

# **Profvis: Profiling tools for faster R code**

Winston Chang  
RStudio

**How do I make  
my R code faster?**

**Why is my R  
code slow?**

# Optimization: normalizing columns

```
# First generate data with 400000 rows and 150 cols
data <- as.data.frame(
  matrix(rnorm(4e5 * 150, mean = 5), ncol = 150)
)
```

```
normCols <- function(d) {
  # Get vector of column means
  means <- apply(d, 2, mean)

  # Subtract mean from each column
  for (i in seq_along(means)) {
    d[, i] <- d[, i] - means[i]
  }
  d
}
```

```
normCols(data)
```

# Optimization: normalizing columns

```
system.time({  
  normCols <- function(d) {  
    # Get vector of column means  
    means <- apply(d, 2, mean)  
  
    # Subtract mean from each column  
    for (i in seq_along(means)) {  
      d[, i] <- d[, i] - means[i]  
    }  
    d  
  }  
  
  normCols(data)  
})
```

# Profiling

# Sampling profiler

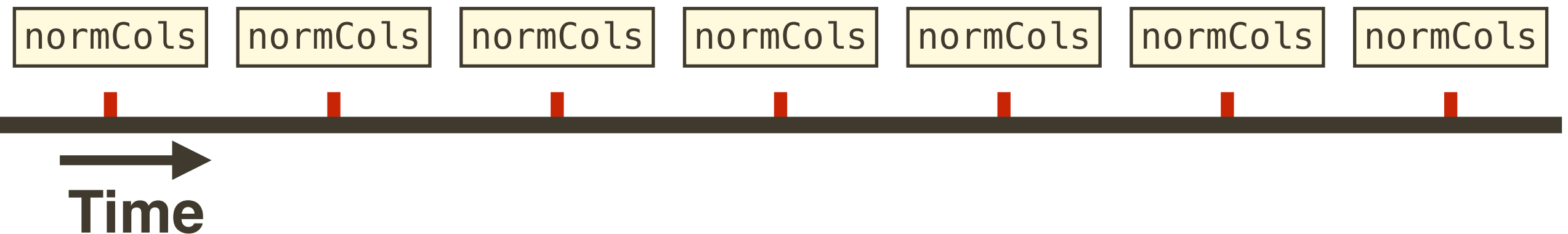
```
Rprof( )          # Start profiling  
normCols(data)  
Rprof(NULL)       # Stop profiling
```

# Sampling profiler

```
Rprof()      # Start profiling
```

```
normCols(data)
```

```
Rprof(NULL)  # Stop profiling
```



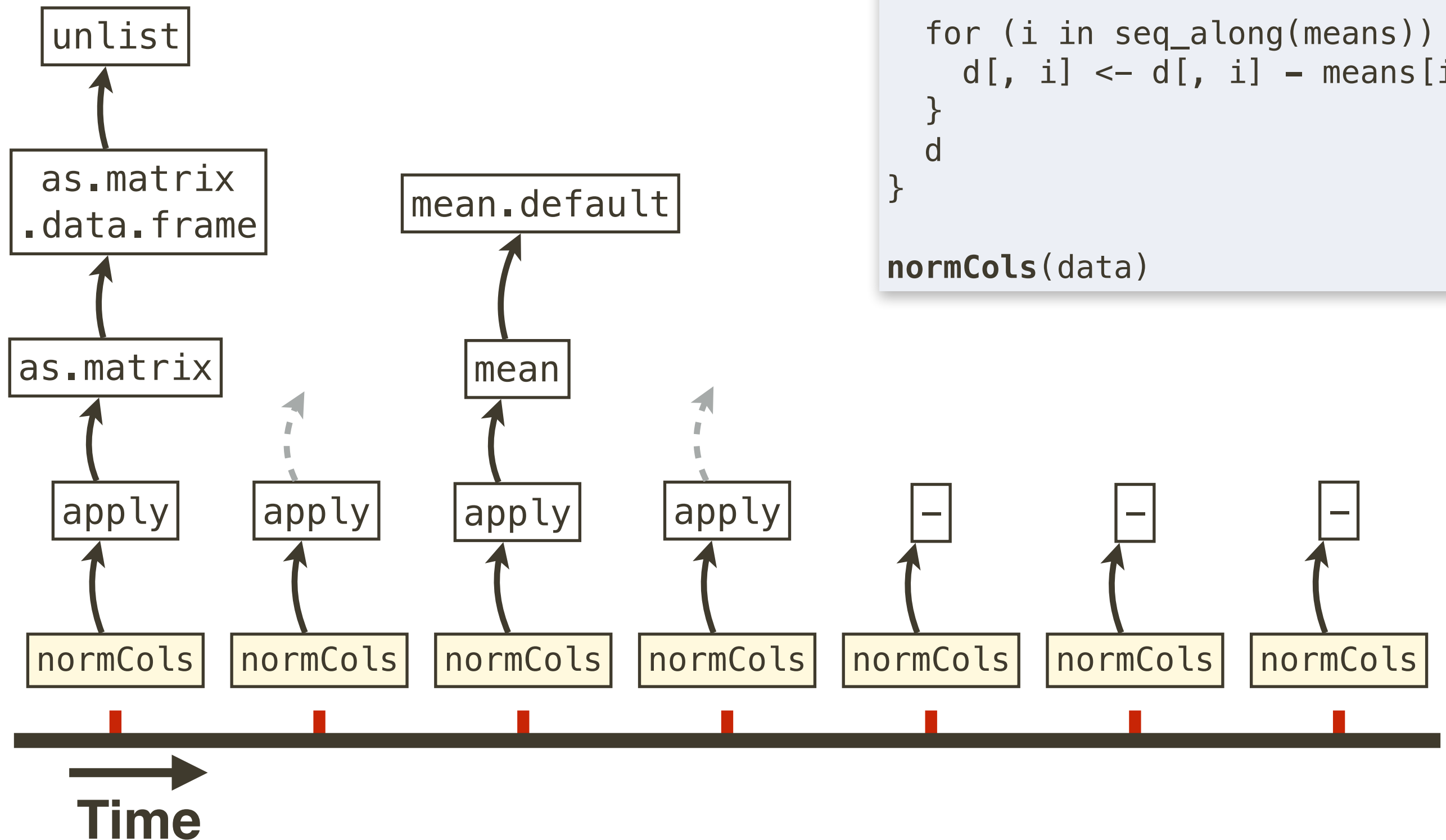


# Sampling profiler

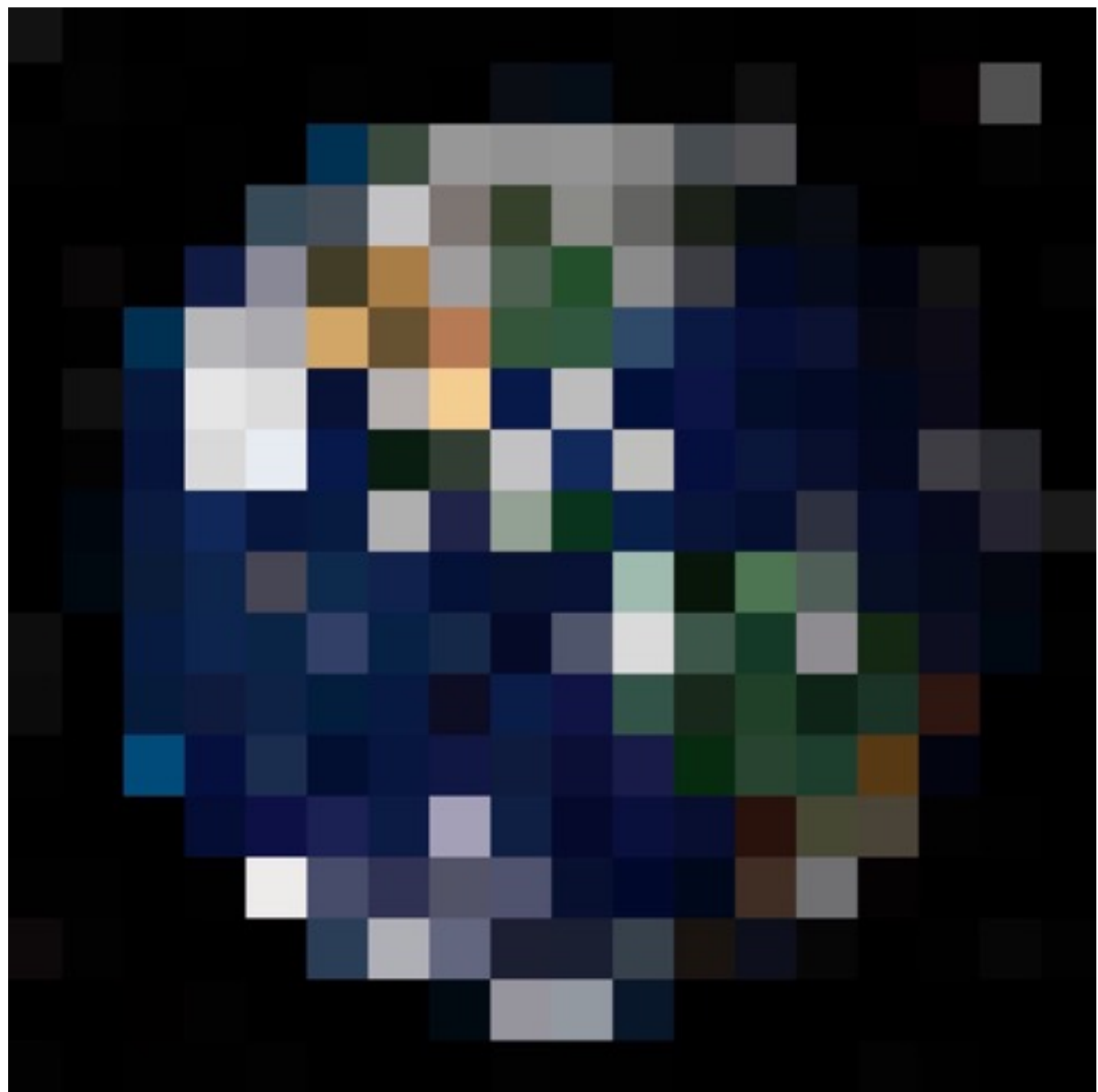
```
normCols <- function(d) {
  means <- apply(d, 2, mean)

  for (i in seq_along(means)) {
    d[, i] <- d[, i] - means[i]
  }
  d
}

normCols(data)
```







**profvis**



# Getting started

```
install.packages("profvis")  
library(profvis)
```

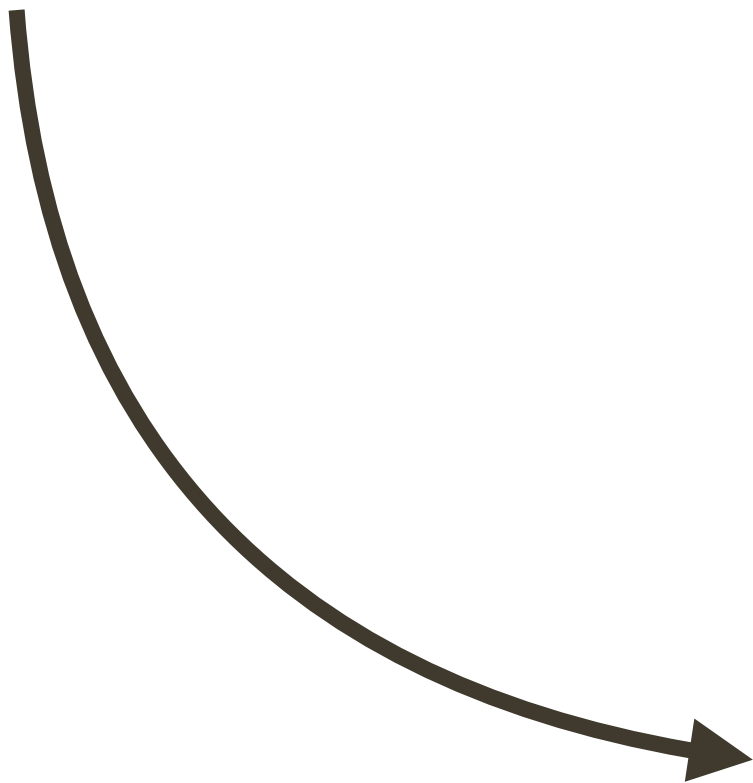
```
profvis({  
  normCols <- function(d) {  
    means <- apply(d, 2, mean)  
  
    for (i in seq_along(means)) {  
      d[, i] <- d[, i] - means[i]  
    }  
    d  
  }  
  
  normCols(data)  
})
```

```
profvis({  
  # Four different ways of getting column means  
  means <- apply(d, 2, mean)  
  means <- colMeans(d)  
  means <- lapply(d, mean)  
  means <- vapply(d, mean, numeric(1))  
})
```

```
profvis({  
  d <- data  
  means <- vapply(d, mean, numeric(1))  
  
  for (i in seq_along(means)) {  
    d[, i] <- d[, i] - means[i]  
  }  
})
```

# Example: Text processing

```
"unlist" "as.matrix.data.frame" "as.matrix" "apply"  
"unlist" "as.matrix.data.frame" "as.matrix" "apply"  
"aperm.default" "aperm" "apply"  
"aperm.default" "aperm" "apply"  
[... 750 lines ...]
```



row	col	label
1	4	"unlist"
1	3	"as.matrix.data.frame"
1	2	"as.matrix"
1	1	"apply"
2	4	"unlist"
2	3	"as.matrix.data.frame"
2	2	"as.matrix"
2	1	"apply"
3	3	"aperm.default"
3	2	"aperm"
3	1	"apply"
4	3	"aperm.default"
4	2	"aperm"
4	1	"apply"



```
lines <- readLines("output.prof")
```

```
proc_lines <- list()
```

For each line...

```
for (i in seq_along(lines)) {  
  line <- lines[i]  
  line <- strsplit(line, " ")[[1]]
```

```
  linedata <- data.frame(  
    row = i,  
    col = rev(seq_along(line)),  
    label = line  
  )
```

Create a data frame from input

Store the data frame  
in a list

```
  proc_lines[[i]] <- linedata  
}
```

```
proc_data <- do.call(rbind, proc_lines)
```

Combine all data  
frames together

```
lines <- readLines("ou
```

List is not preallocated

```
proc_lines <- list()
```

for loop

```
for (i in seq_along(lines)) {  
  line <- lines[i]  
  line <- strsplit(line, " ")[[1]]
```

```
  linedata <- data.frame(  
    row = i,  
    col = rev(seq_along(line)),  
    label = line  
  )
```

```
proc_lines[[i]] <- linedata  
}
```

Growing list in a loop

```
proc_data <- do.call(rbind, proc_lines)
```

```
lines <- readLines("output.prof")
```

```
proc_lines <- list()
```

```
for (i in seq_along(lines)) {  
  line <- lines[i]  
  line <- strsplit(line, " ")[[1]]
```

```
  linedata <- list(  
    row = rep(i, length(line)),  
    col = rev(seq_along(line)),  
    label = line  
  )  
  
  proc_lines[[i]] <- linedata  
}
```

Put line data in a list  
instead of a data frame

```
extract_vector <- function(x, name) {  
  vecs <- lapply(x, `[`, name)  
  do.call(c, vecs)  
}
```

```
proc_data <- data.frame(  
  row = extract_vector(proc_lines, "row"),  
  col = extract_vector(proc_lines, "col"),  
  label = extract_vector(proc_lines, "label")  
)
```

Create a data frame directly,  
instead of calling rbind

# profvis with RStudio

- Need a recent daily build of RStudio
- Profiling menu
- Saving
- Publishing
- Profvis can work without RStudio

# Things to remember

- Understand how the sampling profiler works
- Understand profvis interface
- Sometimes performance bottlenecks are counterintuitive

**[rstudio.github.io/profvis/](https://rstudio.github.io/profvis/)**