An overview of the R programming environment

Deepayan Sarkar

Software for Statistics

- Computing software is essential for modern statistics
  - Large datasets
  - Visualization
  - Simulation
  - Iterative methods
- Many softwares are available
- I will talk about R
  - Available for Free (Open Source)
  - Very popular (both academia and industry)
  - Easy to try out on your own

Outline

- Some examples
- A little bit of history
- Some thoughts on why R has been successful

Before we start, an experiment!

Color combination: Is it **white & gold** or **blue & black** ? Let’s count!

Question: What proportion of population sees white & gold?

- Statistics uses data to make inferences
- Model:
  - Let p be the probability of seeing white & gold
  - Assume that individuals are independent
- Data:
  - Suppose X out of N sampled individuals see white & gold; e.g., $N = 45$, $X = 26$.
  - According to model, $X \sim Bin(N, p)$
- “Obvious” estimate of p = X / N = 26 / 45 = 0.58
- But how is this estimate derived?

Generally useful method: maximum likelihood

- Likelihood function: probability of observed data as function of p

$$L(p) = P(X = 26) = \binom{45}{26} p^{26}(1-p)^{(45-26)}, p \in (0,1)$$

- Intuition: p that gives higher $L(p)$ is more “likely” to be correct
- Maximum likelihood estimate $\hat{p} = \arg\max L(p)$
- By differentiating

$$\log L(p) = c + 26 \log p + 19 \log(1-p)$$

we get

$$\frac{d}{dp} \log L(p) = \frac{26}{p} - \frac{19}{1-p} = 0 \implies 26(1-p) - 19p = 0 \implies p = \frac{26}{45}$$

How could we do this numerically?

- Pretend for the moment that we did not know how to do this. How could we arrive at the same solution numerically?
- Basic idea: Compute $L(p)$ for various values of p and find minimum.
- To do this in R, the most important thing to understand is that R works like a calculator:
  - The user types in an expression, R calculates the answer
  - The expression can involve numbers, variables, and functions
- For example:

```r
N = 45
x = 26
p = 0.5
choose(N, x) * p^x * (1-p)^(N-x)
```

```
# [1] 0.06930242
```

"Vectorized" computations

- One distinguishing feature of R is that it operates on "vectors"

```
pvec = seq(0, 1, by = 0.01)
pvec
```

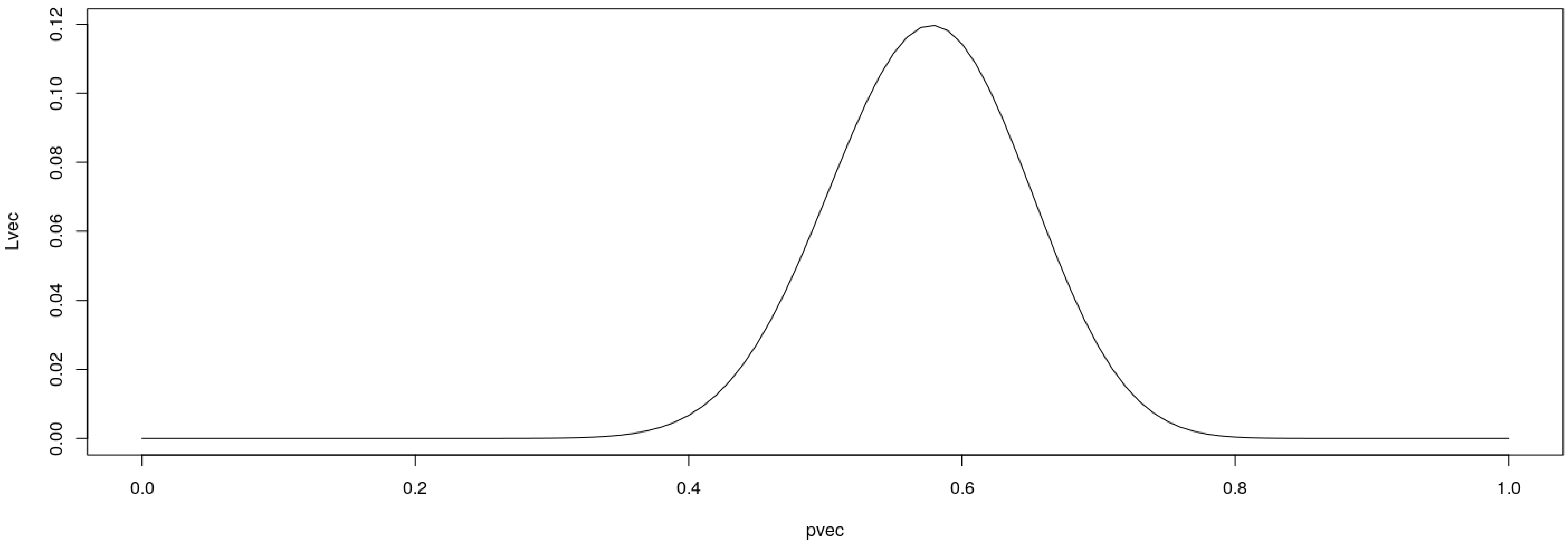
```
#   [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19 0.20 0.21 0.22
#  [24] 0.23 0.24 0.25 0.26 0.27 0.28 0.29 0.30 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.40 0.41 0.42 0.43 0.44 0.45
#  [47] 0.46 0.47 0.48 0.49 0.50 0.51 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59 0.60 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68
#  [70] 0.69 0.70 0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.80 0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.90 0.91
#  [93] 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99 1.00
```

```
Lvec = choose(N, x) * pvec^x * (1-pvec)^(N-x)
Lvec
```

```
#   [1] 0.000000e+00 2.014498e-40 1.114740e-32 3.474672e-28 5.056051e-25 1.371093e-22 1.283689e-20 5.765318e-19
#   [9] 1.511495e-17 2.625366e-16 3.293866e-15 3.174813e-14 2.460262e-13 1.586687e-12 8.747777e-12 4.211439e-11
#  [17] 1.801043e-10 6.938314e-10 2.435828e-09 7.868776e-09 2.358239e-08 6.602594e-08 1.737342e-07 4.318627e-07
#  [25] 1.018706e-06 2.289299e-06 4.918220e-06 1.013189e-05 2.006894e-05 3.831376e-05 7.065023e-05 1.260767e-04
#  [33] 2.181057e-04 3.663379e-04 5.982529e-04 9.510890e-04 1.473611e-03 2.227478e-03 3.287864e-03 4.742910e-03
#  [41] 6.691627e-03 9.239888e-03 1.249429e-02 1.655390e-02 2.150009e-02 2.738512e-02 3.422026e-02 4.196469e-02
#  [49] 5.051658e-02 5.970760e-02 6.930242e-02 7.900386e-02 8.846442e-02 9.730387e-02 1.051320e-01 1.115747e-01
#  [57] 1.163022e-01 1.190543e-01 1.196637e-01 1.180712e-01 1.143327e-01 1.086179e-01 1.011977e-01 9.242411e-02
#  [65] 8.270372e-02 7.246667e-02 6.213552e-02 5.209643e-02 4.267559e-02 3.412296e-02 2.660425e-02 2.020120e-02
#  [73] 1.491921e-02 1.070050e-02 7.440747e-03 5.006696e-03 3.252859e-03 2.035570e-03 1.223457e-03 7.039944e-04
#  [81] 3.863739e-04 2.013850e-04 9.918271e-05 4.588367e-05 1.979882e-05 7.901767e-06 2.887291e-06 9.539431e-07
#  [89] 2.806024e-07 7.206085e-08 1.575446e-08 2.836495e-09 4.020606e-10 4.212284e-11 2.973693e-12 1.225581e-13
#  [97] 2.318943e-15 1.283711e-17 7.560423e-21 1.877644e-26 0.000000e+00
```

Plotting is very easy

```
plot(x = pvec, y = Lvec, type = "l")
```



Functions

- Functions can be used to encapsulate repetitive computations
- Like mathematical functions, they take arguments as input and "returns" an output

```
L = function(p) choose(N, x) * p^x * (1-p)^(N-x)
L(0.5)
```

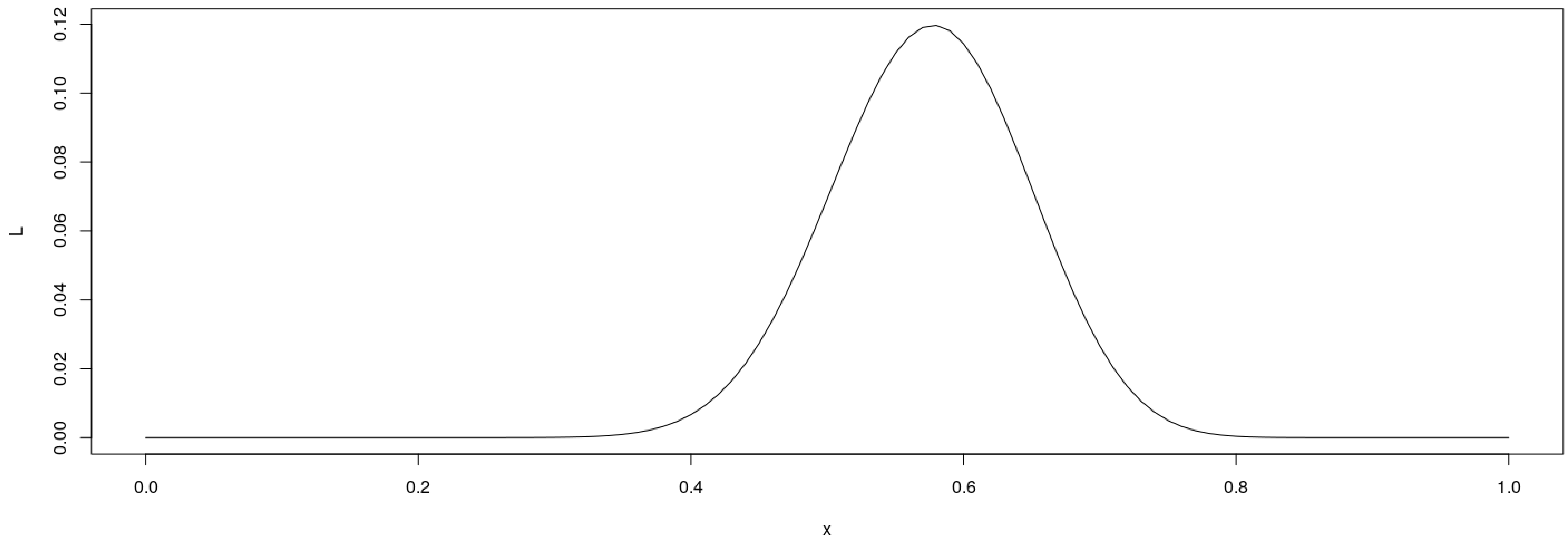
```
# [1] 0.06930242
```

```
L(26/45)
```

```
# [1] 0.1197183
```

Functions can be plotted directly

```r
plot(L, from = 0, to = 1)
```



…and they can be numerically "optimized"

```
optimize(L, interval = c(0, 1), maximum = TRUE)
```

```
# $maximum
# [1] 0.5777774
#
# $objective
# [1] 0.1197183
```

A more complicated example

- Suppose $X_1, X_2, \ldots, X_n \sim Bin(N, p)$, and are independent
- Instead of observing each X_i, we only get to know $M = \max(X_1, X_2, \ldots, X_n)$
- What is the maximum likelihood estimate of p? (N and n are known, $M = m$ is observed)

A more complicated example

To compute likelihood, we need p.m.f. of M :

$$P(M \leq m) = P(X_1 \leq m, \ldots, X_n \leq m) = \left[\sum_{x=0}^{m} \binom{N}{x} p^x (1-p)^{(N-x)}\right]^n$$

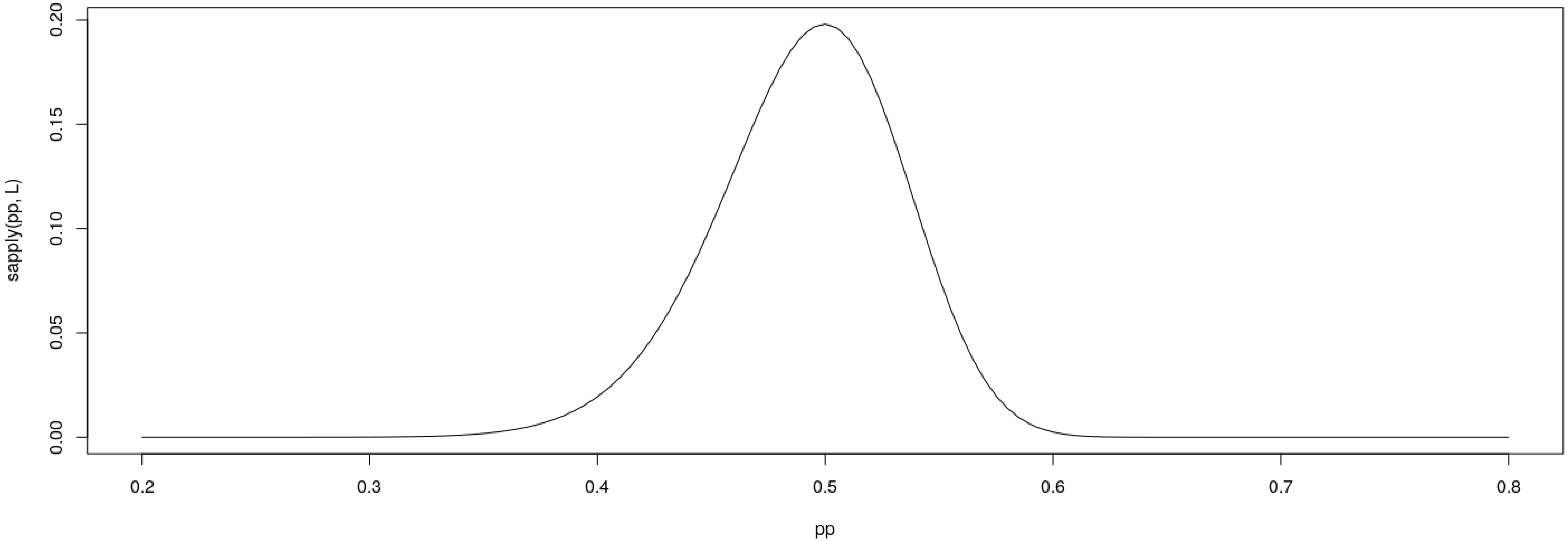
and

$$P(M = m) = P(M \leq m) - P(M \leq m-1)$$

In R,

```
n = 10
N = 50
M = 30
F <- function(p, m)
{
    x = seq(0, m)
    (sum(choose(N, x) * p^x * (1-p)^(N-x)))^n
}
L = function(p)
{
    F(p, M) - F(p, M-1)
}
```

Maximum Likelihood estimate

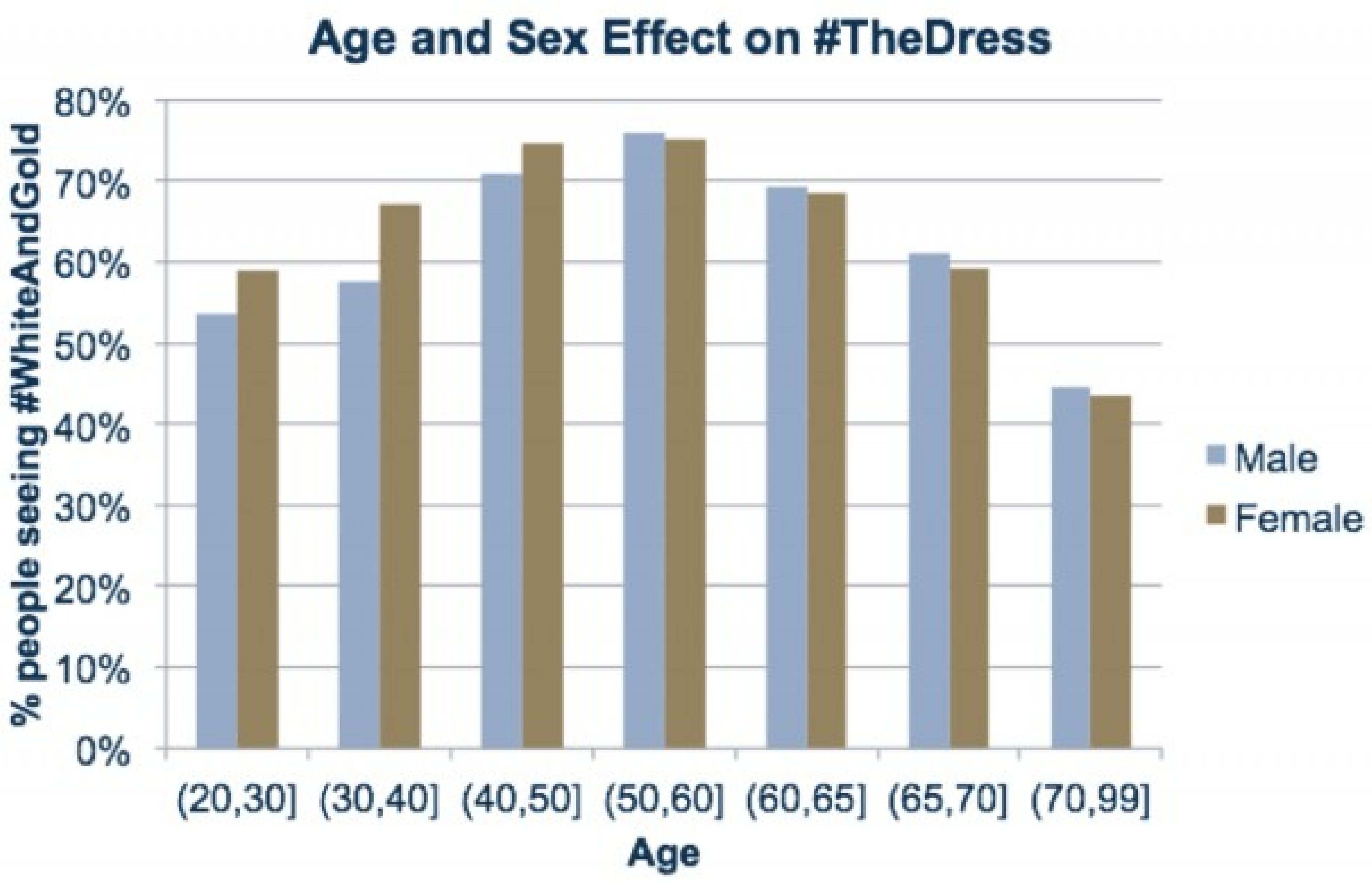



```r
optimize(L, interval = c(0, 1), maximum = TRUE)
```

```
# $maximum
# [1] 0.4996703
#
# $objective
# [1] 0.1981222
```

“The Dress” revisited

- What factors determine perceived color? (From 23andme.com)



Simulation: birthday problem

- R can be used to simulate random events
- Example: how likely is a common birthday in a group of 20 people?

```r
N = 20
days = sample(365, N, rep = TRUE)
days
```

```
#  [1]  65  34 194 198 111  45 192 133  24 188   1   7  18 202 355 178 275  78 261  51
```

```r
length(unique(days))
```

```
# [1] 20
```

Law of Large Numbers

- With enough replications, sample proportion should converge to probability

```
haveCommon = function()
{
    days = sample(365, N, rep = TRUE)
    length(unique(days)) < N
}
haveCommon()
```

```
# [1] FALSE
```

```
haveCommon()
```

```
# [1] FALSE
```

```
haveCommon()
```

```
# [1] TRUE
```

```
haveCommon()
```

```
# [1] TRUE
```

Law of Large Numbers

- With enough replications, sample proportion should converge to probability
- Do this sytematically:

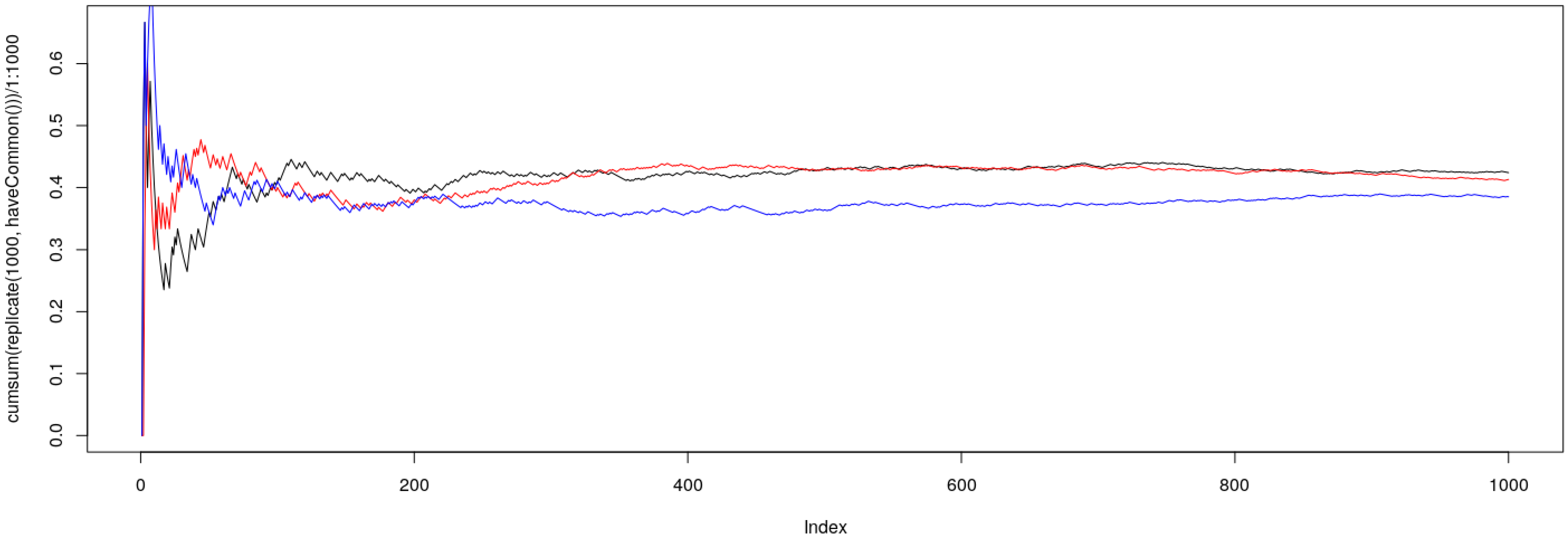
```
replicate(100, haveCommon())
```

```
#   [1]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE
#  [20]  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE
#  [39] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
#  [58]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
#  [77]  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
#  [96] FALSE  TRUE  TRUE  TRUE  TRUE
```

Law of Large Numbers

- With enough replications, sample proportion should converge to probability

```
plot(cumsum(replicate(1000, haveCommon())) / 1:1000, type = "l")
lines(cumsum(replicate(1000, haveCommon())) / 1:1000, col = "red")
lines(cumsum(replicate(1000, haveCommon())) / 1:1000, col = "blue")
```



A more serious example: climate change

Show 10 entries Search:

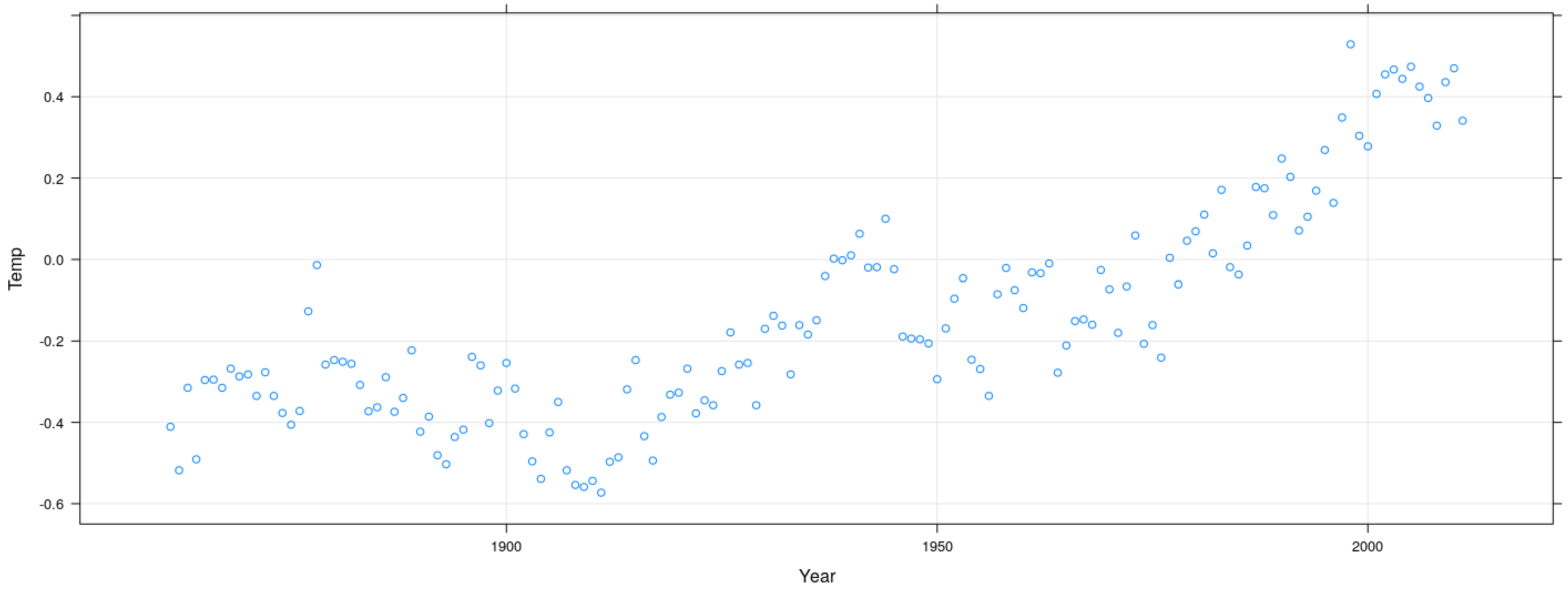
Year	Temp	CO2	CH4	NO2
1861	-0.411	286.5	838.2	288.9
1862	-0.518	286.6	839.6	288.9
1863	-0.315	286.8	840.9	289.0
1864	-0.491	287.0	842.3	289.1
1865	-0.296	287.2	843.8	289.1
1866	-0.295	287.4	845.5	289.2
1867	-0.315	287.6	847.1	289.3
1868	-0.268	287.8	848.6	289.3
1869	-0.287	288.0	850.2	289.4
1870	-0.282	288.2	851.8	289.5

Showing 1 to 10 of 151 entries

Previous 1 2 3 4 5 … 16 Next

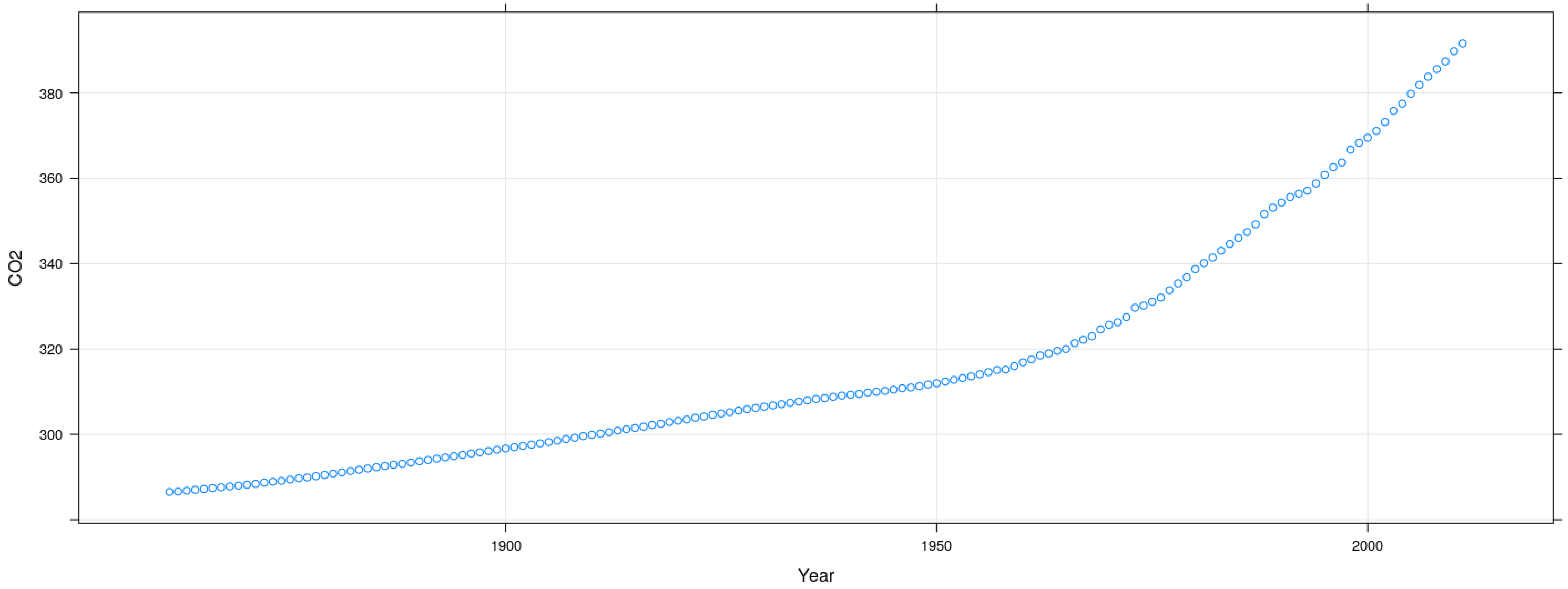
Change in temperature (global average deviation) since 1851

```r
library(lattice)
xyplot(Temp ~ Year, data = globalTemp, grid = TRUE)
```



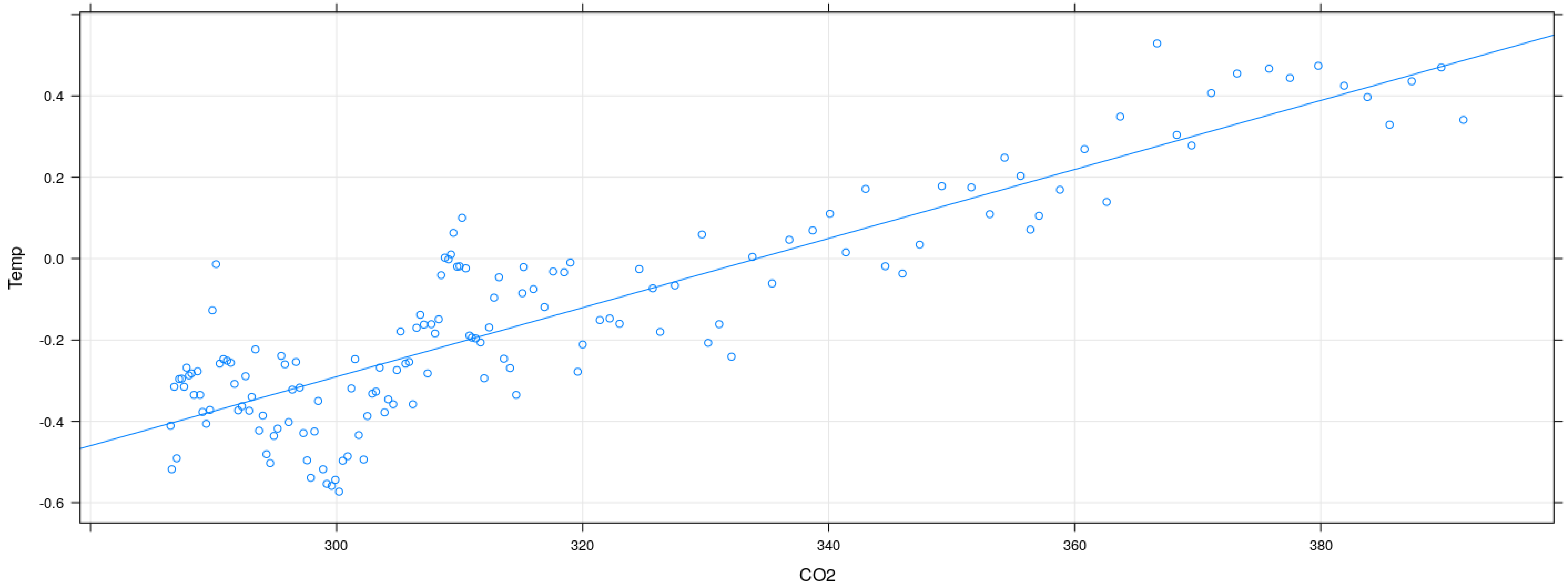
Change in atmospheric carbon dioxide

```
xyplot(CO2 ~ Year, data = globalTemp, grid = TRUE)
```



Does change in CO_2 explain temperature rise?

```
xyplot(Temp ~ CO2, data = globalTemp, grid = TRUE, type = c("p", "r"))
```



Plot includes the Least Squares regression line

Fitting the regression model

```r
fm = lm(Temp ~ 1 + CO2, data = globalTemp)
coef(fm) # estimated regression coefficients
```

```
#   (Intercept)           CO2
# -2.836082117  0.008486628
```

We can confirm using a general optimizer:

```r
SSE = function(beta)
{
    with(globalTemp,
         sum((Temp - beta[1] - beta[2] * CO2)^2))
}
optim(c(0, 0), fn = SSE)
```

```
# $par
# [1] -2.836176636  0.008486886
#
# $value
# [1] 2.210994
#
# $counts
# function gradient
#       93       NA
#
# $convergence
# [1] 0
#
# $message
# NULL
```

Fitting the regression model

`lm()` gives exact solution and more statistically relevant details

```r
summary(fm)
```

```
#
# Call:
# lm(formula = Temp ~ 1 + CO2, data = globalTemp)
#
# Residuals:
#      Min       1Q   Median       3Q      Max
# -0.28460 -0.09004 -0.00101  0.08616  0.35926
#
# Coefficients:
#               Estimate Std. Error t value Pr(>|t|)
# (Intercept) -2.8360821  0.1145766  -24.75   <2e-16 ***
# CO2          0.0084866  0.0003602   23.56   <2e-16 ***
# ---
# Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#
```

Fitting the regression model

`lm()` gives exact solution and more statistically relevant details

```
str(fm$qr)
```

```
# List of 5
#  $ qr   : num [1:151, 1:2] -12.2882 0.0814 0.0814 0.0814 0.0814 ...
#   ..- attr(*, "dimnames")=List of 2
#   .. ..$ : chr [1:151] "1" "2" "3" "4" ...
#   .. ..$ : chr [1:2] "(Intercept)" "CO2"
#   ..- attr(*, "assign")= int [1:2] 0 1
#  $ qraux: num [1:2] 1.08 1.08
#  $ pivot: int [1:2] 1 2
#  $ tol  : num 1e-07
#  $ rank : int 2
#  - attr(*, "class")= chr "qr"
```

Changing the model-fitting criteria

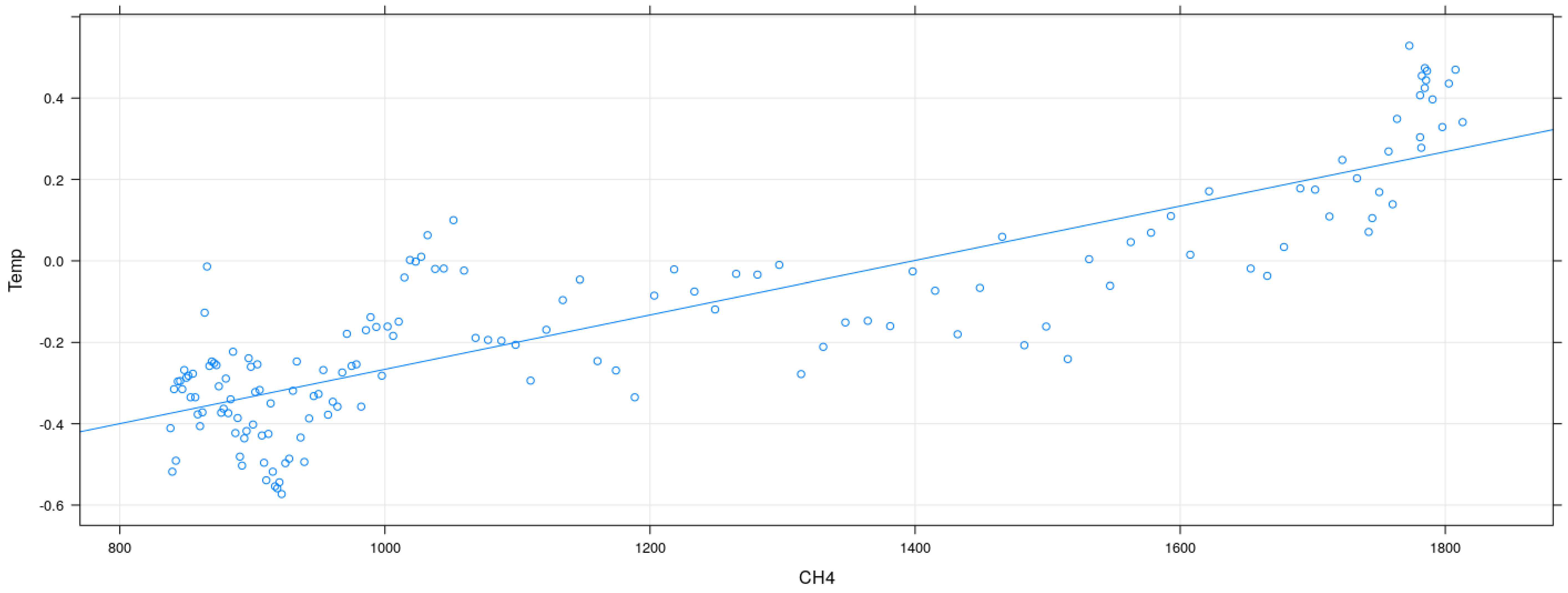
- Suppose we wanted to minimize *sum of absolute errors* instead of sum of squares
- No closed form solution any more, but general optimizer will still work:

```
SAE = function(beta)
{
    with(globalTemp,
         sum(abs(Temp - beta[1] - beta[2] * CO2)))
}
opt = optim(c(0, 0), fn = SAE)
opt
```

```
# $par
# [1] -2.832090898  0.008471257
#
# $value
# [1] 14.5602
#
# $counts
# function gradient
#      123       NA
#
# $convergence
# [1] 0
#
# $message
# NULL
```

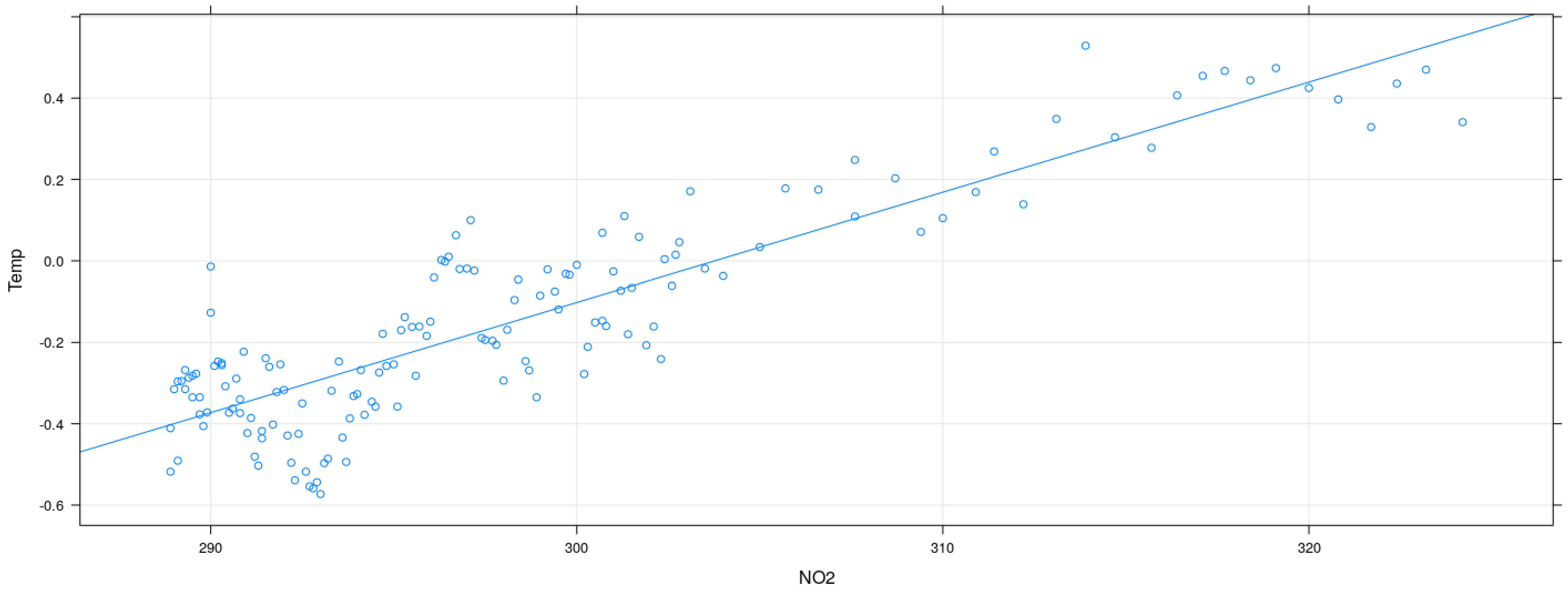
What about NH_4, NO_2?

```
xyplot(Temp ~ CH4, data = globalTemp, grid = TRUE, type = c("p", "r"))
```



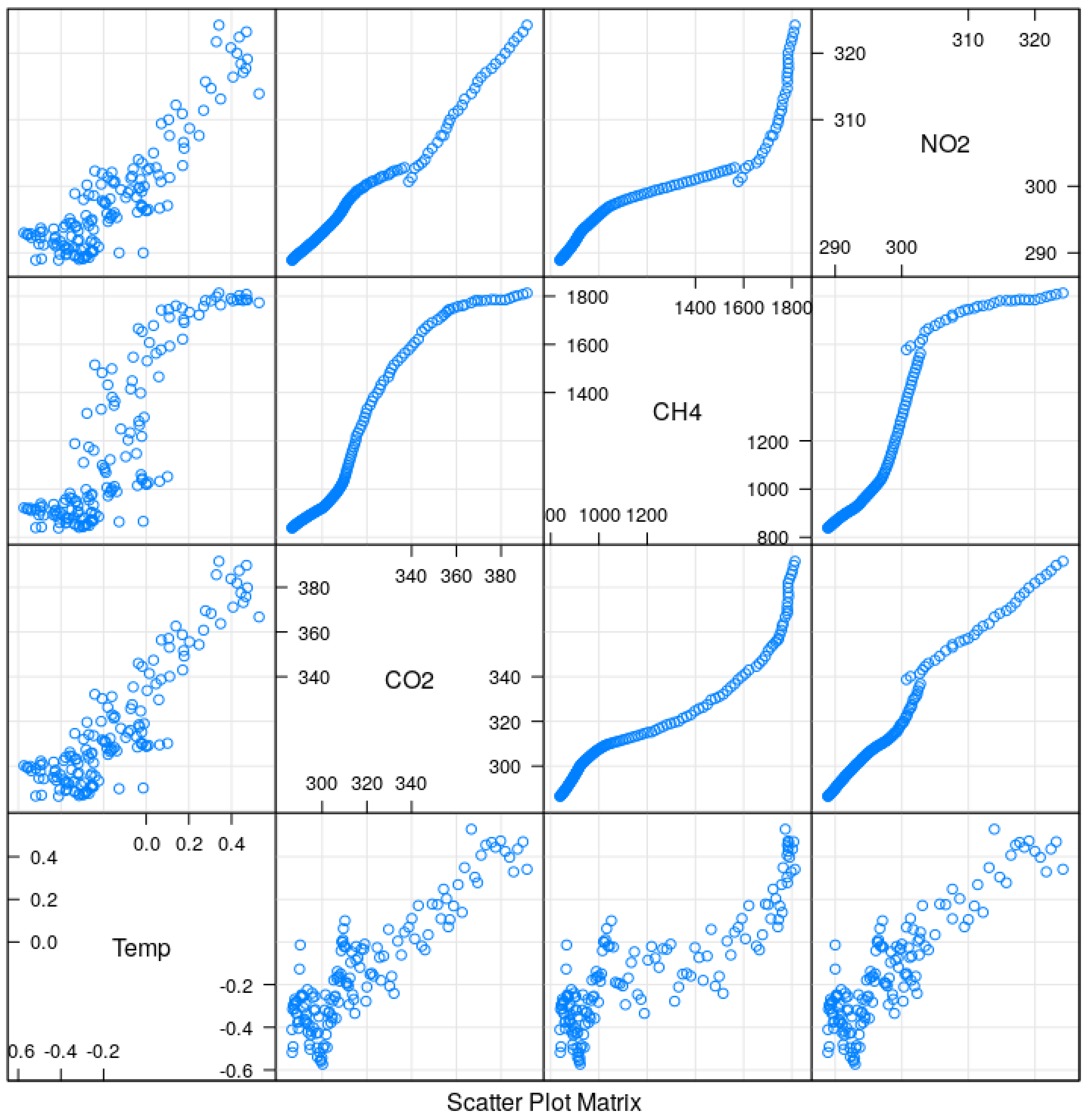
What about NH_4, NO_2?

```r
xyplot(Temp ~ NO2, data = globalTemp, grid = TRUE, type = c("p", "r"))
```



What about NH_4, NO_2? Difficult to distinguish

```r
splom(globalTemp[2:5], grid = TRUE)
```



A very brief history of R

What is R?

From its own website:

> *R is a free software environment for statistical computing and graphics.*

> *It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S.*

The origins of S

- Developed at Bell Labs (statistics research department) 1970s onwards
- Primary goals
  - Interactivity: Exploratory Data Analysis vs batch mode
  - Flexibility: Novel vs routine methodology
  - Practical: For actual use, not (just) academic research

John Chambers received the prestigious *ACM Software System Award* in 1998

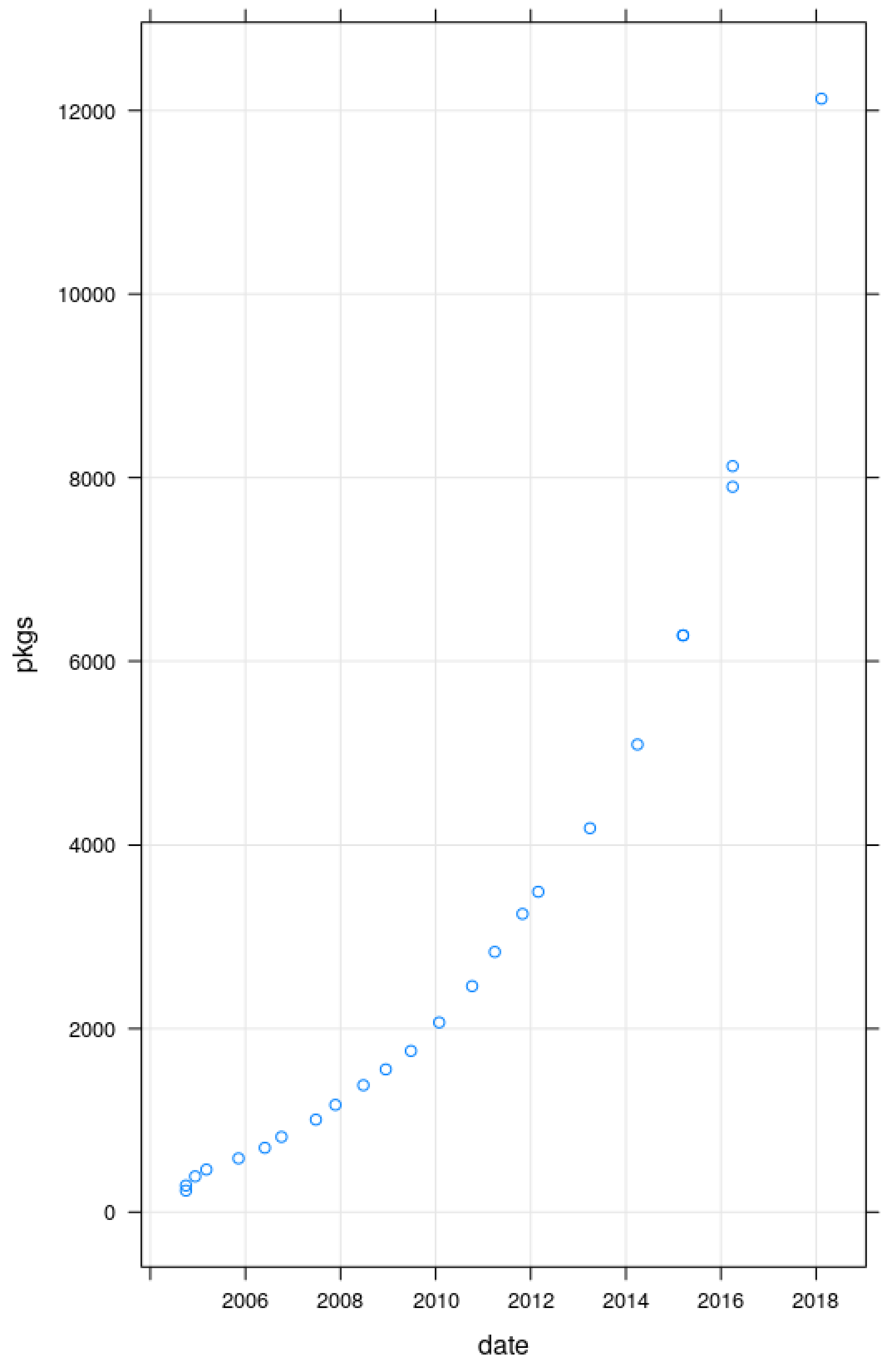
> *For The S system, which has forever altered how people analyze, visualize, and manipulate data.*

The origins of R

- Early 1990s: Started as teaching tool by Robert Gentleman & Ross Ihaka at the University of Auckland
- 1995: Convinced by Martin Mächler to release as Free Software (GPL)
- 2000: Version 1.0 released

Has since far surpassed S in popularity

Number of R packages on CRAN



Why the success? The user's perspective

- R is designed for data analysis
  - Basic data structures are vectors
  - Large collection of statistical functions
  - Advanced statistical graphics capabilities
- The vast majority of R users use it as a statistical toolbox
- R "base" comes with a large suite of statistical modeling and graphics functions
- If these are not enough, more than 10000 add-on packages are available

The developer's perspective

- Easy dissemination of research (through add-on packages)
- Rapid prototyping
- Interfaces to external software

Rapid prototyping

John Chambers, *Programming with Data*:

> *S is a programming language and environment for all kinds of computing involving data. It has a simple goal: To turn ideas into software, quickly and faithfully.*

A silly example: generate Fibonacci sequence

```
fibonacci <- function(n) {
    if (n < 2)
        x <- seq(length = n) - 1
    else {
        x <- c(0, 1)
        while (length(x) < n) {
            x <- c(x, sum(tail(x, 2)))
        }
    }
    x
}
fib10 <- fibonacci(10)
fib10
```

```
#   [1]  0  1  1  2  3  5  8 13 21 34
```

Also easy to call C for efficiency

File `fib.c`:

```c
#include <Rdefines.h>

SEXP fibonacci_c(SEXP nr)
{
    int i, n = INTEGER_VALUE(nr);
    SEXP ans = PROTECT(NEW_INTEGER(n));
    int *x = INTEGER_POINTER(ans);
    x[0] = 0; x[1] = 1;
    for (i = 2; i < n; i++) x[i] = x[i-1] + x[i-2];
    UNPROTECT(1);
    return ans;
}
```

Compile into shared library:

```
$ R CMD SHLIB fib.c
```

Load into R and call:

```r
dyn.load("fib.so")
cfib10 = .Call("fibonacci_c", as.integer(10))
cfib10
```

```
#  [1]  0  1  1  2  3  5  8 13 21 34
```

Even easier to call C++ with Rcpp package

File `fib.cpp`:

```
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
NumericVector fibonacci_cpp(int n)
{
    NumericVector x(n);
    x[0] = 0; x[1] = 1;
    for (int i = 2; i < n; i++) x[i] = x[i-1] + x[i-2];
    return x;
}
```

Compile and call:

```
Rcpp::sourceCpp("fib.cpp")
fibonacci_cpp(10)
```

```
#  [1]  0  1  1  2  3  5  8 13 21 34
```

Summary

- Strengths of R: flexibility and extensibility
  - Powerful built-in tools
  - Programming language
  - Compiled code for efficiency
- Further demos of interfaces (if time)

Parting comments: reproducible documents

- Creating reports / presentations with numerical analysis is usually a two-step process:
  - Do the analysis using a computational software
  - Write report in a word processor, copy-pasting results
- R makes it very convenient to write “literate documents” that contain both analsyis code and report text
- Basic idea:
  - Start with source text file containing code+text
  - Transform file by running code and embedding results
  - Produces another text file (LaTeX, HTML, markdown)
  - Processed further using standard tools
- Example: this presentation is created from this source file (R Markdown) using knitr and pandoc
- As the source format is markdown, output could also be PDF instead of HTML