Quality Assurance for R Graphics

Paul Murrell & Kurt Hornik

The University of Auckland
New Zealand

Technische Universität Wien
Austria

Overview

- An Anonymous Example
- Quality Assurance (in R)
- Quality Assurance for Graphics (in R)

Overview

- An Anonymous Example
- Quality Assurance (in R)
- Quality Assurance for Graphics (in R)

Overview

- An Anonymous Example
- Quality Assurance (in R)
- Quality Assurance for Graphics (in R)

Overview

- An Anonymous Example
- Quality Assurance (in R)
- Quality Assurance for Graphics (in R)

An Anonymous Example

```
> grid.rect(w=unit(1, "grobwidth",
                 data=grid.text(5000)))
```

- grid-0.7-2 (December 2002)

  ```
  R: line 1: 26386 Segmentation fault
  ```

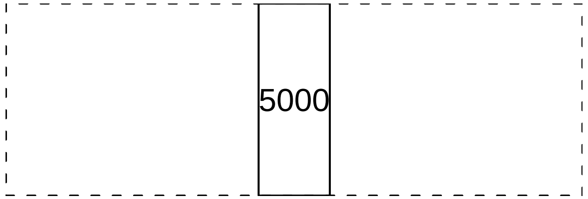
An Anonymous Example

```
> grid.rect(w=unit(1, "grobwidth",
                   data=grid.text(5000)))
```

- grid-0.7-2 (December 2002)

  `R: line 1: 26386 Segmentation fault`

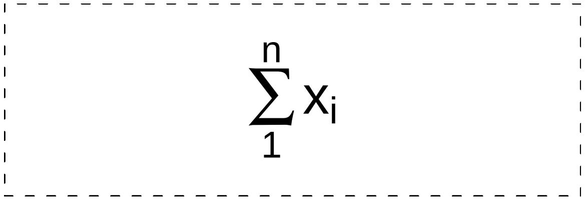
- grid-0.7-3 (December 2002)



An Anonymous Example

```
> grid.text(expression(sum(x[i],1,n)))
```

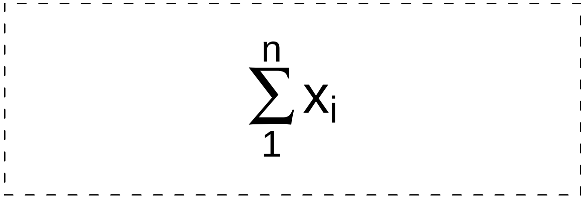
- grid-0.7-2 (December 2002)



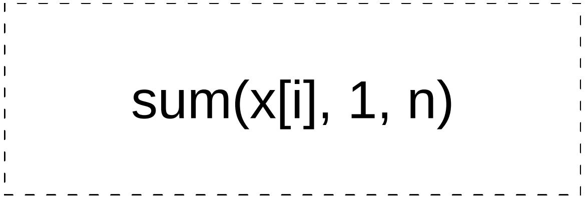
An Anonymous Example

```
> grid.text(expression(sum(x[i],1,n)))
```

- grid-0.7-2 (December 2002)



- grid-0.7-3 (December 2002)



Quality Assurance

- Everyone writes buggy code
- Not all bugs are so obvious
- There are a lot of cases to check

- **Quality Control:** need (automated) **tools** for **detecting** bugs.
- **Quality Assurance:** need **processes** for **preventing** the release of bugs

Quality Assurance

- Everyone writes buggy code
- Not all bugs are so obvious
- There are a lot of cases to check

- **Quality Control:** need (automated) **tools** for **detecting** bugs.
- **Quality Assurance:** need **processes** for **preventing** the release of bugs

Quality Assurance

- Everyone writes buggy code
- Not all bugs are so obvious
- There are a lot of cases to check

- **Quality Control:** need (automated) **tools** for **detecting** bugs.
- **Quality Assurance:** need **processes** for **preventing** the release of bugs

Quality Assurance

- Everyone writes buggy code
- Not all bugs are so obvious
- There are a lot of cases to check

- **Quality Control:** need (automated) **tools** for **detecting** bugs.
- **Quality Assurance:** need **processes** for **preventing** the release of bugs

Quality Assurance

- Everyone writes buggy code
- Not all bugs are so obvious
- There are a lot of cases to check

- **Quality Control:** need (automated) **tools** for **detecting** bugs.
- **Quality Assurance:** need **processes** for **preventing** the release of bugs

Quality Assurance

- Everyone writes buggy code
- Not all bugs are so obvious
- There are a lot of cases to check

- **Quality Control:** need (automated) **tools** for **detecting** bugs.
- **Quality Assurance:** need **processes** for **preventing** the release of bugs

Quality Assurance in R

- R has `make check` (and more)
- R has `R CMD check`
- Code must pass `make check` before it is committed to the repository
- A package must pass `R CMD check` before it is allowed on CRAN
- There are daily runs of `make check` and `R CMD check` on all of CRAN and on several different platforms.

- R's QC for **graphics** is weak
- No automated check that the **output** is correct

Quality Assurance in R

- R has `make check` (and more)
- R has `R CMD check`
- Code must pass `make check` before it is committed to the repository
- A package must pass `R CMD check` before it is allowed on CRAN
- There are daily runs of `make check` and `R CMD check` on all of CRAN and on several different platforms.

- R's QC for **graphics** is weak
- No automated check that the **output** is correct

Quality Assurance in R

- R has `make check` (and more)
- R has `R CMD check`
- Code must pass `make check` before it is committed to the repository
- A package must pass `R CMD check` before it is allowed on CRAN
- There are daily runs of `make check` and `R CMD check` on all of CRAN and on several different platforms.

- R's QC for **graphics** is weak
- No automated check that the **output** is correct

Quality Assurance in R

- R has `make check` (and more)
- R has `R CMD check`
- Code must pass `make check` before it is committed to the repository
- A package must pass `R CMD check` before it is allowed on CRAN
- There are daily runs of `make check` and `R CMD check` on all of CRAN and on several different platforms.

- R's QC for **graphics** is weak
- No automated check that the **output** is correct

Quality Assurance in R

- R has `make check` (and more)
- R has `R CMD check`
- Code must pass `make check` before it is committed to the repository
- A package must pass `R CMD check` before it is allowed on CRAN
- There are daily runs of `make check` and `R CMD check` on all of CRAN and on several different platforms.

- R's QC for **graphics** is weak
- No automated check that the **output** is correct

Quality Assurance in R

- R has `make check` (and more)
- R has `R CMD check`
- Code must pass `make check` before it is committed to the repository
- A package must pass `R CMD check` before it is allowed on CRAN
- There are daily runs of `make check` and `R CMD check` on all of CRAN and on several different platforms.

- R's QC for **graphics** is weak
- No automated check that the **output** is correct

Quality Assurance in R

- R has `make check` (and more)
- R has `R CMD check`
- Code must pass `make check` before it is committed to the repository
- A package must pass `R CMD check` before it is allowed on CRAN
- There are daily runs of `make check` and `R CMD check` on all of CRAN and on several different platforms.

- R's QC for **graphics** is weak
- No automated check that the **output** is correct

Quality Assurance in R

- R has `make check` (and more)
- R has `R CMD check`
- Code must pass `make check` before it is committed to the repository
- A package must pass `R CMD check` before it is allowed on CRAN
- There are daily runs of `make check` and `R CMD check` on all of CRAN and on several different platforms.

- R's QC for **graphics** is weak
- No automated check that the **output** is correct

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party “viewers”
  - Easier to “see” problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics

- Check that code runs **and** produces the correct output
- Need **model** output to compare against
- There are multiple output formats (devices)
- Bitmaps should be the basis for testing
  - Only some output devices use a text representation
  - A change in text representation may be unimportant
  - Will detect problems in 3rd party "viewers"
  - Easier to "see" problems
  - Close as possible to what the user sees

QC for Graphics in `R`

- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

QC for Graphics in R

- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

QC for Graphics in `R`

- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

QC for Graphics in R

- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

QC for Graphics in R

- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

QC for Graphics in R

- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

QC for Graphics in R

- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

QC for Graphics in `R`

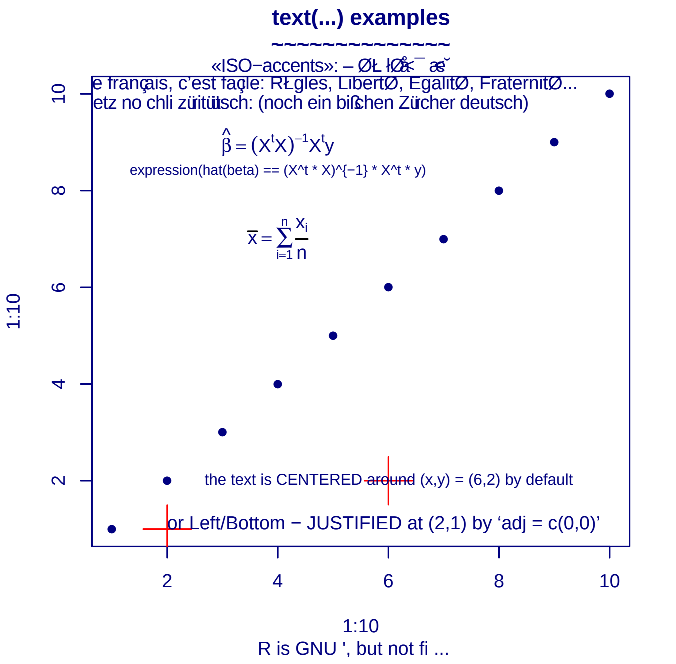
- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

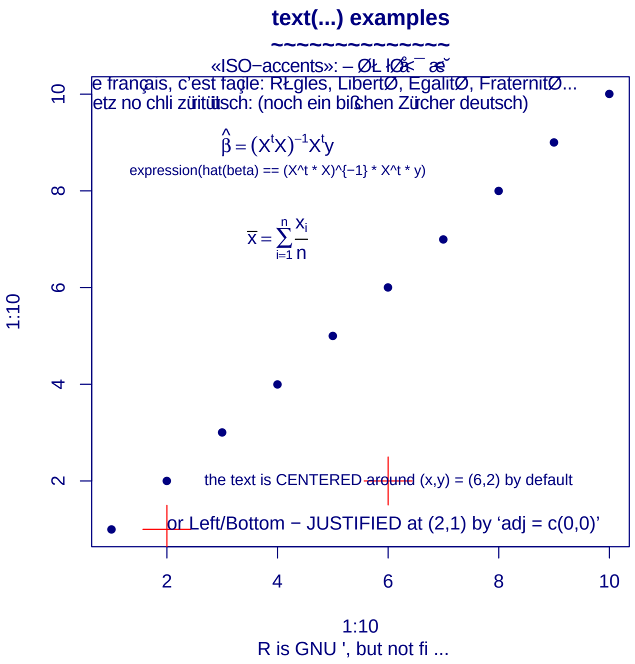
QC for Graphics in R

- Tools for producing model output
- Tools for producing test output, comparing with model, and showing differences
- Linking these tools into `make check` and `R CMD check`
- Package `graphicsQC`
  - model.graphics()
  - test.graphics()
  - clean.graphics()
  - `http://www.stat.auckland.ac.nz/~paul/R/graphicsQC_0.1.tar.gz`

Package graphicsQC

```
test.graphics(funs = NULL,
              package = NULL,
              names = NULL,
              omit = NULL,
              width = 600, height = 600,
              device = postscript,
              format = "pbm",
              model.loc = ".",
              test.loc = model.loc,
              verbose = FALSE, quiet=FALSE,
              reset.rng = TRUE,
              ...)
```


text(...) examples
~~~~~~~~~~~~~~
«ISO−accents»: – ØŁ łØå<¯ æ˘
e français, c'est façile: RŁgles, LibertØ, EgalitØ, FraternitØ...
etz no chli züritütsch: (noch ein bißchen Zürcher deutsch)
$\hat{\beta} = (X^tX)^{-1}X^ty$
expression(hat(beta) == (X^t * X)^{−1} * X^t * y)
$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$
the text is CENTERED around (x,y) = (6,2) by default
or Left/Bottom − JUSTIFIED at (2,1) by 'adj = c(0,0)'
1:10
R is GNU ', but not fi ...


text(...) examples
~~~~~~~~~~~~~~
«ISO−accents»: – ØŁ łØå<¯ æ˘
e français, c'est façile: RŁgles, LibertØ, EgalitØ, FraternitØ...
etz no chli züritütsch: (noch ein bißchen Zürcher deutsch)
$\hat{\beta} = (X^tX)^{-1}X^ty$
expression(hat(beta) == (X^t * X)^{−1} * X^t * y)
$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$
the text is CENTERED around (x,y) = (6,2) by default
or Left/Bottom − JUSTIFIED at (2,1) by 'adj = c(0,0)'
1:10
R is GNU ', but not fi ...

Testing

- `graphicsQC` existed in December 2002
- There was no example `grid` code that tested mathematical annotation
- Not enough just to have testing tools; must use them too!

- Extreme Programming has the notion of **unit tests**

Testing

- `graphicsQC` existed in December 2002
- There was no example `grid` code that tested mathematical annotation
- Not enough just to have testing tools; must use them too!

- Extreme Programming has the notion of **unit tests**

Testing

- `graphicsQC` existed in December 2002
- There was no example `grid` code that tested mathematical annotation
- Not enough just to have testing tools; must use them too!

- Extreme Programming has the notion of **unit tests**

Testing

- `graphicsQC` existed in December 2002
- There was no example `grid` code that tested mathematical annotation
- Not enough just to have testing tools; must use them too!
- Extreme Programming has the notion of **unit tests**

Testing

- `graphicsQC` existed in December 2002
- There was no example `grid` code that tested mathematical annotation
- Not enough just to have testing tools; must use them too!

- Extreme Programming has the notion of **unit tests**

Summary

- There are two important aspects to producing quality software: Quality Assurance and Quality Control (Testing)
- R has reasonably good QA and is getting better QC tools for graphics
- Programmers still need to create appropriate tests for the QC tools and QA process to be of any use

Summary

- There are two important aspects to producing quality software: Quality Assurance and Quality Control (Testing)
- R has reasonably good QA and is getting better QC tools for graphics
- Programmers still need to create appropriate tests for the QC tools and QA process to be of any use

Summary

- There are two important aspects to producing quality software: Quality Assurance and Quality Control (Testing)

- R has reasonably good QA and is getting better QC tools for graphics

- Programmers still need to create appropriate tests for the QC tools and QA process to be of any use

Summary

- There are two important aspects to producing quality software: Quality Assurance and Quality Control (Testing)
- R has reasonably good QA and is getting better QC tools for graphics
- Programmers still need to create appropriate tests for the QC tools and QA process to be of any use