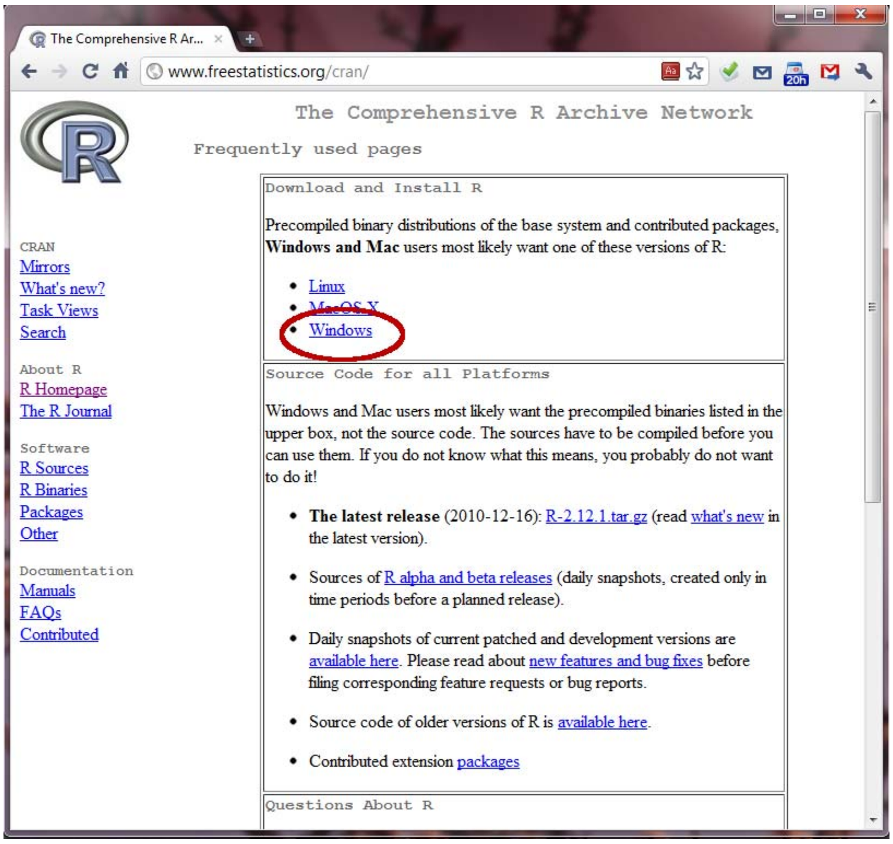


CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

Australia	
http://cran.ms.unimelb.edu.au/	University of Melbourne
Austria	
http://cran.at.r-project.org/	Wirtschaftsuniversitaet Wien
Belgium	
http://www.freestatistics.org/cran/	K.U.Leuven Association
Brazil	
http://cran-r.c3sl.ufpr.br/	Universidade Federal do Parana
http://cran.fiocruz.br/	Oswaldo Cruz Foundation, Rio de Janeiro
http://www.vps.fmvz.usp.br/CRAN/	University of Sao Paulo, Sao Paulo
http://brieger.esalq.usp.br/CRAN/	University of Sao Paulo, Piracicaba
Canada	
http://cran.stat.sfu.ca/	Simon Fraser University, Burnaby
http://mirror.its.dal.ca/cran/	Dalhousie University, Halifax
http://probability.ca/cran/	University of Toronto
http://cran.skazkaforyou.com/	iWeb, Montreal
http://cran.parentingamerica.com/	iWeb, Montreal
Chile	
http://dirichlet.mat.puc.cl/	Pontificia Universidad Catolica de Chile, Santiago
China	
http://ftp.ctex.org/mirrors/CRAN/	CTEX.ORG
http://cran.csdb.cn/	Computer Network Information Center, CAS, Beijing

Select a mirror close to your location



The Comprehensive R Archive Network

Frequently used pages

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

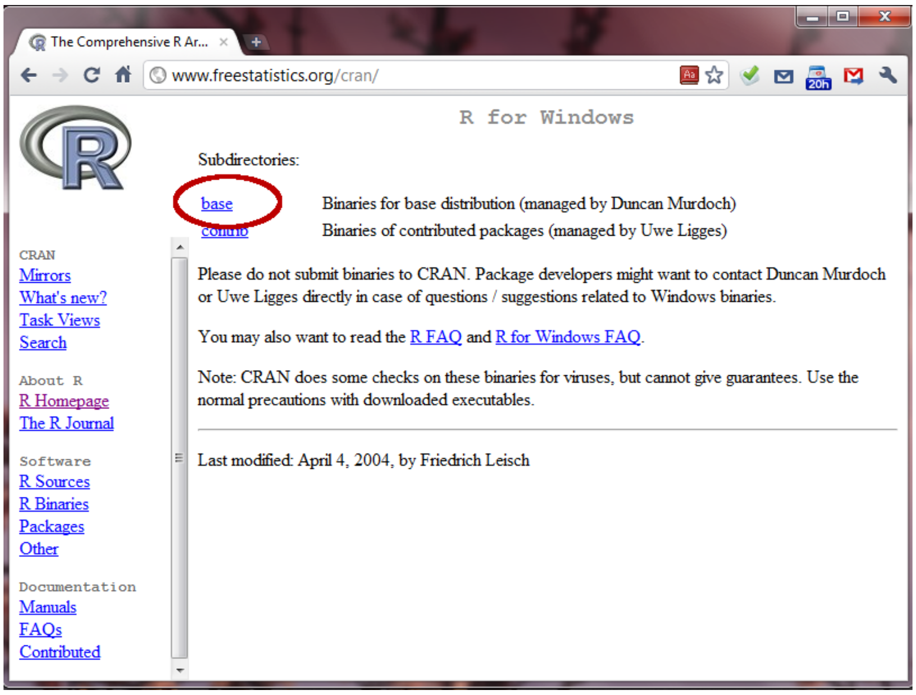
- Linux
- MacOS X
- Windows

Source Code for all Platforms

Windows and Mac users most likely want the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- **The latest release** (2010-12-16): R-2.12.1.tar.gz (read what's new in the latest version).
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R



R for Windows

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

Subdirectories:

base	Binaries for base distribution (managed by Duncan Murdoch)
contrib	Binaries of contributed packages (managed by Uwe Ligges)

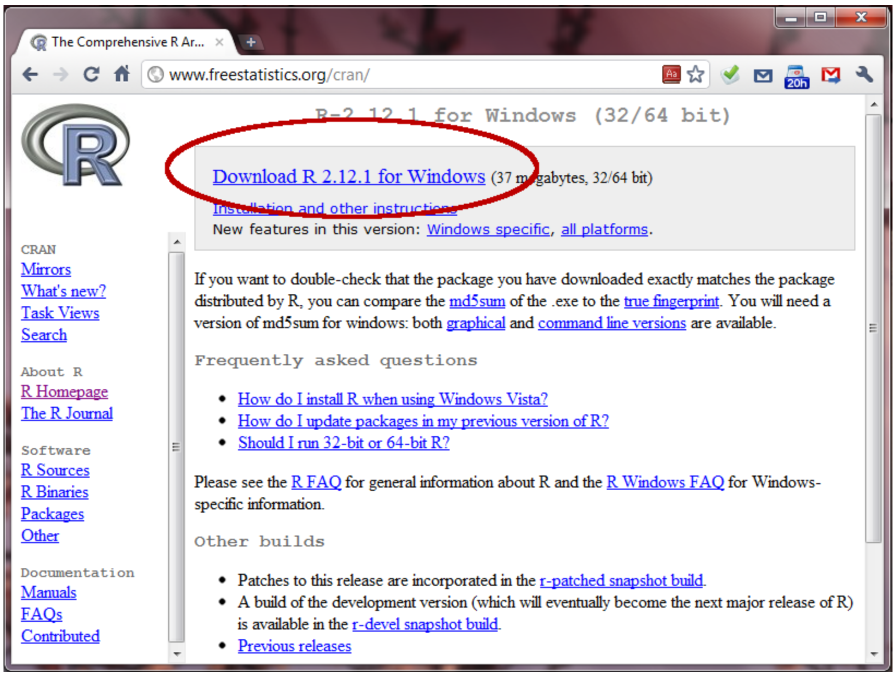
Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

Last modified: April 4, 2004, by Friedrich Leisch

If you choose “contrib”, you can download individual packages (see point 3)



R-2.12.1 for Windows (32/64 bit)

Download R 2.12.1 for Windows (37 megabytes, 32/64 bit)
Installation and other instructions
New features in this version: Windows specific, all platforms.

If you want to double-check that the package you have downloaded exactly matches the package distributed by R, you can compare the md5sum of the .exe to the true fingerprint. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

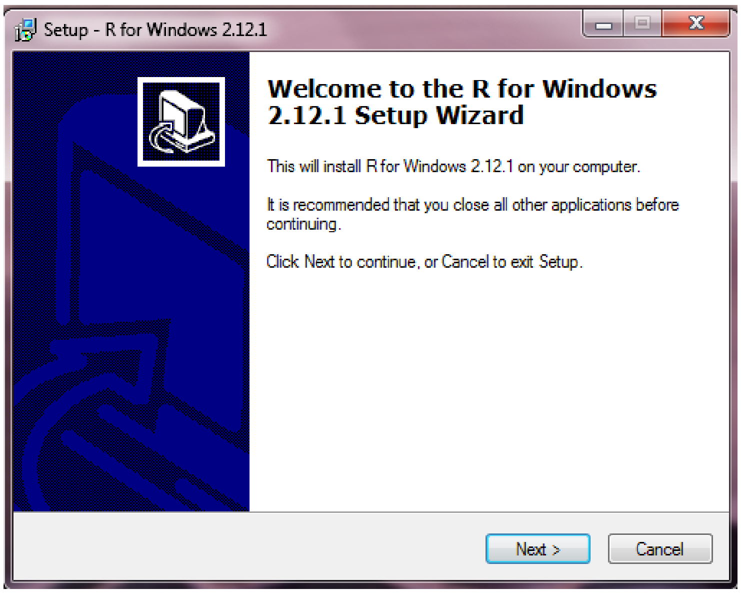
- How do I install R when using Windows Vista?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

2. Install the base system. As usually in Windows: double click and follow the Wizard installation.



Setup - R for Windows 2.12.1

Welcome to the R for Windows 2.12.1 Setup Wizard

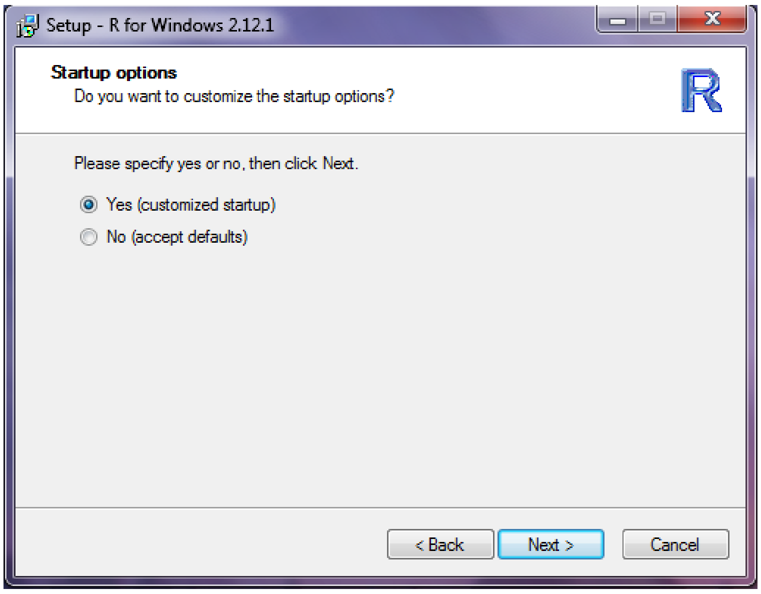
This will install R for Windows 2.12.1 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

It is recommended to select the SDI option when required, especially if you are planning to use **RCommander**[10] package.

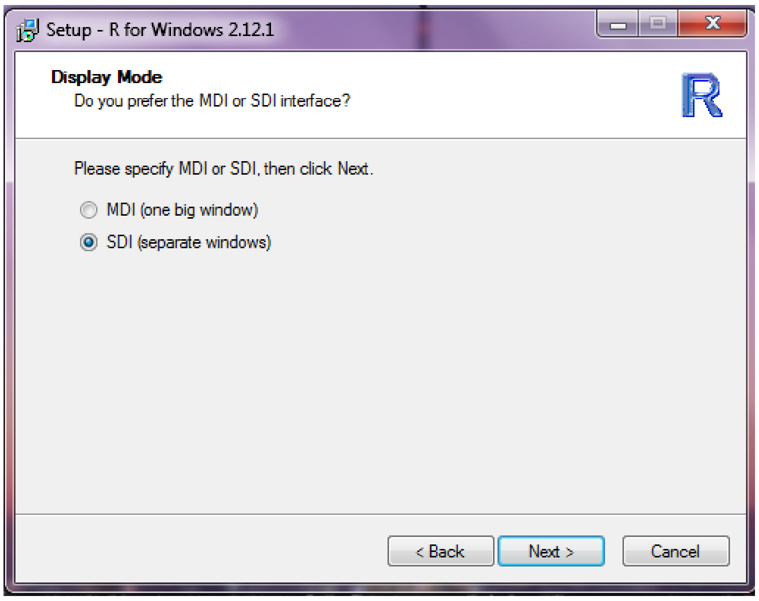
[10] John Fox <jfox@mcmaster.ca>, with contributions from Liviu Andronic, Michael Ash, Theophilius Boye, Stefano Calza, Andy Chang, Philippe Grosjean, Richard Heiberger, G. Jay Kerns, Renaud Lancelot, Matthieu Lesnoff, Uwe Ligges, Samir Messad, Martin Maechler, Robert Muenchen, Duncan Murdoch, Erich Neuwirth, Dan Putler, Brian Ripley, Miroslav Ristic and and Peter Wolf. (2011). Rcmdr: R Commander. R package version 1.6-3. http://CRAN.R-project.org/package=Rcmdr



Startup options
Do you want to customize the startup options?

Please specify yes or no, then click Next.

[✓ Yes (customized startup)]
No (accept defaults)

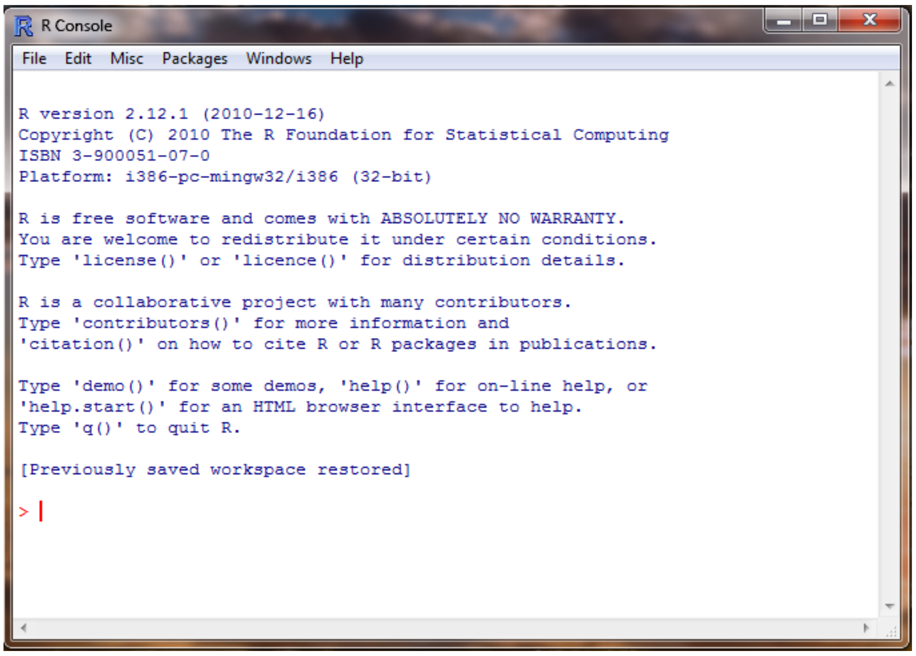


Display Mode
Do you prefer the MDI or SDI interface?

Please specify MDI or SDI, then click Next.

MDI (one big window)
[✓ SDI (separate windows)]

Once R is installed, we can open the program double-clicking the desktop icon, and R Console will start:

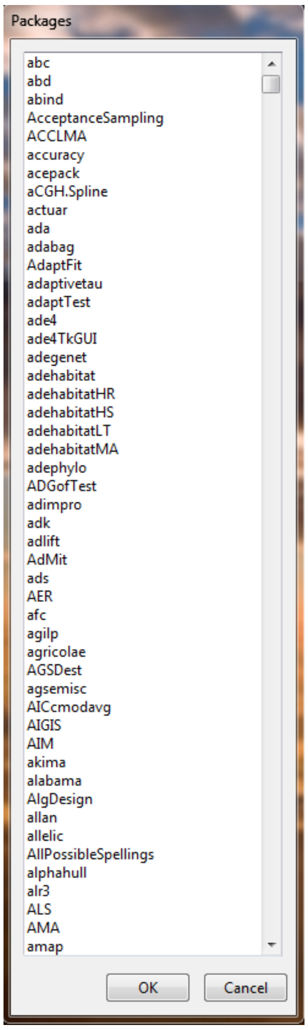


```
R version 2.12.1 (2010-12-16)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

>
```

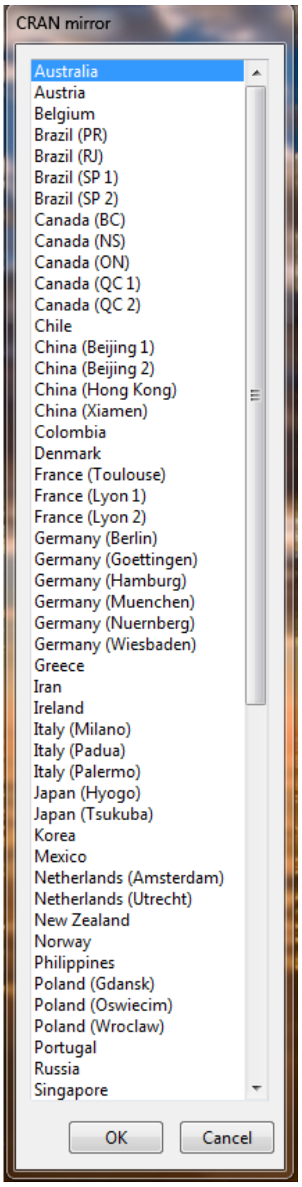
3. Install additional libraries. You can download them from CRAN (see point 1), or install from R interface (either commands or menus)

   With text commands, type (i.g.):
   install.packages("SixSigma", dependencies=TRUE)

   With menu commands, select Packages/Install Package(s) , select the name of the package you want to install, and click Ok.



Regardless the former or the latter method chosen, the first time in the session you try to download a package, you are requested to select a mirror. Simply select one close to your actual location as in step 1, and click Ok.



The time to download and install the package(s) depends on the size and the dependencies of the package.

Once the library is installed, you must type **`library(name_of_the_library)`** or select the menu command Packages/Load Package so as to be able to make use of the package.

4. Enjoy
   The funniest way to realize the power of R is by viewing demos. Type **`demo(graphics)`** the first time you open R Console.
   Install the **`lattice`** package as shown in step 3. Load the package and type **`demo(lattice)`**.

> Not all packages have demos, but every function must have its example. Type **`example(persp)`**.

Probably when you are reading this essay, there is a newer version of R, and maybe the snapshots are slightly different, but the process is very likely the same.

1.4.3. Basic use

The basic way we interact with R, is writing commands in the R Console. When you type a command, you get a response, like a text output or a graph. Some functions have no output, like the assign function.

We recommend typing the commands in a script window (File/New Script) or in RCommander[11] (**`library(Rcmdr)`**), in order to take control of your commands. Press CTRL+R to run the lines selected. You can also save your scripts, and load later.

With RCommander you have as well a graphic interface with some of the more used statistical functions, and a friendly way to manage data.

The code in this book can be straight executed in R Console, in order to get the output shown.

There is a considerable amount of publications about R. Some are referred in the Bibliography. You can start with the R-project on-line documentation and the packages documentation as well.

1.5. An approach to Six Sigma definition

In the American Society for Quality (ASQ) web page[12], we can find several definitions of Six Sigma, taken from (Donald W. Benbow, 2005):

> "Six Sigma is a fact-based, data-driven **philosophy** of quality improvement that values defect prevention over defect detection. It drives customer satisfaction and bottom-line results by reducing variation and waste, thereby promoting a competitive advantage. It applies anywhere variation and waste exist, and every employee should be involved.

[11] There is a basic guide here: http://socserv.socsci.mcmaster.ca/jfox/Misc/Rcmdr/
[12] http://asq.org/learn-about-quality/six-sigma/overview/overview.html

In simple terms, Six Sigma quality performance means no more than 3.4 defects per million opportunities.

Several different definitions have been proposed for Six Sigma, but they all share some common themes:

> Use of teams that are assigned well-defined projects that have direct impact on the organization's bottom line.
>
> Training in "**statistical thinking**" at all levels and providing key people with extensive training in advanced statistics and project management. These key people are designated black belts […]
>
> Emphasis on the **DMAIC** approach (define, measure, analyze, improve and control) to problem solving.
>
> A management environment that supports these initiatives as a business strategy.

Differing opinions on the definition of Six Sigma:

Six Sigma is a philosophy— This perspective views all work as processes that can be defined, measured, analyzed, improved and controlled. Processes require inputs (x) and produce outputs (y). If you control the inputs, you will control the outputs: This is generally expressed as $y = f(x)$.

Six Sigma is a set of tools— The Six Sigma expert uses qualitative and quantitative techniques to drive process improvement. A few such tools include statistical process control (SPC), control charts, failure mode and effects analysis and flowcharting.

Six Sigma is a methodology— This view of Six Sigma recognizes the underlying and rigorous approach known as DMAIC (define, measure, analyze, improve and control). DMAIC defines the steps a Six Sigma practitioner is expected to follow, starting with identifying the problem and ending with the implementation of long-lasting solutions. While DMAIC is not the only Six Sigma methodology in use, it is certainly the most widely adopted and recognized.

Excerpted from Donald W. Benbow and T. M. Kubiak, The Certified Six Sigma Black Belt Handbook, ASQ Quality Press, 2005, pages 1-2.”

2. Define

2.1. Process Map

With Minitab

There is no such functionality in Minitab© Statistical software. Nevertheless, you can buy other software from Minitab Inc: Quality Companion©. It has a complete graphic interface to insert all the information concerning the processes, input and output variables (x's, y's), and even Lean information.

With existing R

Though there is a package named **diagram**[13] than could plot some flow charts and use it for our purpose, provided that it was developed for botanical analysis, and has other functions that are not used for Six Sigma, I have developed a single function to perform this task, using **grid** package (Murrell, 2005)

With R new development

First of all, we need de data to represent. These data are character strings representing the process name and its components: inputs (X), outputs (Y), parameter inputs (x), parameter outputs (y) and their types. In our sample, we may have four processes: *inspection, assembly, test* and *labeling.*

```
> procs<-c("inspection", "assembly", "test", "labeling")
```

Each process has inputs and outputs. The process *i* inputs are process *i-1* outputs.

We represent inputs/outputs as a vector of strings with the inputs for *i* process.

```
> input.output<-vector(mode="list",
   length=length(procs))
> input.output[1]<-c("sheets")
> input.output[2]<-c("sheets")
> input.output[3]<-c("helicopter")
> input.output[4]<-c("helicopter")
```

[13] (Soetaert, 2009)

We have two more vectors: for overall process inputs, and overall process outputs.

```
> inputs.overall<-c("operators", "tools", "raw material", "facilities")
> outputs.overall<-c("helicopter")
```

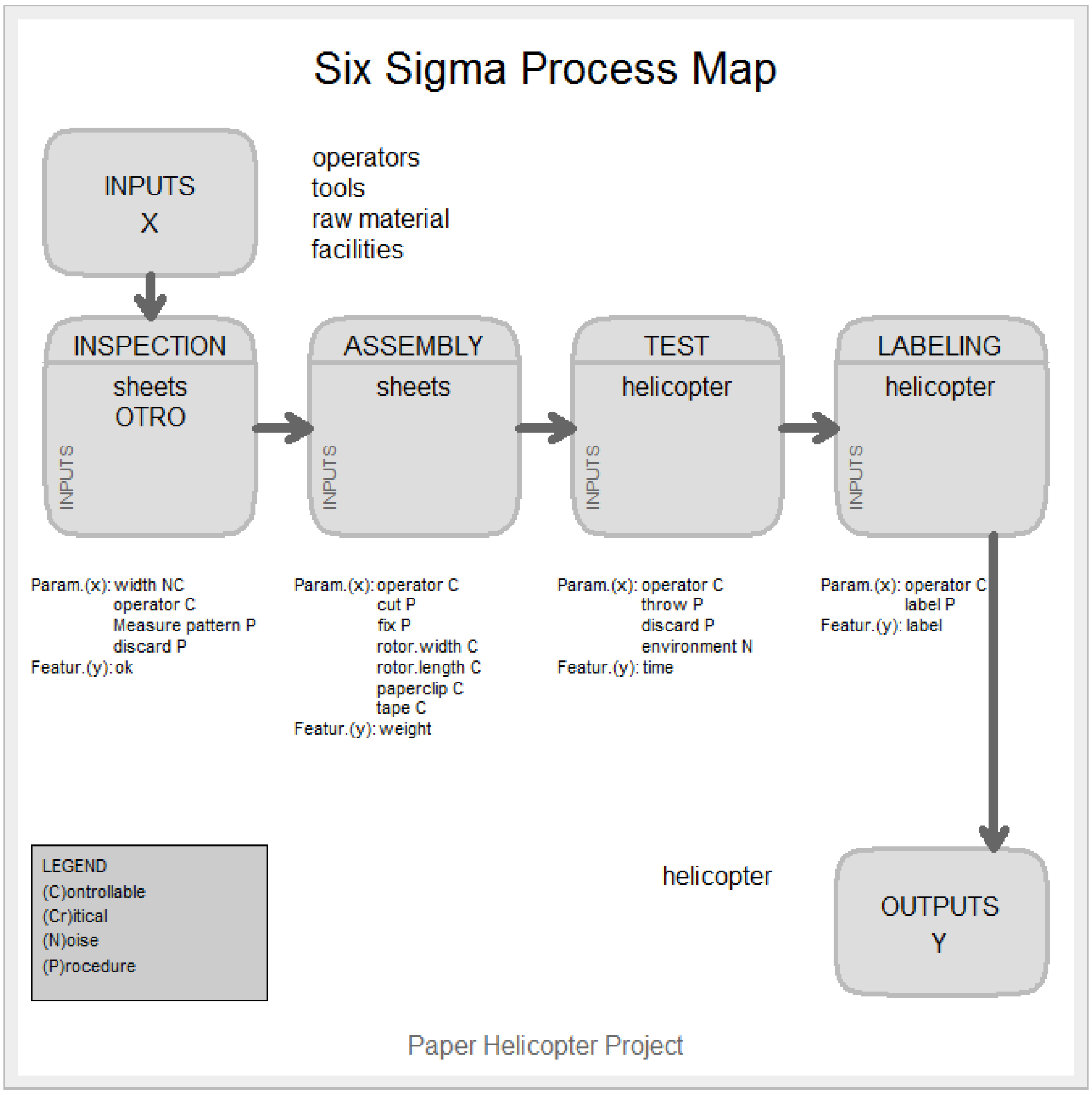
For the parameters (x's) and features (y's) of the process, we use vectors of lists with a list of the x parameters of the process. The parameter is a vector with two values: the name and the type. The type of the x parameters and y features can be: C(controllable), N(noise), Cr(Critical), P(Procedure)

```
> x.parameters<-vector(mode="list",length=length(procs))
> x.parameters[1]<-list(c(list(c("width", "NC")),list(c("operator", "C")),
   list(c("Measure pattern", "P")), list(c("discard", "P"))))
> x.parameters[2]<-list(c(list(c("operator", "C")),list(c("cut", "P")),
   list(c("fix", "P")), list(c("rotor.width", "C")),list(c("rotor.length",
   "C")), list(c("paperclip", "C")), list(c("tape", "C"))))
> x.parameters[3]<-list(c(list(c("operator", "C")),list(c("throw", "P")),
   list(c("discard", "P")), list(c("environment", "N"))))
> x.parameters[4]<-list(c(list(c("operator", "C")),list(c("label", "P"))))
> x.parameters
> y.features<-vector(mode="list",length=length(procs))
> y.features[1]<-list(c(list(c("ok", "Cr"))))
> y.features[2]<-list(c(list(c("weight", "Cr"))))
> y.features[3]<-list(c(list(c("time", "Cr"))))
> y.features[4]<-list(c(list(c("label", "Cr"))))
```

Now we can call the function:

```
> ss.pMap(procs, inputs.overall, outputs.overall, input.output, x.parameters,
   y.features, sub="Paper Helicopter Project")
```

And get this chart as a result:

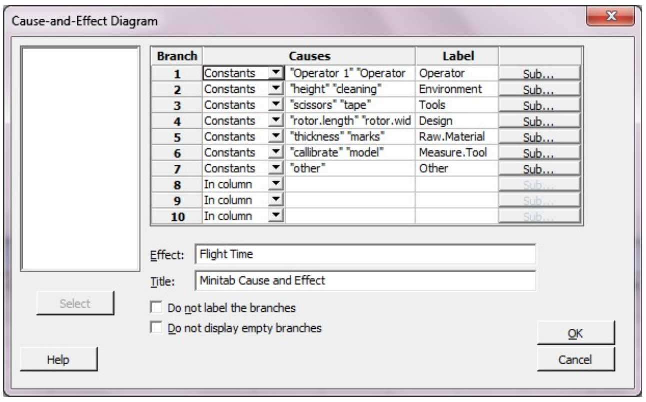


Graph 1: Six Sigma Process Map

2.2. Cause-and-effect Diagram

With Minitab

This diagram can be shown in Minitab with the command *Cause-and-effect…*, under the menu *Stat/Quality tools*. You can write the causes interactively in a dialog box, or take from the data sheet of Minitab.



Cause-and-Effect Diagram

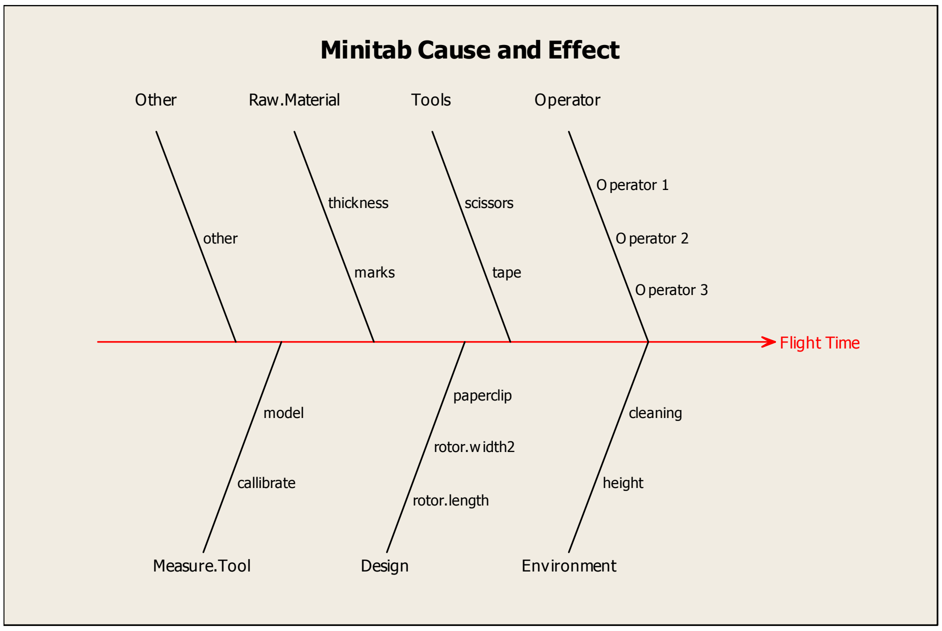
Branch		Causes	Label	
1	Constants	"Operator 1" "Operator	Operator	Sub…
2	Constants	"height" "cleaning"	Environment	Sub…
3	Constants	"scissors" "tape"	Tools	Sub…
4	Constants	"rotor.length" "rotor.wid	Design	Sub…
5	Constants	"thickness" "marks"	Raw.Material	Sub…
6	Constants	"callibrate" "model"	Measure.Tool	Sub…
7	Constants	"other"	Other	Sub…
8	In column			Sub…
9	In column			Sub…
10	In column			Sub…

Effect: Flight Time

Title: Minitab Cause and Effect

☐ Do not label the branches

☐ Do not display empty branches

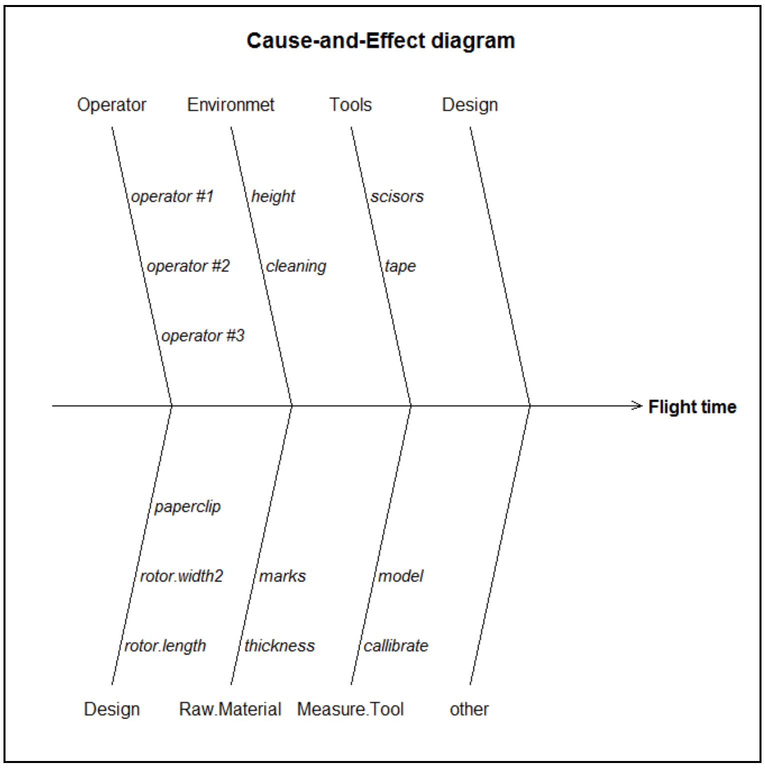


Graph 2; Minitab Cause and Effect diagram

With existing R

The existing package `qcc` (Scrucca, 2004) in **R** can show a Cause-and-Effect diagram with the function cause.and.effect:

```
> cause.and.effect(
   cause=list(Operator=
   c("operator #1", "operator #2", "operator #3"), Environmet=c("height",
   "cleaning"),                                Tools=c("scisors", "tape"),
   Design=c("rotor.length", "rotor.width2", "paperclip"),
   Raw.Material=c("thickness", "marks"), Measure.Tool=c("callibrate",
   "model"),                                other=c("other")),
   effect="Flight time")
```



Graph 3: qcc package Cause-and-Effect diagram

With new R development

I have developed a new function in order to improve de existing Diagram.

The effect is a simple string of characters:

```
> effect<-"Flight Time"
```

We use again vectors and lists for the data. First, the groups of causes:

```
> causes.gr<-c("Operator", "Environment", "Tools", "Design", "Raw.Material",
   "Measure.Tool", "test")
```

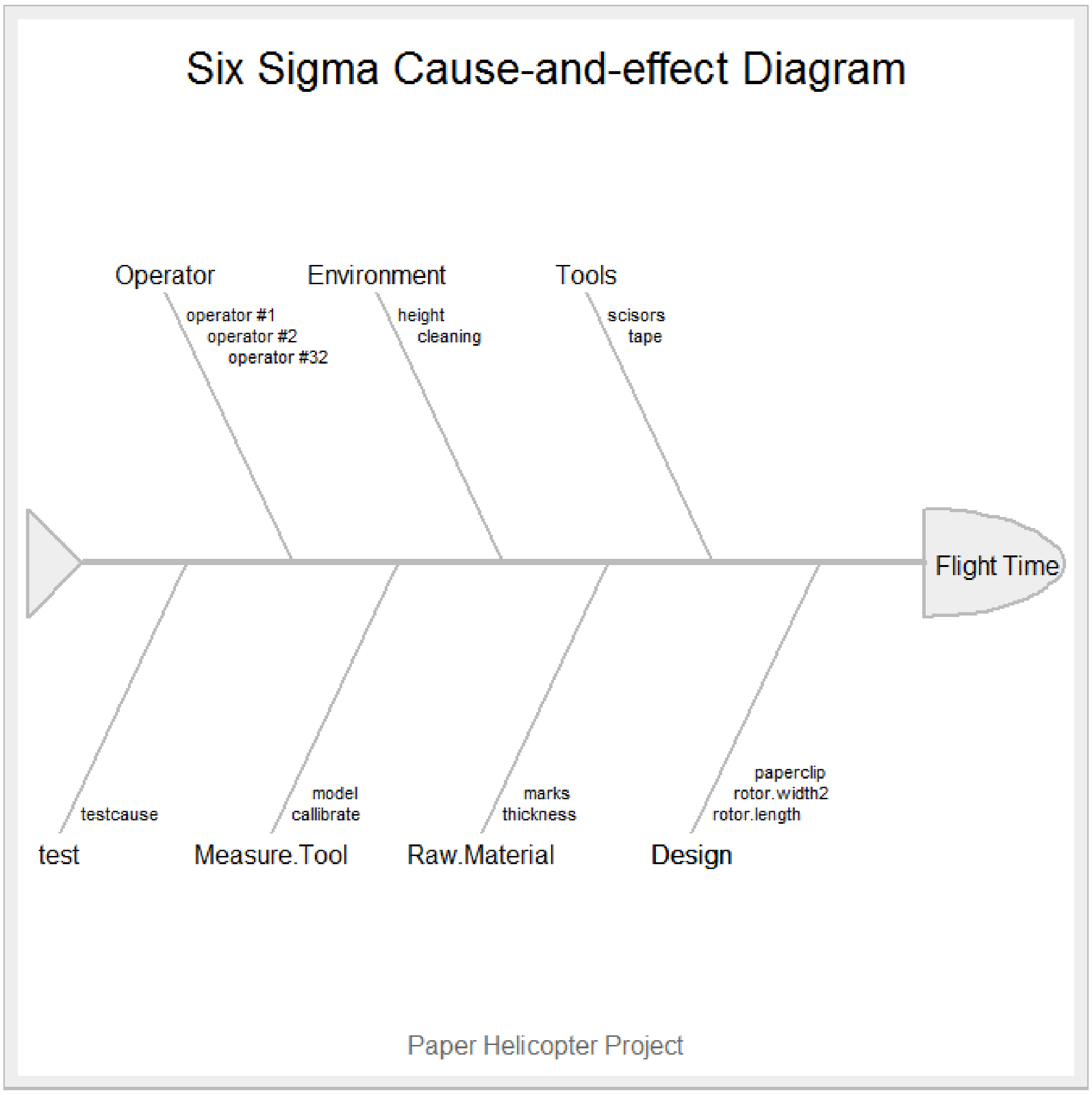
And finally, the individual causes, as e vector of lists (one for each group):

```
> causes<-vector(mode="list", length=length(causes.gr))
> causes[1]<-list(c("operator #1", "operator #2", "operator #32"))
> causes[2]<-list(c("height", "cleaning"))
> causes[3]<-list(c("scisors", "tape"))
> causes[4]<-list(c("rotor.length", "rotor.width2", "paperclip"))
> causes[5]<-list(c("thickness", "marks"))
> causes[6]<-list(c("callibrate", "model"))
> causes[7]<-list(c("testcause"))
```

Now we can call the function:

```
> ss.pMap(procs, inputs.overall, outputs.overall, input.output, x.parameters,
   y.features, sub="Paper Helicopter Project")
```

And get the diagram shown in Graph 4.



Graph 4: Cause and effect diagram

Appendix 2A: R functions Code

Process Map (ss.pMap)

```
ss.pMap <-
function(procs, inputs.overall, outputs.overall,
                 input.output, x.parameters, y.features,
        main="Six Sigma Process Map", sub,
        palette=c("#666666", "#BBBBBB", "#CCCCCC", "#DDDDDD", "#EEEEEE")){
     nprocs<-length(procs)
     paintBox<-function(){
      x<-c(rep(0.10,4),0.30,0.70,rep(0.90,4),0.7,0.3)
      y<-c(0.10,0.37,0.64,rep(0.91,4),0.64,0.37,rep(0.1,3))
      grid.xspline(x, y,
           shape=c(1,1,1,1,1,1),open=FALSE,
           gp=gpar(fill=palette[4],lwd=3, col=palette[2])
           )
     }

     grid.newpage()
     grid.rect(gp=gpar(col=palette[2], lwd=2, fill=palette[5]))
     vp.canvas<-viewport(width=unit(1,"npc")-unit(5,"mm"),
      height=unit(1,"npc")-unit(5,"mm"),
      name="canvas",
           layout=grid.layout(3,3,
           widths=unit(c(1,1,1), c("lines", "null", "lines")),
           heights=unit(c(3,1,2), c("lines", "null", "lines"))
           )
      )
     pushViewport(vp.canvas)
     grid.rect(gp=gpar(col="#FFFFFF", lwd=0, fill="#FFFFFF"))
#Title
     vp.title<-viewport(layout.pos.col=1:3, layout.pos.row=1, name="title")
     pushViewport(vp.title)
     grid.text(main, name="titulo",   gp=gpar(fontsize=20))
     popViewport()

#Subtitle
     vp.subtitle<-viewport(layout.pos.col=1:3, layout.pos.row=3,
     name="subtitle")
     pushViewport(vp.subtitle)
     grid.text ( sub, gp=gpar(col=palette[1]))
     popViewport()

#ProcessMap container
     vp.map<-viewport(name="map",layout.pos.col=1:3, layout.pos.row=2,
      layout=grid.layout(3,nprocs, heights=c(0.2,0.6,0.2)))     ##
     pushViewport(vp.map)

#overall inputs
     vp.inputs<-viewport(layout.pos.col=1,layout.pos.row=1, name="inputs")
     pushViewport(vp.inputs)
     paintBox()
     grid.text("INPUTS\nX")
```

```
	grid.move.to(x=0.5, y=0.1)
	upViewport()
	vp.inputsText<-viewport(layout.pos.col=2:4, layout.pos.row=1,
	name="inputst")
	pushViewport(vp.inputsText)
	for (i in 1:length(inputs.overall)){
	 grid.text(x=unit(0.5,"cm"),
	 y=unit(1, "npc")-unit(i,"lines"),
	 paste(inputs.overall[i],"\n"),
	 just=c("left","top"), name="inputst")
	}
	upViewport()

#Processes
	for (i in 1:nprocs){
	 vp.proc<-viewport(layout.pos.col=i, layout.pos.row=2)
	 pushViewport(vp.proc)
	 pushViewport(viewport(y=1, h=0.5, just=c("center","top")))
	 paintBox()
	 grid.lines(x=c(0.1,0.9), y=c(0.74, 0.74), gp=gpar(lwd=3,
	col=palette[2]))
	 grid.text(procs[i], y=0.85, just=c("center", "top"))
	 grid.text("INPUTS", rot=90, x=0.20, y=0.20,
		just=c("left", "bottom"), gp=gpar(fontsize=8, col=palette[1]))
	 for (j in 1:length(input.output[[i]])){
		grid.text(input.output[[i]][j], y=unit(0.7, "npc")-unit(j-
	1,"lines"),
			just=c("center","top"))
	 }
	 if (i==1){
		grid.line.to(x=0.5, y=0.91,
			arrow=arrow(angle = 30, length = unit(0.15, "inches"),
			ends = "last", type = "open"), gp=gpar(lwd=6,
	col=palette[1]))
	 }
	 if (i>1){
		grid.line.to(x=0.1, y=0.5,
		arrow=arrow(angle = 30, length = unit(0.15, "inches"),
		ends = "last", type = "open"), gp=gpar(lwd=6, col=palette[1]))
	 }
	 grid.move.to(x=0.9, y=0.5)
	 if (i==nprocs){
		grid.move.to(x=0.7, y=0.1)
	 }
	 upViewport()
	 pushViewport(viewport(y=0.5, h=0.5, just=c("center","top")))
	 grid.text("Param.(x): ", y=unit(1,"npc")-unit(2,"mm"),
	gp=gpar(fontsize=8),
		x=unit(0,"npc")+unit(2,"mm"),
		just=c("left", "top"))
	 for (j in 1:length(x.parameters[[i]])){

	 grid.text(paste(x.parameters[[i]][[j]][1],x.parameters[[i]][[j]][2]),
			y=unit(1,"npc")-unit(2,"mm")-unit(j-1,"lines"),
			x=unit(1,"strwidth","Param.(x):   "),
			gp=gpar(fontsize=8), just=c("left", "top"))
	 }
	 grid.text("Featur.(y): ",
            y=unit(1,"npc")-unit(2,"mm")-
      unit(length(x.parameters[[i]]),"lines"),
            x=unit(0,"npc")+unit(2,"mm"),
            gp=gpar(fontsize=8),
            just=c("left", "top"))
       for (j in 1:length(y.features[[i]])){
            grid.text(y.features[[i]][[j]],
                 y=unit(1,"npc")-unit(2,"mm")-unit(j-
      1+length(x.parameters[[i]]),
                      "lines"),
                 x=unit(1,"strwidth","Featur.(y):   "),
                 gp=gpar(fontsize=8), just=c("left", "top"))
       }
       upViewport()
       upViewport()
      }

#overalloutputs
      vp.outputs<-viewport(layout.pos.col=nprocs, layout.pos.row=3,
      name="outputs")
      pushViewport(vp.outputs)
      paintBox()
      grid.text("OUTPUTS\nY")
      grid.line.to(x=0.7, y=0.91, arrow=arrow(angle = 30, length = unit(0.15,
      "inches"),
       ends = "last", type = "open"), gp=gpar(lwd=6, col=palette[1]))
      upViewport()
      vp.outputsText<-viewport(layout.pos.col=1:3, layout.pos.row=3,
      name="outputst")
      pushViewport(vp.outputsText)
      for (i in 1:length(outputs.overall)){
       grid.text(x=unit(1,"npc")-unit(0.5,"cm"),
            y=unit(1, "npc")-unit(i,"lines"),
      paste(outputs.overall[i],"\n"),
            just=c("right","top"), name="outputst")
      }
      vp.legend<-viewport(x=unit(0.2, "cm"),
       y=unit(0.2,"cm"),
       just=c("left","bottom"),
       height=unit(1,"npc")-unit(0.4, "cm"),
       width=0.3)
      pushViewport(vp.legend)
      grid.rect(gp=gpar(fill=palette[3]))
      grid.text("LEGEND\n\t(C)ontrollable\n\t(Cr)itical\n\t(N)oise\n\t(P)roced
      ure",
       y=unit(1, "npc")-unit(0.2, "cm") , x=unit(0, "npc")+unit(0.2,"cm"),
       just=c("left", "top"),
       gp=gpar(fontsize=8))
      upViewport()
      upViewport()
}
```

Cause and effect diagram (ss.ceDiag)

```
ss.ceDiag<-function(effect, causes.gr, causes, main="Six Sigma Cause-and-
      effect Diagram", sub, palette=c("#666666", "#BBBBBB", "#CCCCCC",
      "#DDDDDD", "#EEEEEE")){
    n.causes<-length(causes.gr)
```

```
     grid.newpage()
     grid.rect(gp=gpar(col=palette[2], lwd=2, fill=palette[5]))
     vp.canvas<-viewport(width=unit(1,"npc")-
     unit(5,"mm"),height=unit(1,"npc")-unit(5,"mm"),
      name="canvas",
      layout=grid.layout(3,3,
            widths=unit(c(1,1,1), c("lines", "null", "lines")),
            heights=unit(c(3,1,2), c("lines", "null", "lines"))
      ))
     pushViewport(vp.canvas)
     grid.rect(gp=gpar(col="#FFFFFF", lwd=0, fill="#FFFFFF"))

#Title
     vp.title<-viewport(layout.pos.col=1:3, layout.pos.row=1, name="title")
     pushViewport(vp.title)
     grid.text (main, name="titulo", gp=gpar(fontsize=20))
     popViewport()

#Subtitle
     vp.subtitle<-viewport(layout.pos.col=1:3, layout.pos.row=3,
     name="subtitle")
     pushViewport(vp.subtitle)
     grid.text ( sub, gp=gpar(col=palette[1]))
     popViewport()

#ProcessMap container
     vp.diagram<-viewport(name="diagram",layout.pos.col=1:3,
     layout.pos.row=2)
     pushViewport(vp.diagram)

#Fish head
     w.head<-unit(1,"strwidth", effect)+unit(4,"mm")
     vp.head<-viewport(x=unit(1,"npc")-w.head,
     h=1, w=w.head, just=c("left", "center"))
     pushViewport(vp.head)

     x.hshape<-c(0.00,0.00,0.50,0.93,0.95,0.93,0.50)
     y.hshape<-c(0.44,0.56,0.55,0.52,0.50,0.48,0.45)
     grid.xspline(x.hshape, y.hshape, shape=c(0,0,-1,1,0,1,-1), open=FALSE,
      gp=gpar(col=palette[2],lwd=2, fill=palette[5]))
     grid.text(effect)
     popViewport()

#Fish tail
     vp.tail<-viewport(x=0.01, h=1, w=0.05, just=c("left","center"))
     pushViewport(vp.tail)
     grid.xspline(x=c(0,0,1), y=c(0.44,0.56,0.5), shape=c(0,0,0), open=FALSE,
      gp=gpar(col=palette[2],lwd=2, fill=palette[5]))
     popViewport()
     vp.body<-viewport(x=0.06,
      h=1, w=unit(1,"npc")-w.head-unit(0.06,"npc"), just=c("left",
     "center"))
     pushViewport(vp.body)
     grid.lines(x=c(0,1), y=c(0.5,0.5),
      gp=gpar(col=palette[2],lwd=4))
     room<-1/floor((n.causes+1)/2)
     up<-seq(room,1, room)
     for (i in 1:(length(up)-1)){
      grid.lines(x=c(up[i]-0.15, up[i] ), y=c(0.8,0.5),
           gp=gpar(col=palette[2],lwd=2))
      grid.text(causes.gr[i], x=up[i]-0.15, y=0.81, just=c("center",
     "bottom"))
      for (j in 1:length(causes[[i]])){
            grid.text(causes[[i]][j], x=unit(up[i]-
     0.15,"npc")+unit(j,"lines"),
                   y=unit(0.80, "npc")-unit(j,"lines"), gp=gpar(fontsize=8),
                   just=c("left","center"))
             }
      }
     x=as.vector(c(unit(1, "npc")-unit(room/2, "npc")-unit(0.15,"npc"),
      unit(1, "npc")-unit(room/2,"npc"))
      )
     for (i in length(up):n.causes){
      if (i==length(up)){
             grid.lines(x=c(1-room/2-0.15,1-room/2),
                   y=c(0.2,0.5),            gp=gpar(col=palette[2],lwd=2))
             grid.text(causes.gr[i], x=1-room/2-0.15, y=0.19,
     just=c("center", "top"))
      }
      grid.lines(x=c((1-(room+(2*(i-4)*room))/2)-0.15,1-(room+(2*(i-
     4)*room))/2 ),
             y=c(0.2,0.5),
             gp=gpar(col=palette[2],lwd=2))
      grid.text(causes.gr[i], x=(1-(room+(2*(i-4)*room))/2)-0.15, y=0.19,
     just=c("center", "top"))
      for (j in 1:length(causes[[i]])){
             grid.text(causes[[i]][j],
                   x=unit((1-(room+(2*(i-4)*room))/2)-0.15,
     "npc")+unit(j,"lines"),
                   y=unit(0.20, "npc")+unit(j,"lines"), gp=gpar(fontsize=8),
                   just=c("left","center"))
      }
     }
}
```

Appendix 2B: Data tables

ss.pMap

```
> procs
[1] "INSPECTION" "ASSEMBLY"   "TEST"       "LABELING"
> inputs.overall
[1] "operators"    "tools"        "raw material" "facilities"
> outputs.overall
[1] "helicopter"
> input.output
[[1]]
[1] "sheets" "OTRO"

[[2]]
[1] "sheets"

[[3]]
[1] "helicopter"

[[4]]
[1] "helicopter"

> x.parameters
[[1]]
[[1]][[1]]
[1] "width" "NC"

[[1]][[2]]
[1] "operator" "C"

[[1]][[3]]
[1] "Measure pattern" "P"

[[1]][[4]]
[1] "discard" "P"


[[2]]
[[2]][[1]]
[1] "operator" "C"

[[2]][[2]]
[1] "cut" "P"

[[2]][[3]]
[1] "fix" "P"

[[2]][[4]]
[1] "rotor.width" "C"

[[2]][[5]]
[1] "rotor.length" "C"
```

```
[[2]][[6]]
[1] "paperclip" "C"

[[2]][[7]]
[1] "tape" "C"


[[3]]
[[3]][[1]]
[1] "operator" "C"

[[3]][[2]]
[1] "throw" "P"

[[3]][[3]]
[1] "discard" "P"

[[3]][[4]]
[1] "environment" "N"


[[4]]
[[4]][[1]]
[1] "operator" "C"

[[4]][[2]]
[1] "label" "P"


> y.features
[[1]]
[[1]][[1]]
[1] "ok" "Cr"


[[2]]
[[2]][[1]]
[1] "weight" "Cr"


[[3]]
[[3]][[1]]
[1] "time" "Cr"


[[4]]
[[4]][[1]]
[1] "label" "Cr"
```

ss.ceDiag

```
> effect
[1] "Flight Time"
> causes.gr
[1] "Operator"     "Environment"  "Tools"        "Design"
     "Raw.Material"
[6] "Measure.Tool" "test"
```

```
> causes
[[1]]
[1] "operator #1"  "operator #2"  "operator #32"

[[2]]
[1] "height"   "cleaning"

[[3]]
[1] "scisors" "tape"

[[4]]
[1] "rotor.length" "rotor.width2" "paperclip"

[[5]]
[1] "thickness" "marks"

[[6]]
[1] "callibrate" "model"

[[7]]
[1] "testcause"
```

Appendix 2C: Functions help

ss.pMap

```
Process Map

Description:

     This function takes information about the process we want to
     represent and draw the Map Process, whith its X's, x's, Y's and
     y's

Usage:

     ss.pMap(procs, inputs.overall, outputs.overall, input.output,
     x.parameters, y.features, main = "Six Sigma Process Map", sub, palette =
     c("#666666", "#BBBBBB", "#CCCCCC", "#DDDDDD", "#EEEEEE"))

Arguments:

   procs: A vector of character with the name of the 'n' processes

inputs.overall: A vector of characters with the name of the overall
          inputs

outputs.overall: A vector of characters with the name of the overall
          outputs

input.output: A vector of lists with the names of the inputs of the
          'i-th' process, that will be the outputs of the (i-1)th
          process

x.parameters: A vector of lists with a list of the x parameters of the
          process. The parameter is a vector with two values: the name
          and the type (view details)

y.features: A vector of lists with a list of the y features of the
          process. The feature is a vector with two values: the name
          and the type (view details)

    main: The main title for the Proecess Map

     sub: Subtitle for the diagram (recommended the Six Sigma project
          name)

 palette: A vector of colors for a personalized drawing. At least five
          colors, sorted by descendant intensity (see details)

Details:

     The type of the x parameters and y features can be:
     C(controllable), N(noise), Cr(Critical), P(Procedure). The default
     value for palette is c("#666666", "#BBBBBB", "#CCCCCC", "#DDDDDD",
     "#EEEEEE"), a grayscale style. You can pass any accepted color
     string.

Value:

     A graphic representation of the Map Process.

Note:

     The process map is the starting point for a Six Sigma Project, and
     it is very important to find out who the x's and y'x are.
```

ss.ceDiag

```
Cause and Effect Diagram

Description:

     Represents a Cause and Effect Diagram by cause group

Usage:

     ss.ceDiag(effect, causes.gr, causes,
     main = "Six Sigma Cause-and-effect Diagram", sub, palette = c("#666666",
     "#BBBBBB", "#CCCCCC", "#DDDDDD","#EEEEEE"))

Arguments:

  effect: A short character string that represents the effect we want
          to analyse.

causes.gr: A vector of characters that represents the causes groups.

  causes: A vector with lists that represents the individual causes for
          each causes group.

    main: Main title for the diagram

     sub: Subtitle for the diagram (recommended the Six Sigma project
          name)

 palette: A vector of colors for a personalized drawing. At least five
          colors, sorted by descendant intensity

Details:

     The default value for palette is c("#666666", "#BBBBBB",
     "#CCCCCC", "#DDDDDD", "#EEEEEE"), a grayscale style. You can pass
     any accepted color string.

Value:

     A drawing of the causes and efect with "fishbone" shape

Note:

     The cause and effect diagram is knowed also as "Ishikawa diagram",
     and has been widely used in Quality Management. It is one of the
     Seven Basic Tools of Quality.
```

3. Measure

3.1. Gage R&R

With Minitab

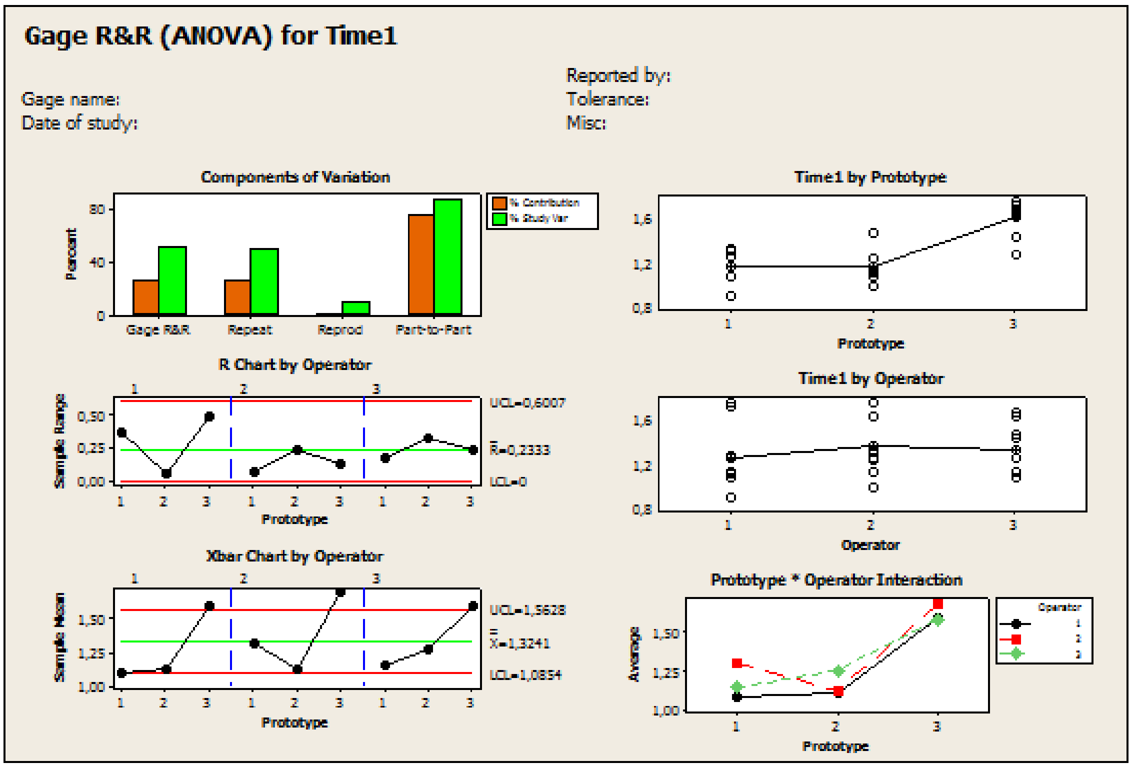
To perform a Gage R&R cross study with Minitab, you can go easily through the menu bar:

Stat / Quality Tools / Gage Study / Gage R&R Study (Cross).

You must select the Part Numbers, Operators and Measurement data columns of the data sheet.

Many options can be configured as well, such as Method of Analysis, spec limits, whether include confidence interval or not, titles, and so forth.

Both the graphic and the text output are:



Graph 5: Minibab Gage R&R Graph

Gage R&R Study - ANOVA Method

Two-Way ANOVA Table With Interaction

```
Source                DF       SS        MS        F      P
Prototype              2  1,20072  0,600359  28,7968  0,004
Operator               2  0,05294  0,026470   1,2697  0,374
Prototype * Operator   4  0,08339  0,020848   0,9737  0,446
Repeatability         18  0,38540  0,021411
Total                 26  1,72245


Alpha to remove interaction term = 0,5
```

Gage R&R

```
                                     %Contribution
Source                    VarComp   (of VarComp)
Total Gage R&R          0,0220358          25,50
  Repeatability         0,0214111          24,77
  Reproducibility       0,0006247           0,72
    Operator            0,0006247           0,72
    Operator*Prototype  0,0000000           0,00
Part-To-Part            0,0643901          74,50
Total Variation         0,0864259         100,00


                                     Study Var  %Study Var
Source                  StdDev (SD)  (5,15 * SD)       (%SV)
Total Gage R&R             0,148445      0,76449       50,49
  Repeatability            0,146325      0,75358       49,77
  Reproducibility          0,024994      0,12872        8,50
    Operator               0,024994      0,12872        8,50
    Operator*Prototype     0,000000      0,00000        0,00
Part-To-Part               0,253752      1,30682       86,32
Total Variation            0,293983      1,51401      100,00


Number of Distinct Categories = 2
```

Gage R&R for Time1

With existing R

We can perform individually all the tasks to obtain the results we are searching.

We can use the base functions: **lm** and **anova** for the first analysis. Then make computations to obtain the correct data to represent, and finally use graphics functions: **barplot** and **plot**. We can use as well the package **qcc** for the control charts, but transforming data for R&R purposes.

I have put all together in a function for **SixSigma** package.

With R new development

Let's use some example data in **SixSigma** package (SixSigma package must be loaded)

```
> data(ss.data.rr)
```

The data frame (see appendix 3C) has 3 measures for each of the 3 operators on 3 prototypes of our paper helicopter. The variable time1 is the first measure, and time2 is the measure after operators training. (type **?ss.data.rr** for details)

For the ANOVA we use **lm** and **anova** functions.

```
> anova(lm(time1~prototype+operator+
   prototype*operator, data=ss.data.rr))
```

```
Analysis of Variance Table

Response: time1
                   Df  Sum Sq Mean Sq F value    Pr(>F)
prototype           2 1.20072 0.60036 28.0396 2.952e-06 ***
operator            2 0.05294 0.02647  1.2363    0.3140
prototype:operator  4 0.08339 0.02085  0.9737    0.4462
Residuals          18 0.38540 0.02141
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

But we must make some computations to get the sources of variation[14]. First, we save the model:

```
> model(lm(time1~prototype+operator+
   prototype*operator, data=ss.data.rr))
```

And then we save data in a matrix:

```
attach(ss.data.rr)
a<- length(levels(prototype))
b<-length(levels(operator))
n<-length(time1)/(a*b)
varComp<-matrix(ncol=5,nrow=7)
rownames(varComp)<-c("Total Gage R&R", "  Repeatability", "
   Reproducibility", paste("   ", rownames(model)[2]),
paste("   ",rownames(model)[3]), "Part-To-Part", "Total Variation")
colnames(varComp)<-c("VarComp","%Contrib","StdDev", "5,15*SD","%StudyVar")
varComp[2,1]<-model[4,3]
varComp[4,1]<-(model[2,3]-model[3,3])/(a*n)
varComp[5,1]<-max(c((model[3,3]-model[4,3])/n,0))
varComp[3,1]<-varComp[4,1]+varComp[5,1]
varComp[6,1]<-(model[1,3]-model[3,3])/(b*n)
```

[14] See methodological notes at Appendix 3A

```
varComp[1,1]<-varComp[2,1]+varComp[3,1]
varComp[7,1]<-varComp[1,1]+varComp[6,1]
varComp[,1]<-round(varComp[,1],7)
varComp[,2]<-round(100*(varComp[,1]/varComp[7,1]),2)
varComp[,3]<-sqrt(varComp[,1])
varComp[,4]<-varComp[,3]*5.15
varComp[,5]<-round(100*(varComp[,4]/varComp[7,4]),2)
```

This is the result:

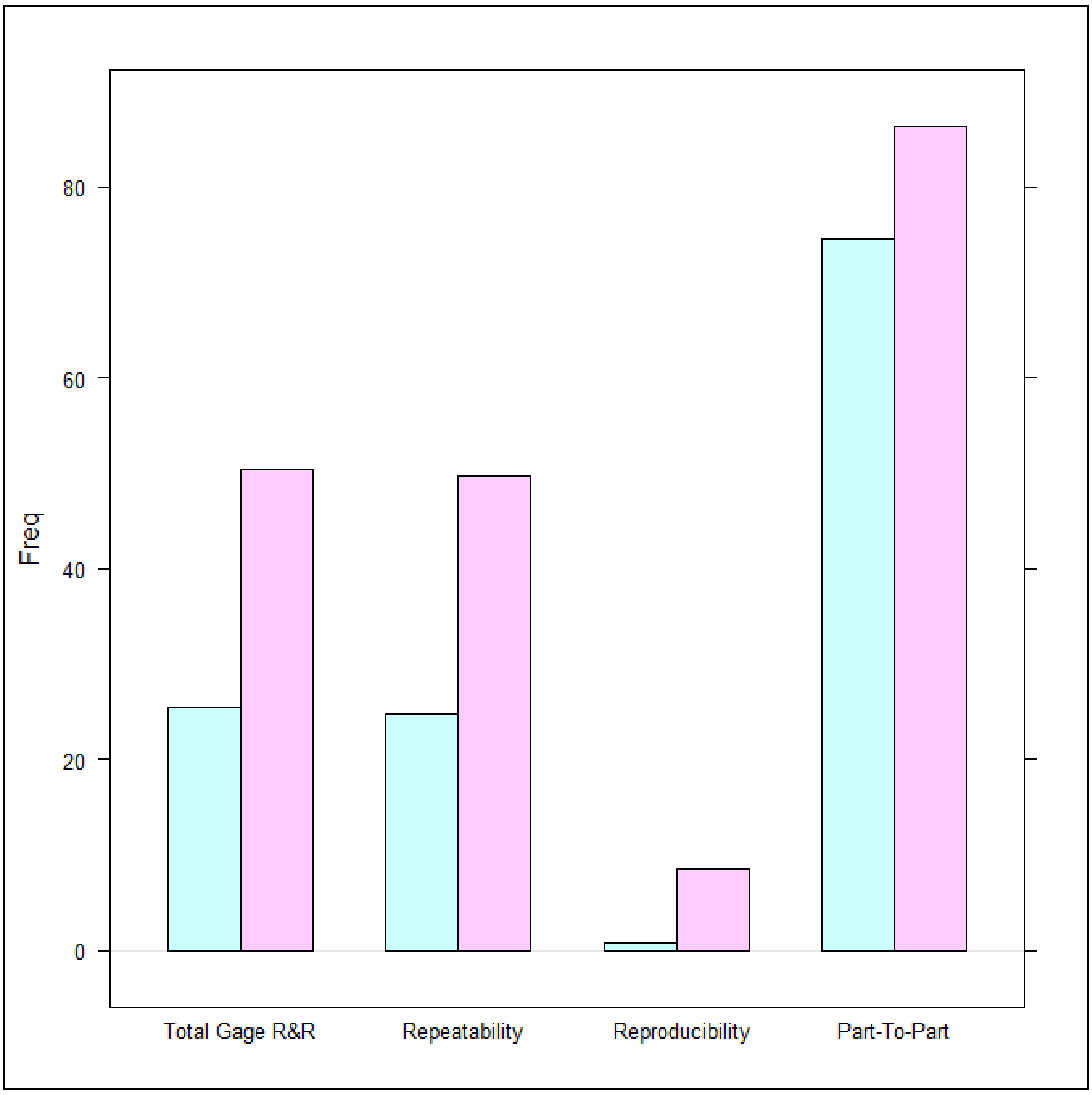
```
> varComp[,1:2]
                        VarComp %Contrib
Total Gage R&R        0.0220358    25.50
  Repeatability       0.0214111    24.77
  Reproducibility     0.0006247     0.72
    operator          0.0006247     0.72
    prototype:operator 0.0000000     0.00
Part-To-Part          0.0643901    74.50
Total Variation       0.0864259   100.00
> varComp[,3:5]
                         StdDev   5,15*SD %StudyVar
Total Gage R&R        0.1484446 0.7644897     50.49
  Repeatability       0.1463253 0.7535754     49.77
  Reproducibility     0.0249940 0.1287191      8.50
    operator          0.0249940 0.1287191      8.50
    prototype:operator 0.0000000 0.0000000      0.00
Part-To-Part          0.2537520 1.3068230     86.32
Total Variation       0.2939828 1.5140115    100.00
```

The number of distinct categories is:

```
> max(c(1,floor((varComp[6,3]/varComp[1,3])*1.41)))
[1] 2
```

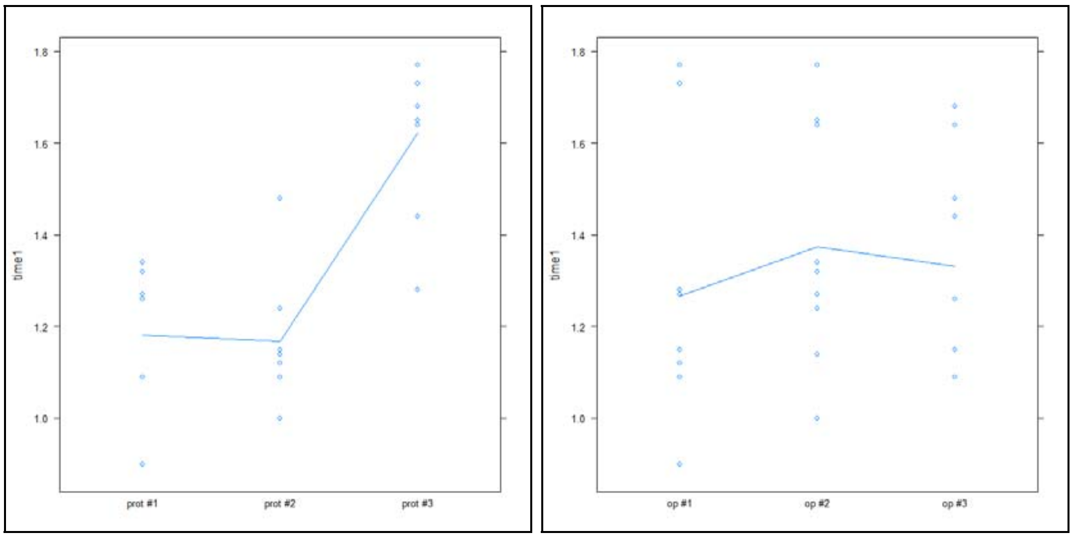
Now we can draw a bar chart:

```
databar<-cbind(varComp[c(1,2,3,6),2],varComp[c(1,2,3,6),5])
barchart(databar, stack=FALSE, horizontal=FALSE, freq=FALSE)
```



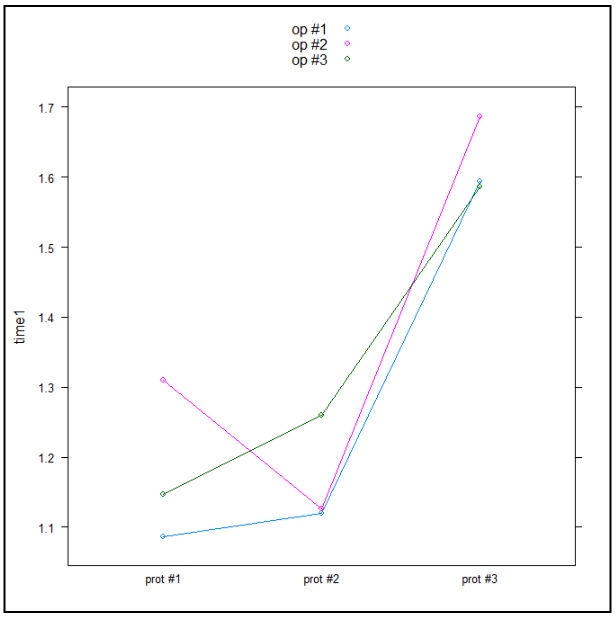
And the plots by part and appraisal:

```
stripplot(time1~prototype,data=ss.data.rr, type=c("p","a"))
stripplot(time1~operator,data=ss.data.rr, type=c("p","a"))
```



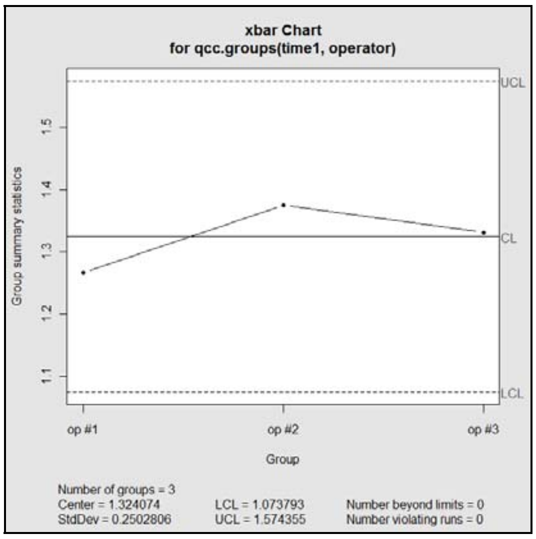
We need the means to plot the interaction plot (and then the control charts):

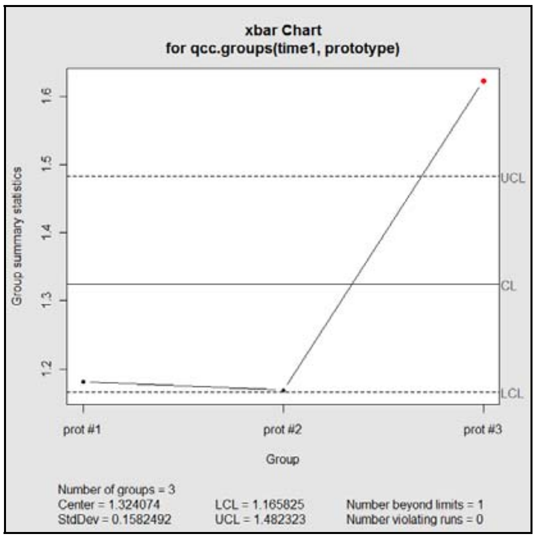
```
data.xbar<-aggregate(data=ss.data.rr,time1~operator+prototype,mean)
stripplot(time1~prototype,groups=operator, data=data.xbar,
   type=c("p","a"),auto.key=TRUE)
```



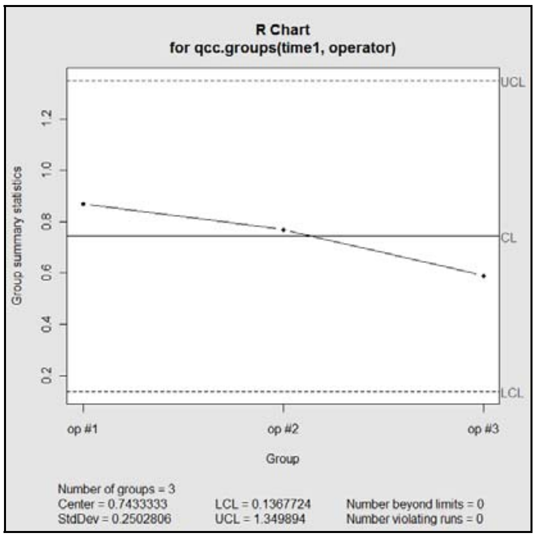
There are no functions in base package for control charts. If we call for a control chart with **qcc** package, we get generic control charts, which are not suitable for R&R Studies, for example:

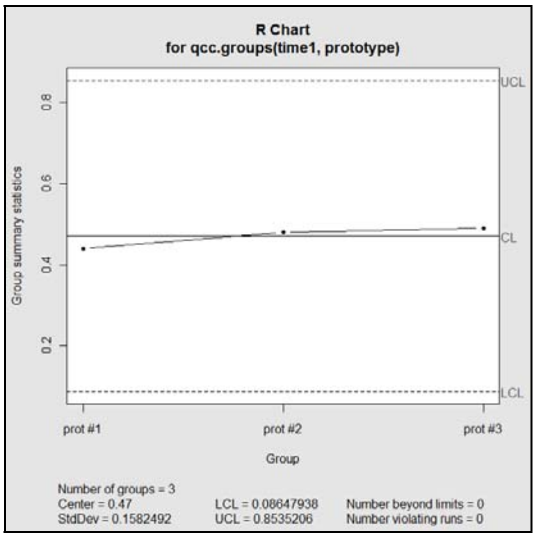
```
qcc(qcc.groups(time1,operator))
qcc(qcc.groups(time1,prototype))
```





```
qcc(qcc.groups(time1,operator),type=”R”)
qcc(qcc.groups(time1,prototype),type=”R”)
```





So let's make the control charts by our own. We already have the means in the variable **`data.xbar`**. We need also the ranges, and the average range:

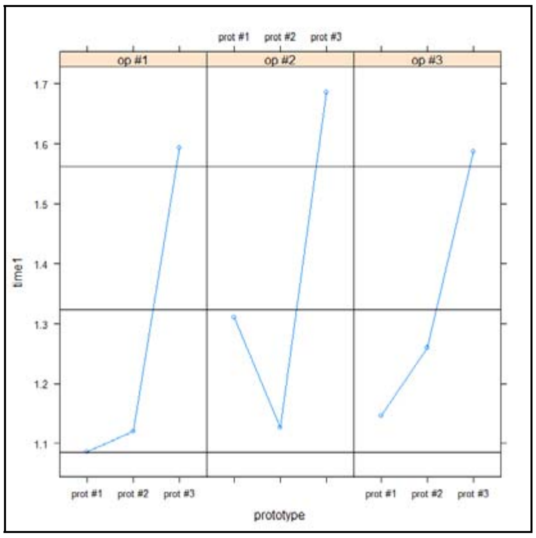
```
data.xrange<-aggregate(data=ss.data.rr,time1~operator+prototype,
   function(x)max(x)-min(x))
ar<-mean(data.xrange$time1)
```

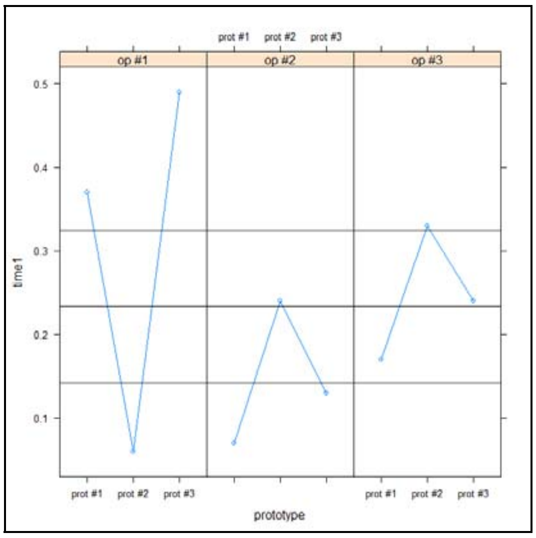
Finally, for the upper and lower limits, we need the constants d_2 (equals to 2.276 for 3 sized samples) and d_3 (equals to 0.888 for 3 sized samples) from the famous constant tables for constructing control charts (see (Montgomery, 2005))

We use the package **`lattice`**[15] so as to represent all the groups in the same plot:

```
xyplot(data=data.xbar, time1~prototype | operator, layout=c(3,1), type="b",
   panel=function(...){
   panel.xyplot(...)
   panel.abline(h=mean(time1,na.rm=TRUE))
   panel.abline(h=mean(time1,na.rm=TRUE)+(3/(1.693*sqrt(3)))*ar)
   panel.abline(h=mean(time1,na.rm=TRUE)(3/(1.693*sqrt(3)))*ar)})
xyplot(data=data.xrange, time1~prototype | operator, layout=c(3,1), type="b",
   panel=function(...){
   panel.xyplot(...)
   panel.abline(h=mean(time1,na.rm=TRUE))
   panel.abline(h=ar*(1+(0.888)/(2.276)))
   panel.abline(h=ar*(1-(0.888)/(2.276)))})
```

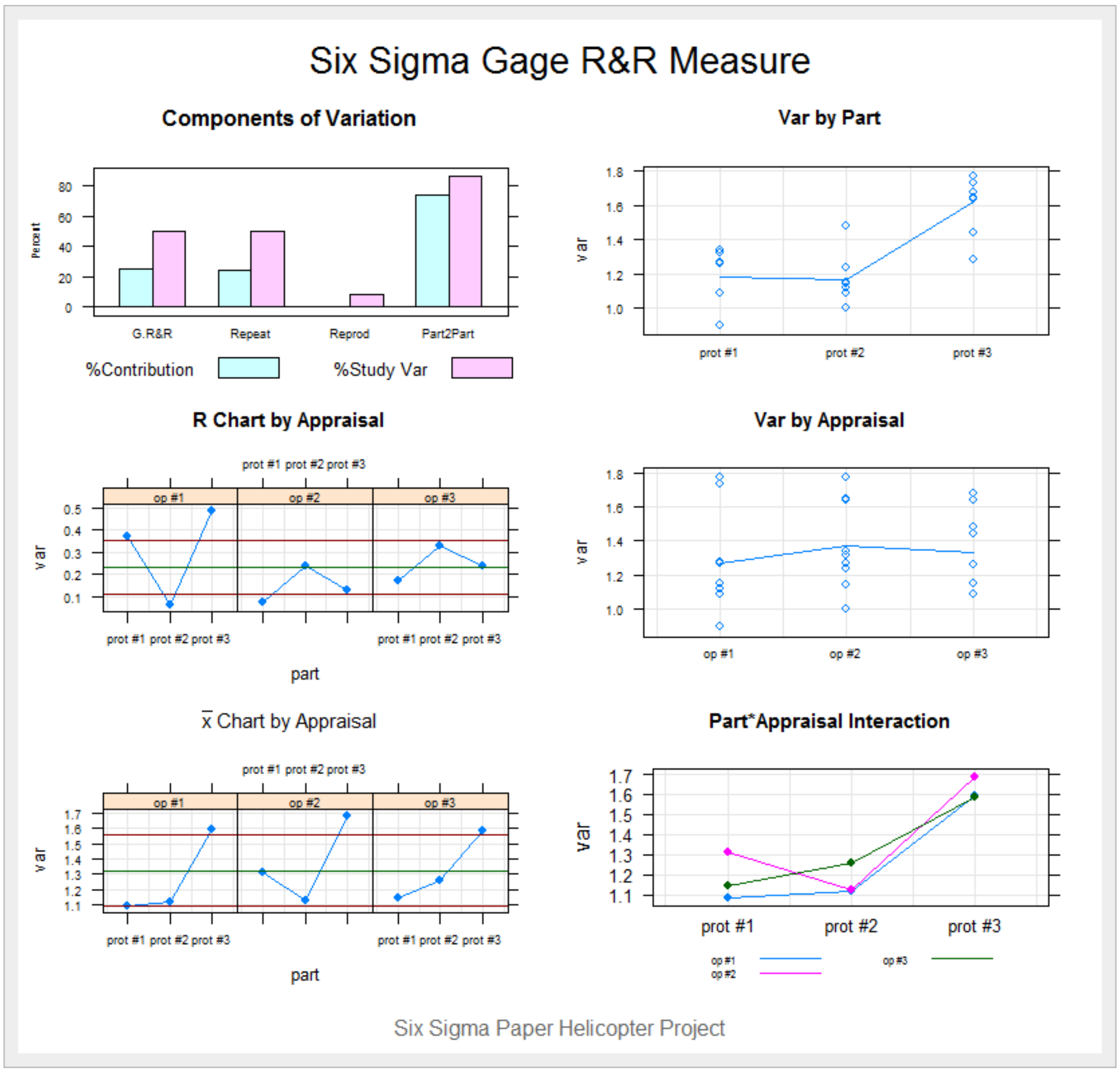
[15] (Sarkar, 2008)





There is included a function in package **SixSigma** that automatically make computations and arrange all plots. That new function is **ss.rr**:

```
> ss.rr(time1, prototype, operator, data=ss.data.rr, sub="Six Sigma Paper
     Helicopter Project")
Analysis of Variance Table

Response: var
              Df  Sum Sq Mean Sq F value    Pr(>F)
part           2 1.20072 0.60036 28.0396 2.952e-06
appr           2 0.05294 0.02647  1.2363    0.3140
part:appr      4 0.08339 0.02085  0.9737    0.4462
Repeatability 18 0.38540 0.02141

Gage R&R
                    VarComp %Contrib
Total Gage R&R    0.0220358    25.50
  Repeatability   0.0214111    24.77
  Reproducibility 0.0006247     0.72
    appr          0.0006247     0.72
    part:appr     0.0000000     0.00
Part-To-Part      0.0643901    74.50
Total Variation   0.0864259   100.00


                     StdDev   5,15*SD %StudyVar
Total Gage R&R    0.1484446 0.7644897     50.49
  Repeatability   0.1463253 0.7535754     49.77
  Reproducibility 0.0249940 0.1287191      8.50
    appr          0.0249940 0.1287191      8.50
    part:appr     0.0000000 0.0000000      0.00
Part-To-Part      0.2537520 1.3068230     86.32
Total Variation   0.2939828 1.5140115    100.00

Number of Distinct Categories=2
```



Graph 6: SixSigma R&R graphics

3.2. Bayesian thoughts

Though there is not much inference in that Sig Sigma methodology step, we may ask some questions:

1. Should we find out about Bayesian ANOVA? We usually have prior information about our processes, and we could use it.
2. Can we improve control charts from the Bayesian perspective?

Appendix 3A: Methodological notes

For the ANOVA table, it is performed a classical analysis. Residuals Sum of Squares, are renamed as Repeatability, because of the Variance Decomposition we need to study R&R:

A = Parts (a items)

B = Appraisals (b items)

n = runs

Total measurements: a*b*n

$SST = SSA + SSB + SSAB + SSE$

$\sigma^2_{Total} = \sigma^2_{Part} + \sigma^2_{Repeatability} + \sigma^2_{Reproducibility}$

$\sigma^2_{Repeatability} = SSE$

$$\sigma^2_{Appraisal} = \frac{SSB - SSAB}{an}$$

$$\sigma^2_{Appr*Part} = \frac{SSAB - SSE}{n}; 0\ if\ negative$$

$$\sigma^2_{Reproducibility} = \sigma^2_{Appraisal} + \sigma^2_{Appr*Part}$$

$$\sigma^2_{Gage\ R\&R} = \sigma^2_{Repeatability} + \sigma^2_{Reproductibility}$$

$$\sigma^2_{Part-to-Part} = \frac{SSA - SSAB}{bn}$$

$$\sigma^2_{Total} = \sigma^2_{Gage\ R\&R} + \sigma^2_{Part-to-Part}$$

Once we have made those computations, we have the matrix to print the figures and plot the bar chart.

The number of distinct categories is taken from:

$$ncat = \frac{\sigma^2_{Part-to-Part}}{\sigma^2_{Reproducibility}}$$

The graphics are simple representation of the data. Only control chart limits need some explanation:

$\bar{x}$ chart

Center line: $\bar{\bar{x}}$

Upper limit: $\bar{\bar{x}} + \frac{3}{d_2\sqrt{n}}\bar{R}$

Lower limit: $\bar{\bar{x}} - \frac{3}{d_2\sqrt{n}}\bar{R}$

$\bar{R}$ chart

Center line: $\bar{R}$

Upper limit: $\bar{R}\left(1 + \frac{d_3}{d_2}\right)$

Lower limit: $\bar{R}\left(1 \quad \frac{d_3}{d_2}\right)$

Where d_3 and d_2 are constants that depends only on sample size (n). See (Montgomery, 2005)

Appendix 3B: R functions Code

```
ss.rr<-function(var, part, appr, data="stop('Data' is requiered for lattice
     graphics)",
                main="Six Sigma Gage R&R Measure", sub=""){

##Figures and facts
    if (is.data.frame(data)){attach(data,warn.conflicts=FALSE)}
    a<-length(levels(part))
    b<-length(levels(appr))
    n<-length(var)/(a*b)
    options(show.signif.stars=FALSE)
    model<-anova(lm(var~part+appr+part*appr))
    rownames(model)[length(rownames(model))]<-"Repeatability"
    print(model)
    varComp<-matrix(ncol=5,nrow=7)
    rownames(varComp)<-c("Total Gage R&R", "  Repeatability", "
     Reproducibility",
                       paste("   ", rownames(model)[2]),
                       paste("   ",rownames(model)[3]), "Part-To-Part",
                        "Total Variation")
    colnames(varComp)<-c("VarComp","%Contrib","StdDev",
     "5,15*SD","%StudyVar")
    varComp[2,1]<-model[4,3]
    varComp[4,1]<-(model[2,3]-model[3,3])/(a*n)
    varComp[5,1]<-max(c((model[3,3]-model[4,3])/n,0))
    varComp[3,1]<-varComp[4,1]+varComp[5,1]
    varComp[6,1]<-(model[1,3]-model[3,3])/(b*n)
    varComp[1,1]<-varComp[2,1]+varComp[3,1]
    varComp[7,1]<-varComp[1,1]+varComp[6,1]
    varComp[,1]<-round(varComp[,1],7)
    varComp[,2]<-round(100*(varComp[,1]/varComp[7,1]),2)
    varComp[,3]<-sqrt(varComp[,1])
    varComp[,4]<-varComp[,3]*5.15
    varComp[,5]<-round(100*(varComp[,4]/varComp[7,4]),2)

    message("\nGage R&R")
    print(varComp[,1:2])
    message("\n")
    print(varComp[,3:5])
    ncat<-max(c(1,floor((varComp[6,3]/varComp[1,3])*1.41)))
    message(paste("\nNumber of Distinct Categories="),ncat,"\n")

##graph
    .ss.prepCanvas(main, sub)
    vp.plots<-viewport(name="plots",
                      layout=grid.layout(3,2))
    pushViewport(vp.plots)
    vp.bar<-viewport(name="barplot", layout.pos.row=1, layout.pos.col=1)
    pushViewport(vp.bar)
    databar<-cbind(varComp[c(1,2,3,6),2],varComp[c(1,2,3,6),5])
    rownames(databar)<-c("G.R&R","Repeat","Reprod","Part2Part")
    #Barchart
    plot<- barchart(databar, freq=FALSE, grid=TRUE,
                    par.settings=list(axis.text=list(cex=0.6),
                                      par.ylab.text=list(cex=0.8),
                                      par.main.text=list(cex=0.85)),
                    ylab=list("Percent",fontsize=8),
                    panel=function(...){
                        panel.barchart(...)
                        panel.abline(h=0)
                    },
                    auto.key=list(text=c("%Contribution","%Study Var"),
                                 cex=0.8,
                                 columns=2, space="bottom", cex=0.8,
                                 rectangles=TRUE, points=FALSE,adj=1,
                                 rep=FALSE
                                 ),
                    stack=FALSE,horizontal=FALSE,
                    main=list("Components of Variation",fontsize=14))
    print(plot, newpage=FALSE)
    popViewport()
    vp.varByPart<-viewport(name="varByPart",layout.pos.row=1,
     layout.pos.col=2)
    pushViewport(vp.varByPart)
    #Var by part
    plot<-stripplot(var~part,data=data, grid=TRUE,
                    par.settings=list(axis.text=list(cex=0.6),
                                   par.xlab.text=list(cex=0.8),
                                   par.ylab.text=list(cex=0.8),
                                   par.main.text=list(cex=0.9)),
                    main="Var by Part",
                    type=c("p","a"))
    print(plot,newpage=FALSE)
    popViewport()
    vp.varByAppr<-viewport(name="varByAppr",layout.pos.row=2,
     layout.pos.col=2)
    pushViewport(vp.varByAppr)
    #var by appraisal
    plot<-stripplot(var~appr,data=data, grid=TRUE,
                    par.settings=list(axis.text=list(cex=0.6),
                                   par.xlab.text=list(cex=0.8),
                                   par.ylab.text=list(cex=0.8),
                                   par.main.text=list(cex=0.9)),
                    main="Var by Appraisal",
                    type=c("p","a"))
    print(plot,newpage=FALSE)
    popViewport()
    vp.Interact<-viewport(name="Interact",layout.pos.row=3, layout.pos.col=2)
    pushViewport(vp.Interact)

    #Interaction
    data.xbar<-aggregate(data=ss.data.rr,var~appr+part,mean)
    plot<-stripplot(var~part,groups=appr, data=data.xbar, pch=16,grid=TRUE,
                    par.settings=list(par.main.text=list(cex=0.9)),
                    main="Part*Appraisal Interaction",
                    type=c("p","a"),
                    auto.key=list(text=levels(appr),
                                  columns=2, space="bottom", cex=0.5,
                                  lines=TRUE, points=FALSE,adj=1))
    print(plot,newpage=FALSE)
    popViewport()
```

```
#Control Charts

data.xrange<-aggregate(data=ss.data.rr,var~appr+part,function(x)max(x)-
 min(x))
ar<-mean(data.xrange$var)
#Mean
vp.ccMean<-viewport(name="ccMean",layout.pos.row=3, layout.pos.col=1)
pushViewport(vp.ccMean)
plot<-xyplot(data=data.xbar, var~part|appr, pch=16,
             par.settings=list( axis.text=list(cex=0.6),
                                par.xlab.text=list(cex=0.8),
                                par.ylab.text=list(cex=0.8),
                                par.main.text=list(cex=0.9)),
             par.strip.text=list(cex=0.6),
             main=expression(bar(x)*" Chart by Appraisal"),grid=TRUE,
             layout=c(n,1),
             type="b",
             panel=function(...){
                 panel.xyplot(...)
                 panel.abline(h=mean(var,na.rm=TRUE), col="darkgreen")
                 panel.abline(h=mean(var,na.rm=TRUE)+
                              (3/(.ss.cc.getConst(n,"d2")*sqrt(n)))*ar,
                              col="darkred")
                 panel.abline(h=mean(var,na.rm=TRUE)-
                              (3/(.ss.cc.getConst(n,"d2")*sqrt(n)))*ar,
                              col="darkred")

             }
             )
print(plot,newpage=FALSE)
popViewport()
##Range
vp.ccRange<-viewport(name="ccRange", layout.pos.row=2, layout.pos.col=1)
pushViewport(vp.ccRange)


plot<-xyplot(data=data.xrange, var~part|appr, pch=16,
             par.settings=list(axis.text=list(cex=0.6),
                                par.xlab.text=list(cex=0.8),
                                par.ylab.text=list(cex=0.8),
                                par.main.text=list(cex=0.9)),
             par.strip.text=list(cex=0.6),
             main="R Chart by Appraisal",grid=TRUE,
             layout=c(n,1),
             type="b",
             panel=function(...){
                 panel.xyplot(...)
                 panel.abline(h=ar, col="darkgreen")
                 panel.abline(h=ar*(1+

 (.ss.cc.getConst(n,"d3")/(.ss.cc.getConst(n,"d2")))),
                              col="darkred")
                 panel.abline(h=ar*(1-

 (.ss.cc.getConst(n,"d3")/(.ss.cc.getConst(n,"d2")))),
                              col="darkred")

             }
             )
```

```
    print(plot,newpage=FALSE)
    popViewport()

}
```

Aux Functions

```
.ss.prepCanvas<-function(main="Six Sigma graph", sub="My Six Sigma Project",
                        palette=c("#666666", "#BBBBBB", "#CCCCCC", "#DDDDDD",
     "#EEEEEE")){
#Plot
    grid.newpage()
    grid.rect(gp=gpar(col=palette[2], lwd=2, fill=palette[5]))
    vp.canvas<-viewport(name="canvas",
                        width=unit(1,"npc")-unit(6,"mm"),
                        height=unit(1,"npc")-unit(6,"mm"),
                        layout=grid.layout(3,1,
             heights=unit(c(3,1,2), c("lines", "null", "lines"))
      ))
    pushViewport(vp.canvas)
    grid.rect(gp=gpar(col="#FFFFFF", lwd=0, fill="#FFFFFF"))

#Title
    vp.title<-viewport(layout.pos.col=1, layout.pos.row=1, name="title")
    pushViewport(vp.title)
    grid.text (main, gp=gpar(fontsize=20))
    popViewport()

#Subtitle
    vp.subtitle<-viewport(layout.pos.col=1, layout.pos.row=3,
     name="subtitle")
    pushViewport(vp.subtitle)
    grid.text ( sub, gp=gpar(col=palette[1]))
    popViewport()

#Container
    vp.container<-viewport(layout.pos.col=1, layout.pos.row=2,
     name="container")
    pushViewport(vp.container)
}

#Getting constans for Control Charts
.ss.cc.getConst<-function(sample.size,const="stop('Constant' requiered"){
    if (sample.size>25){
        stop("Not developed yet for sample sizes greater than 25")
    }
    data(ss.data.ccConstants)
    switch (const,
            "d2"=subset(ss.data.ccConstants,rr.n==sample.size)$d2,
            "d3"=subset(ss.data.ccConstants,rr.n==sample.size)$d3,
            "c4"=subset(ss.data.ccConstants,rr.n==sample.size)$c4,
            stop("Incorrect constant")
            )
}
```

Appendix 3C: Data tables

```
> ss.data.rr
   prototype operator    run time1 time2
1    prot #1    op #1 run #1  1.27  1.15
2    prot #1    op #1 run #2  0.90  1.31
3    prot #1    op #1 run #3  1.09  1.25
4    prot #2    op #1 run #1  1.12  1.36
5    prot #2    op #1 run #2  1.09  1.41
6    prot #2    op #1 run #3  1.15  1.37
7    prot #3    op #1 run #1  1.73  1.95
8    prot #3    op #1 run #2  1.28  1.86
9    prot #3    op #1 run #3  1.77  1.86
10   prot #1    op #2 run #1  1.34  1.38
11   prot #1    op #2 run #2  1.32  1.27
12   prot #1    op #2 run #3  1.27  1.33
13   prot #2    op #2 run #1  1.24  1.26
14   prot #2    op #2 run #2  1.00  1.31
15   prot #2    op #2 run #3  1.14  1.24
16   prot #3    op #2 run #1  1.64  1.86
17   prot #3    op #2 run #2  1.65  2.07
18   prot #3    op #2 run #3  1.77  2.10
19   prot #1    op #3 run #1  1.09  1.28
20   prot #1    op #3 run #2  1.26  1.16
21   prot #1    op #3 run #3  1.09  1.37
22   prot #2    op #3 run #1  1.48  1.24
23   prot #2    op #3 run #2  1.15  1.28
24   prot #2    op #3 run #3  1.15  1.39
25   prot #3    op #3 run #1  1.64  1.75
26   prot #3    op #3 run #2  1.44  2.05
27   prot #3    op #3 run #3  1.68  1.81
```

```
ss.data.rr              package:SixSigma              R Documentation

Gage R&R data

Description:

     Data for Measure phase of the Six Sigma breakthrough strategy.

Usage:

     data(ss.data.rr)

Format:

     A data frame with 27 observations on the following 5 variables.

     'prototype' a factor with levels 'prot #1' 'prot #2' 'prot #3'

     'operator' a factor with levels 'op #1' 'op #2' 'op #3'
     'run' a factor with levels 'run #1' 'run #2' 'run #3'

     'time1' a numeric vector

     'time2' a numeric vector

Details:

     The flight time has been measured 54 times: time1 and time2 are
     the measure before and after the operators training. There are
     trhee runs for each of the trhee operators and three prototypes.

Source:

     Our own data experience

Examples:

     data(ss.data.rr)
     str(ss.data.rr)
```

Appendix 3D: Functions help

```
ss.rr                 package:SixSigma                 R Documentation

Gage R & R

Description:

     Performs Gage R&R analysis. Related to the Measure phase of DMAIC
     strategy of Six Sigma.

Usage:

     ss.rr(var, part, appr, data = "stop('Data' is requiered for lattice
     graphics)",
     main = "Six Sigma Gage R&R Measure", sub = "")

Arguments:

     var: Measured variate

    part: Factor for parts

    appr: Factor for appraisals (operators, machines, ...)

    data: Data frame containing the variates

    main: Main title for the graphic output

     sub: Subtitle for the graphic output (recommended the name of the
          project)

Details:

     Performs an R&R study for the measured variable R, taking in
     account part and appr factors. It outputs the sources of
     Variability, and six graphs: bar chart with the sources of
     Variablity, plots by appraisal, part and interaction and x-mean
     and R control charts.

Value:

     Analysis of Variance Table. Variance composition and %Study Var.
     Graphics.

Author(s):

     Emilio Lopez Cano

References:

     Montgomery, D. C. (2005). Introduction to Statistical Quality
     Control (Fifth Edition ed.). New York: Wiley & Sons, Inc.
     Allen, T. T. (2010). Introduction to Engineering Statistics and
     Lean Six Sigma - Statistical Quality Control and Design of
     Experiments and Systems (Second Edition ed.). London: Springer.

See Also:

     'ss.data.rr'

Examples:

     data(ss.data.rr)
     ss.rr(time1, prototype, operator, data=ss.data.rr, sub="Six Sigma Paper
     Helicopter Project")
```

4. Conclusions

The use of R environment and programming language is increasing exponentially among every uses of Statistics. It is quite feasible deal with any statistical task with R, and even improving or deploying one's specific function for individual needs with some training.

Six Sigma and other enterprise-oriented methodologies, traditionally use commercial software rather than free software, due to some prejudice. Fortunately this is changing, and Companies are more and more changing their minds, and realizing the advantages of shared knowledge.

This essay and the related package developed, have explained how to perform some of the tasks regarding Six Sigma. But there is not much information available out there about how to perform Six Sigma with R.

The further research of this essay aims at covering this room and helping those who want to exploit the advantages of using R in their everyday Six Sigma Management, especially Six Sigma Black-Belts.

Further Research

This Document is the beginning of a more ambitious work. There are planned several work lines, taking this Master Thesis as a starting point.

`SixSigma` R package is already available at CRAN, and everybody can install and use it. But there are lots of Six Sigma tasks to be described in R. New functions can be developed, in order to make possible a complete Six Sigma project deployment in R, as it was said in the introduction.

While developing the package, too much information will be taking in account, and it will be written new documentation about foundations of every Six Sigma stage (DMAIC), Statistical Techniques to be used, and how to perform them with R.

Maybe some of the "Bayesian Thoughts" will be gathered in new functions or documentation. Other advanced statistical techniques can appear by the way: Optimal Experimental Design, Non-linear models, Data Mining, Optimization, Simulation …

There could be also a Graphic Interface that makes the use of the package friendly for the users, via **`Rcmdr`** plug-in, or even a more sophisticated system.

All that research and development eventually will end up in some publications, surely very useful for Six Sigma practitioners.

Madrid, 7th june 2011

References

Allen, T. T. (2010). *Introduction to Engineering Statistics and Lean Six Sigma - Statistical Quality Control and Design of Experiments and Systems* (Second Edition ed.). London: Springer.

Bolstad, W. M. (2007). *Introduction to Bayesian Statistics.* New Jersey: Wiley.

Chambers, J. M. (2008). *Software for data analysis. Programming with R.* New York: Springer.

Crawley, M. J. (2007). *The R Book.* Chichester (England): Wiley.

Daniela R. Recchia <daniela_recchia@yahoo.com.br>, Emanuel P. Barbosa and Elias de Jesus Goncalves <eliasjg@gmail.com>. (2010). IQCC: Improved Quality Control Charts. R package version 0.5. *http://CRAN.R-project.org/package=IQCC* .

Donald W. Benbow, T. K. (2005). *The Certified Six Sigma Black Belt Handbook.* ASQ Quality Press.

fox, J. (2011). Rcmdr: R Commander. R package version 1.6-3. http://CRAN.R-project.org/package=Rcmdr.

Hornik, K. (2010). *The R FAQ.* Retrieved 02 23, 2010, from cran.r-project.org: http://CRAN.R-project.org/doc/FAQ/R-FAQ.html

http://rwiki.sciviews.org/. (n.d.).

Kiermeier, A. (2008). Visualising and Assessing Acceptance Sampling Plans: The R Package AcceptanceSampling. *Journal of Statistical Software* , 26(6).

Minitab Inc. (2010) website: http://www.mititab.com. (n.d.).

Montgomery, D. C. (2005). *Introduction to Statistical Quality Control* (Fifth Edition ed.). New York: Wiley & Sons, Inc.

Murrell, P. (2005). *R Graphics.* London: Chapman&Hall/CRC.

R Development Core Team (2010). R: A language and environment for statistical computing. R Foundation for Statistical Computing, V. A.-9.-0.-0.-p. (n.d.).

Roth, T. (2010). qualityTools: A Package for Teaching Statistics in Quality Science. *R package version 1.3* .

Sarkar, D. (2008). *Multivariate Data Visualization with R.* New York: Springer.

Scrucca, L. (2004). qcc: an R package for quality control charting and statistical process control. *R News* (4/1), 11-17.

Soetaert, K. (2009). diagram: Functions for visualising simple graphs (networks), plotting flow diagrams. R package version 1.5. *http://CRAN.R-project.org/package=diagram* .