



# RLint: Reformating R Code to Follow the Google Style Guide

Alex Blocker, **Andy Chen** (andych@google.com),  
Andy Chu, Tim Hesterberg, Jeffrey D. Oldham, Caitlin  
Sadowski, Tom Zhang

2014-07-02

## Summary

[RLint](#) checks and reformats R code to follow [R style guide](#).

RLint used within Google.

- Eases checking correctness.
- Improves programmer productivity.

Suggest experiment adopting consistent style guide + RLint.

- Does it improve your team's productivity?

# Style guides improve correctness and productivity

Q: How do we produce correct R code when

- correctness is hard to check,
- R programmer time is expensive?

Q: How do we maintain correct R code when

- modified by different programmers?

## Many R files modified by multiple users

~40% files modified by >1 Googler.

# Googlers modifying R file	% R files
1	60.9%
2-3	33.7%
4-5	3.8%
6+	1.5%

~50% directories contain code written by >1 Googler.

# Googlers modifying R code in directory	% R directories
1	52.0%
2-3	36.6%
4-5	7.0%
6+	4.4%

# Style guides improve correctness and productivity

Q: How do we produce correct R code when

- correctness is hard to check,
- R programmer time is expensive?

Q: How do we maintain correct R code when

- modified by different programmers?

A: [R style guide](#) specifies uniform coding

# Style guides specify program structure

## [Google R style guide](#) specifies

- identifier naming: `variable.name` or `variableName`, `FunctionName`
- layout: indentation, spacing, ...
- comments
- function commenting
- ...

Success criterion: Any programmer should be able to

- **instantly** understand structure of **any** code.

Consistent style more important than "perfect" style.

## RLint: Automate style checking and correction

Goal: Minimize overhead of following style guide.

RLint: Program warning style violations.

- Optionally produce style-conforming code.
- Key idea: Computers are cheap.

Use within Google:

- All code violations flagged by code review tool.
- Reviewer must sign off before code submission.

## Ex: Spacing

Code:

```
foo <-function(x){
  return(list(
    a = sum(x[,1]),
    b = 1/3+1e-7*(x[1,1])) ...
```

### Warnings:

- *Place spaces around all binary operators (=, +, -, <-, etc.).*
- *Place a space before left parenthesis, except in a function call.*

Corrected:

```
foo <- function(x) {
  return(list(
    a = sum(x[,1]),
    b = 1/3 + 1e-7 * (x[1,1]) ...
```



## Ex: Indentation

### Code

```
if (x == 5)
while (x > 1)
    x <- x - 1
    print(x)
```

*Is anything wrong?*

## Ex: Indentation

### Code

```
if (x == 5)
while (x > 1)      # R-bleed bug? ; )
  x <- x - 1
  print(x)
```

### Corrected code

```
if (x == 5)
  while (x > 1)
    x <- x - 1
  print(x)
```

## Ex: Ease checking program correctness

### Code

```
x <- -5:-1  
x[x <-2]
```

*Is anything wrong?*

## Ex: Ease checking program correctness


### Code

```
x <- -5:-1  
x[x <-2]      # Hmm ...
```

### Warning

*Must have whitespace around <-, <<-, etc*

### Corrected code

```
x <- -5:-1  
x[x <-  2]
```

## Ex: Ease checking program correctness

### Code

```
if (format(Sys.time(), "%Y") == "2014") {  
    print(paste("UseR!", "2014"))  
}
```

*Is anything wrong?*

## Ex: Ease checking program correctness

### Code

```
if (format(Sys.time(), "%Y") == "2014") {  
    print(paste("UseR!", "2014"))  
}
```

### Error

```
CRITICAL:root:Unbalanced brackets in  
{  
    print(paste("UseR!", "2014"))  
}
```

## RLint implementation uses Python

Use Python string functions and regular expressions.

Algorithm:

Stub out comments, strings, user-defined operators.

- Ex: Comment may contain code!
- Ex: Multi-line string

Check spacing.

Align & indent lines within {}, () and [].

- Align lines by opening bracket.
- Align lines by '=' if they are in the same bracket.

Align `if/while/for (...)` not followed by {}.

Unstub comments, strings, user-defined operators.

## Application: Improve R community's style consistency

Proposal: Adopt [R style guide](#) + [RLint](#).

- Run experiments to determine net benefit.

Small scale: Individual teams (pkgs) adopt style guide + checker.

- Are these programmers more productive?
- More bug fixes and fewer (un-fixed) bug reports?

Medium scale: CRAN packages opt into style guide + checker.

- Specify style guide + checker program.
- Enforced by [CRAN server farm](#).



## Summary

[RLint](#) checks and reformats R code to follow [R style guide](#).

RLint used within Google

- Eases checking correctness.
- Improves programmer productivity.

Suggest experiment adopting consistent style guide + RLint.

- Does it improve your team's productivity?



# RLint: Reformating R Code to Follow the Google Style Guide

Alex Blocker, **Andy Chen** (andych@google.com),  
Andy Chu, Tim Hesterberg, Jeffrey D. Oldham, Caitlin  
Sadowski, Tom Zhang

2014-07-02

# Coding conventions and checkers

Coding conventions have existed for decades.

- 1918: [\*The Elements of Style\*](#) by Strunk & White (writing English)
- 1974: [\*The Elements of Programming Style\*](#) (writing code)
- 1997: [Java code conventions](#)
- 2001: [Python style guide](#)
- 2014: [Google style guides](#) for 12 languages available

Style checkers have existed for decades.

- 1977: [lint](#) checks C style
- 2002: [PyChecker](#) checks Python style
- 2011: [gofmt](#) reformats Go code ([70% adoption](#) in 2013)