An overview of the R programming environment

Deepayan Sarkar

Indian Statistical Institute, Delhi

Software for Statistics

- Computing software is essential for modern statistics
  - Large datasets
  - Visualization
  - Simulation
  - Iterative methods
- Many softwares are available

Software for Statistics

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

Outline

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

Installing R

- R is most commonly used as a REPL (Read-Eval-Print-Loop)
- This is essentially the model used by a calculator:
  - Waits for user input
  - Evaluates and prints result
  - Waits for more input

Installing R

- R is most commonly used as a REPL (Read-Eval-Print-Loop)
- This is essentially the model used by a calculator:
  - Waits for user input
  - Evaluates and prints result
  - Waits for more input
- There are several different *interfaces* to do this
- R itself works on many platforms (Windows, Mac, UNIX, Linux)
- Some interfaces are platform-specific, some work on most

Installing R

- R is most commonly used as a REPL (Read-Eval-Print-Loop)
- This is essentially the model used by a calculator:
  - Waits for user input
  - Evaluates and prints result
  - Waits for more input
- There are several different *interfaces* to do this
- R itself works on many platforms (Windows, Mac, UNIX, Linux)
- Some interfaces are platform-specific, some work on most
- R and the interface may need to be installed separately

Installing R

- Go to https://cran.r-project.org/ (or choose a mirror first)
- Follow instructions depending on your platform (probably Windows)

Installing R

- Go to https://cran.r-project.org/ (or choose a mirror first)
- Follow instructions depending on your platform (probably Windows)
- This will install R, as well as a default graphical interface on Windows and Mac

Installing R

- Go to https://cran.r-project.org/ (or choose a mirror first)
- Follow instructions depending on your platform (probably Windows)
- This will install R, as well as a default graphical interface on Windows and Mac
- I will recommend a different interface called R Studio that needs to be installed separately
- I personally use yet another interface called ESS which works with a general purpose editor called Emacs (download link for Windows)

Running R

- Once installed, you can start the appropriate interface (or R directly) to get something like this:

```
R Under development (unstable) (2018-05-05 r74699) -- "Unsuffered Consequen
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-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.

  Natural language support but running in an English locale

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.
```

Before we start, an experiment!

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

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., $N = 44$, $X = 26$.
  - According to model, $X \sim Bin(N, p)$

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., $N = 44$, $X = 26$.
  - According to model, $X \sim Bin(N, p)$
- "Obvious" estimate of $p = X/N = 26/44 = 0.5909$
- But how is this estimate derived?

Generally useful method: maximum likelihood

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

$$L(p) = P(X = 26) = \binom{44}{26} p^{26}(1-p)^{(44-26)}, p \in (0, 1)$$

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

Generally useful method: maximum likelihood

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

$$L(p) = P(X = 26) = \binom{44}{26} p^{26}(1-p)^{(44-26)}, p \in (0,1)$$

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

$$\log L(p) = c + 26 \log p + 18 \log(1-p)$$

we get

$$\frac{d}{dp} \log L(p) = \frac{26}{p} - \frac{18}{1-p} = 0 \implies 26(1-p) - 18p = 0 \implies p = \frac{26}{44}$$

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.

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

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:

```
N = 44
x = 26

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

```
[1] 0.05852204
```

"Vectorized" computations

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

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

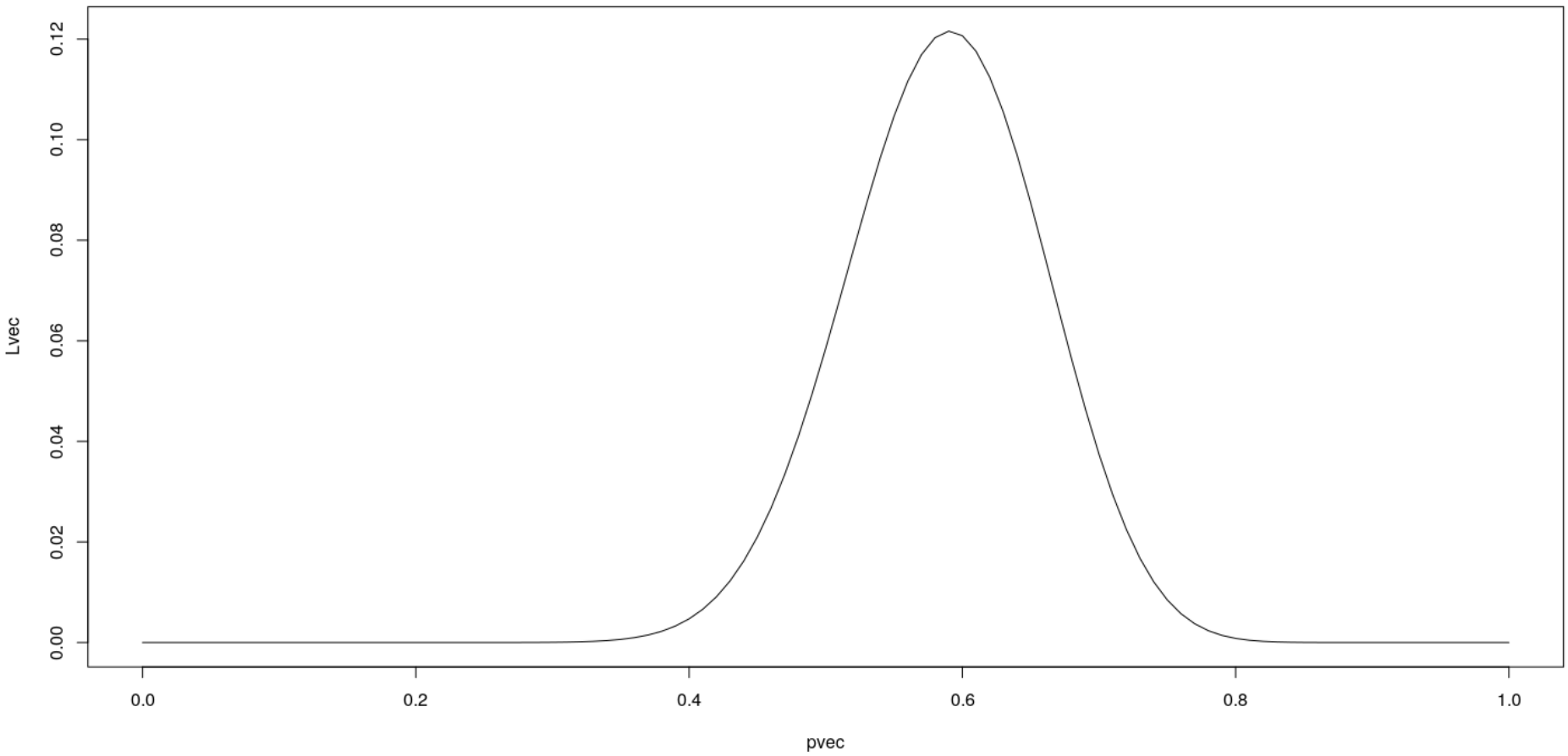
```
  [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13
 [24] 0.23 0.24 0.25 0.26 0.27 0.28 0.29 0.30 0.31 0.32 0.33 0.34 0.35 0.36
 [47] 0.46 0.47 0.48 0.49 0.50 0.51 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59
 [70] 0.69 0.70 0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.80 0.81 0.82
 [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 8.591575e-41 4.802734e-33 1.512457e-28 2.223726e-25 6.09
  [9] 6.936811e-18 1.218119e-16 1.545270e-15 1.506153e-14 1.180429e-13 7.70
 [17] 9.052864e-11 3.529530e-10 1.254220e-09 4.101694e-09 1.244626e-08 3.52
 [25] 5.659476e-07 1.288790e-06 2.806191e-06 5.860149e-06 1.176882e-05 2.27
 [33] 1.354251e-04 2.308597e-04 3.827207e-04 6.178014e-04 9.721737e-04 1.49
 [41] 4.708923e-03 6.612349e-03 9.095461e-03 1.226215e-02 1.621039e-02 2.10
 [49] 4.101773e-02 4.943113e-02 5.852204e-02 6.807589e-02 7.781593e-02 8.74
 [57] 1.116031e-01 1.169009e-01 1.202969e-01 1.215909e-01 1.206845e-01 1.17
 [65] 9.699819e-02 8.742011e-02 7.716176e-02 6.665536e-02 5.630807e-02 4.64
 [73] 2.249722e-02 1.673329e-02 1.208326e-02 8.455753e-03 5.722622e-03 3.73
```

Plotting is very easy

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



Functions

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

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

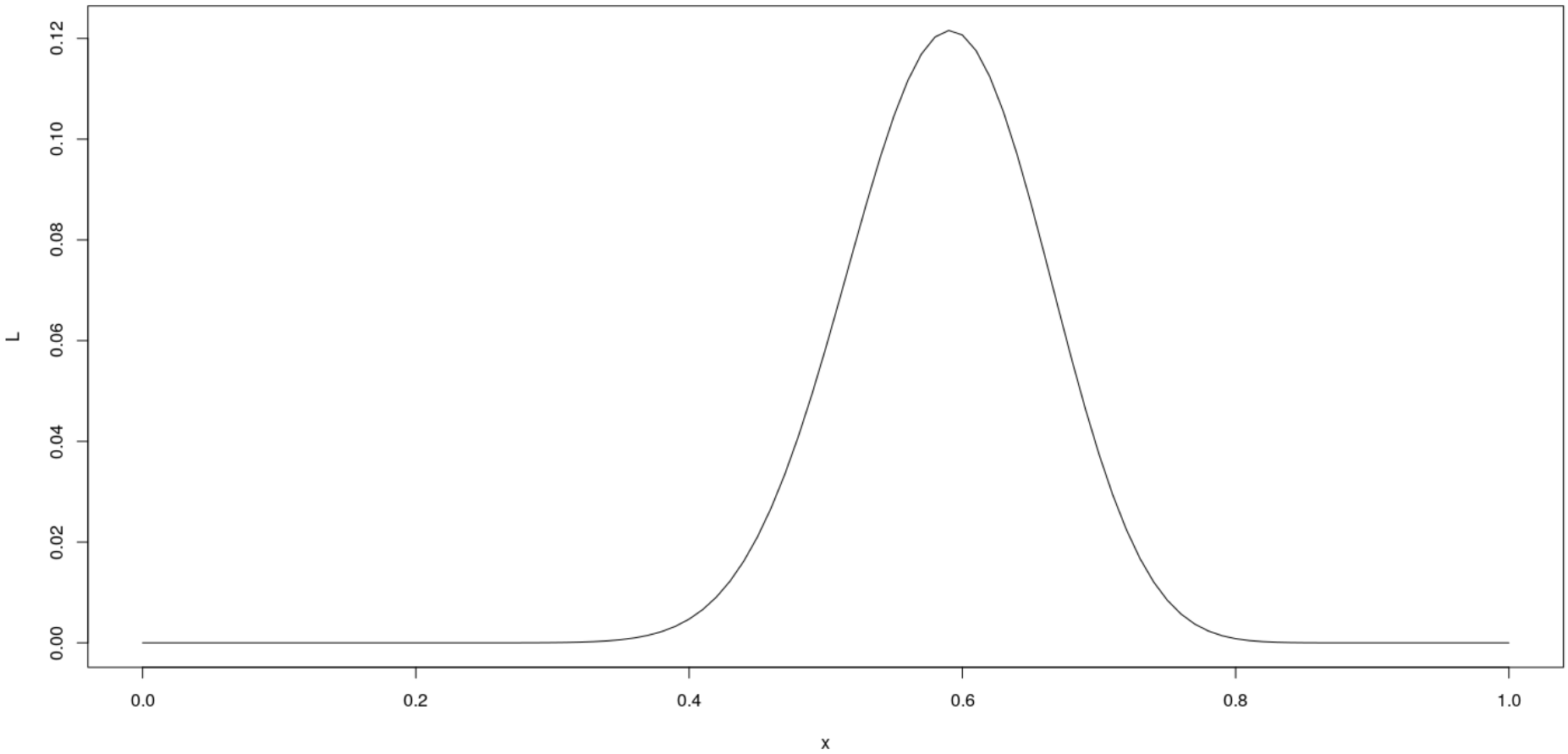
```
[1] 0.05852204
```

```
L(x/N)
```

```
[1] 0.1216
```

Functions can be plotted directly

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



. . . and they can be numerically "optimized"

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

```
$maximum
[1] 0.5909084

$objective
[1] 0.1216
```

- Compare with

```
x / N
```

```
[1] 0.5909091
```

A more complicated example

- Suppose $X_1, X_2, ..., X_n \sim Bin(N, p)$, and are independent
- Instead of observing each X_i, we only get to know $M = \max(X_1, X_2, ..., X_n)$
- 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 :

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

and

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

A more complicated example

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

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

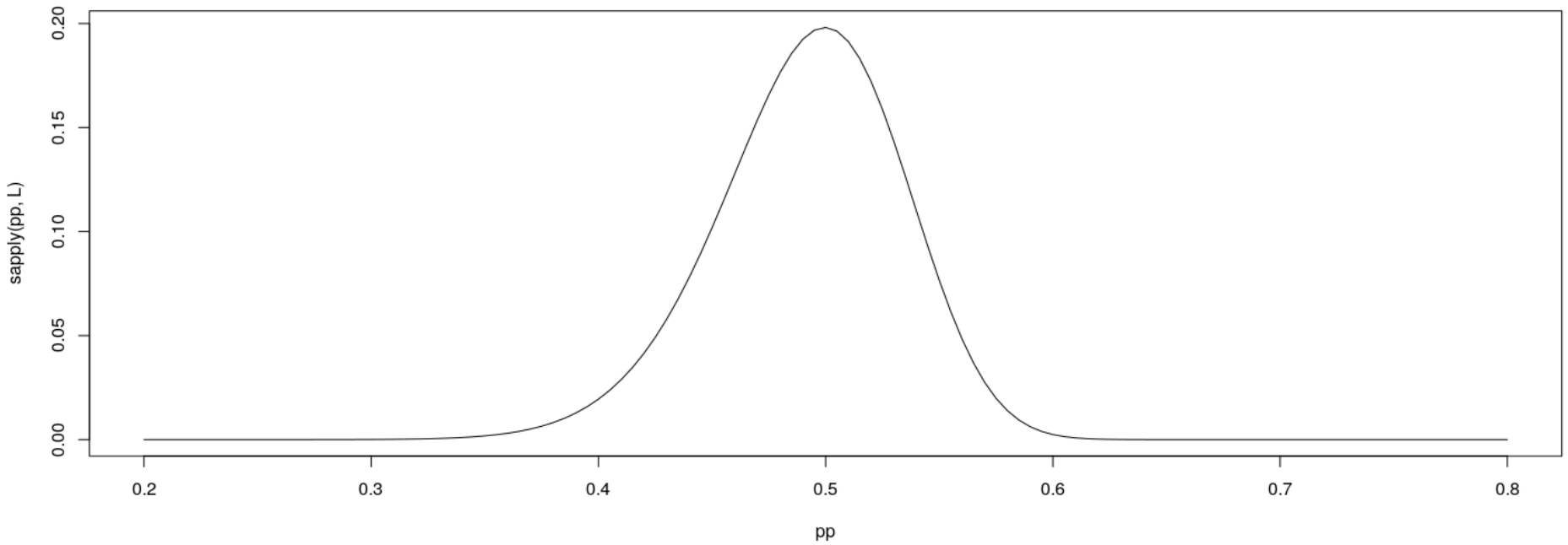
and

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

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

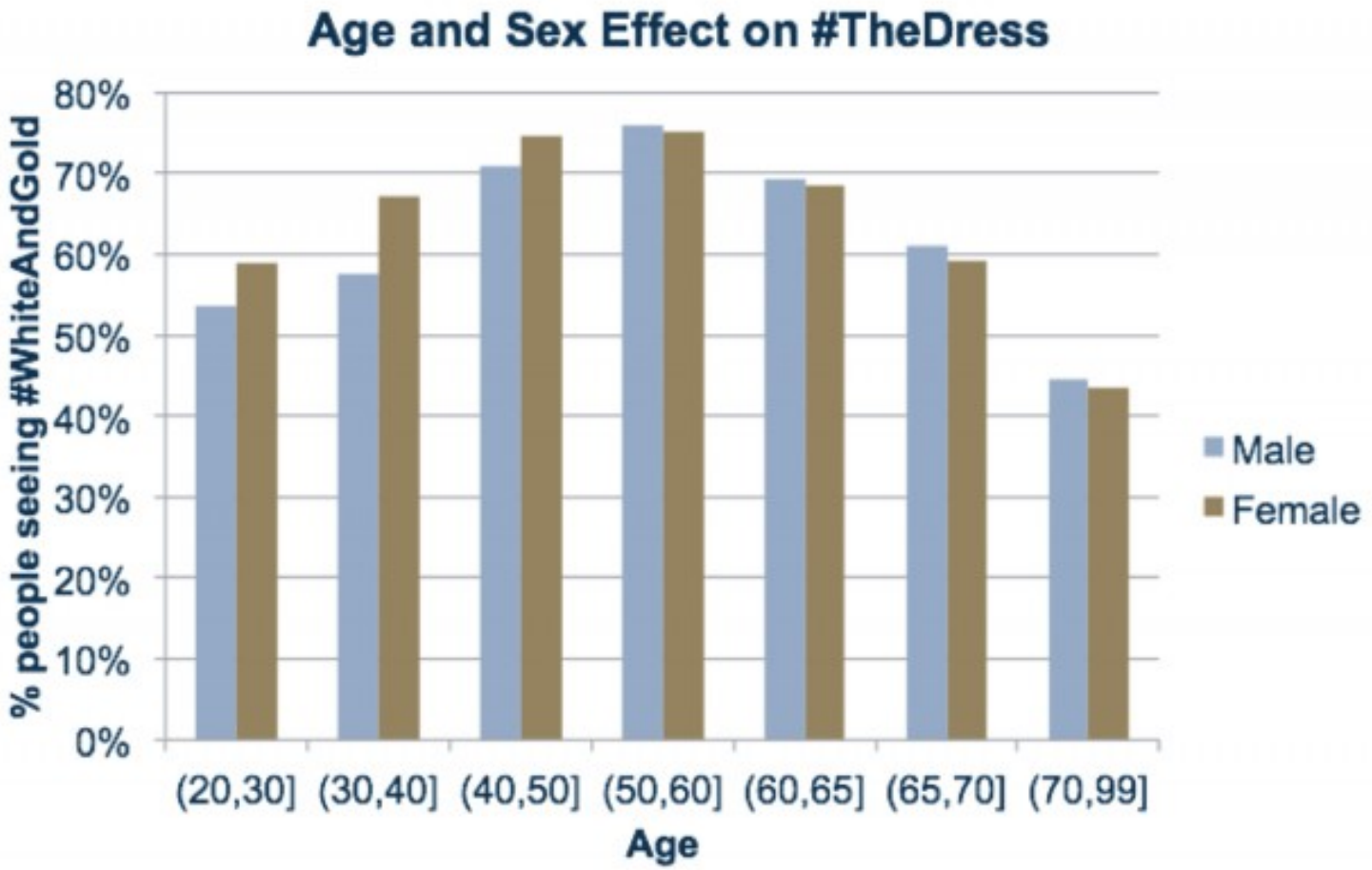



```
optimize(L, interval = c(0, 1), maximum = TRUE)
```

```
$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?

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

```
 [1] 112 320  19  42  66  41  73 182 314 266 154 313 351 276 218 359 257 24
```

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

```
[1] 19
```

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:

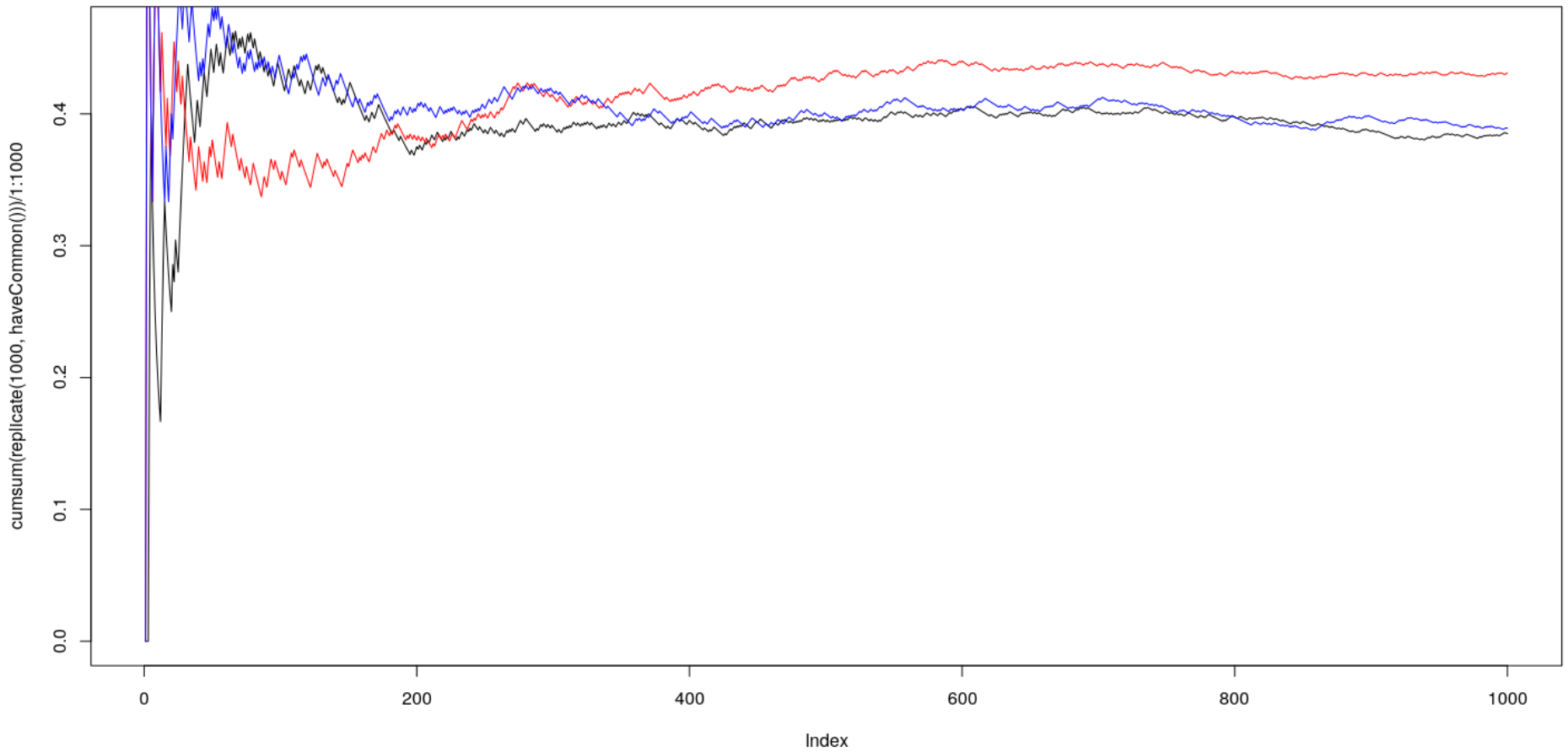
```
replicate(100, haveCommon())
```

```
  [1] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FAL
 [20]  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE  TR
 [39]  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FAL
 [58]  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FAL
 [77] FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FAL
 [96] FALSE  TRUE FALSE FALSE FALSE
```

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

Year

Temp

CO2

CH4

NO2

1861

-0.411

286.5

838.2

288.9

1862

-0.518

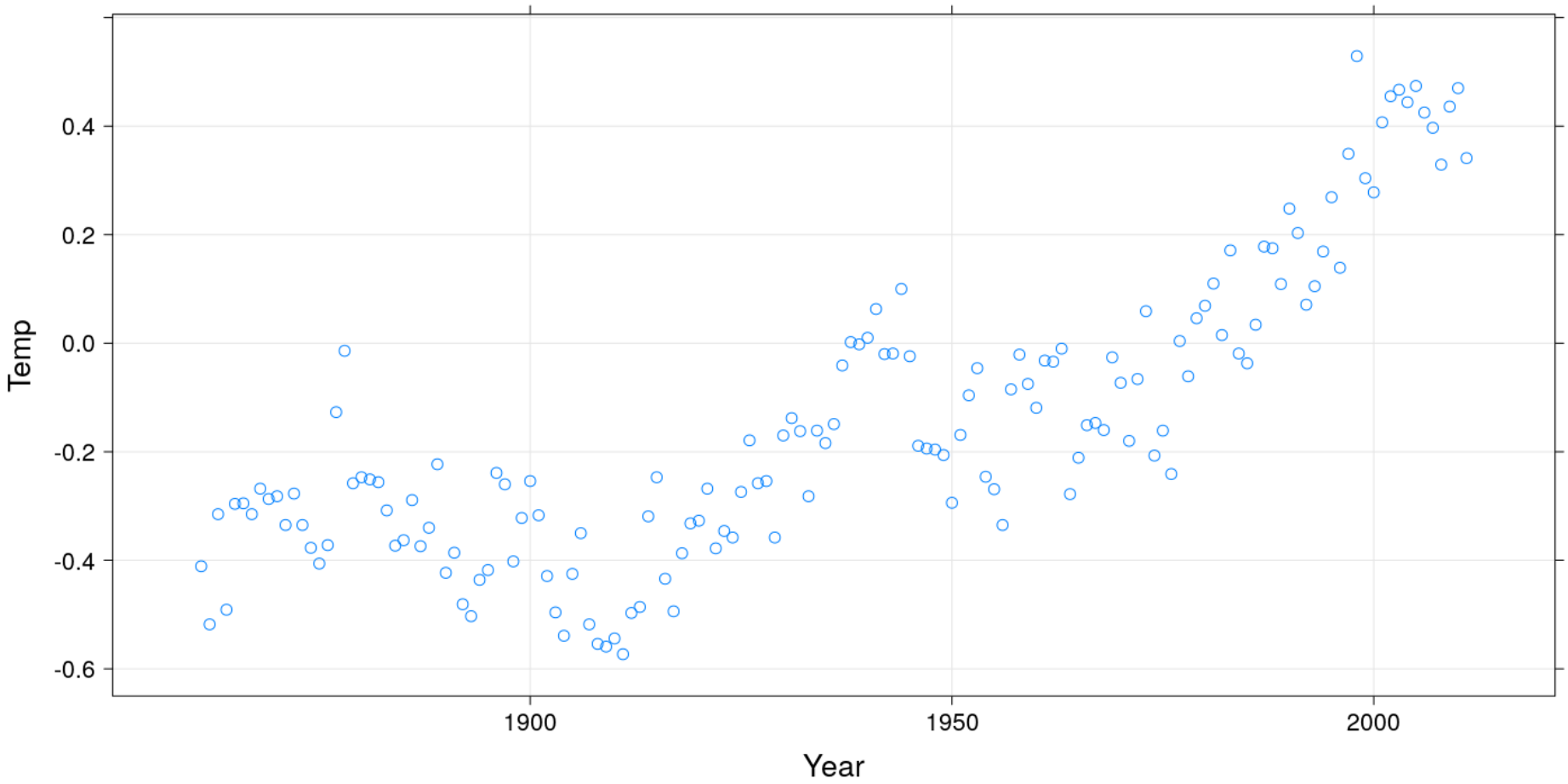
286.6

839.6

288.9

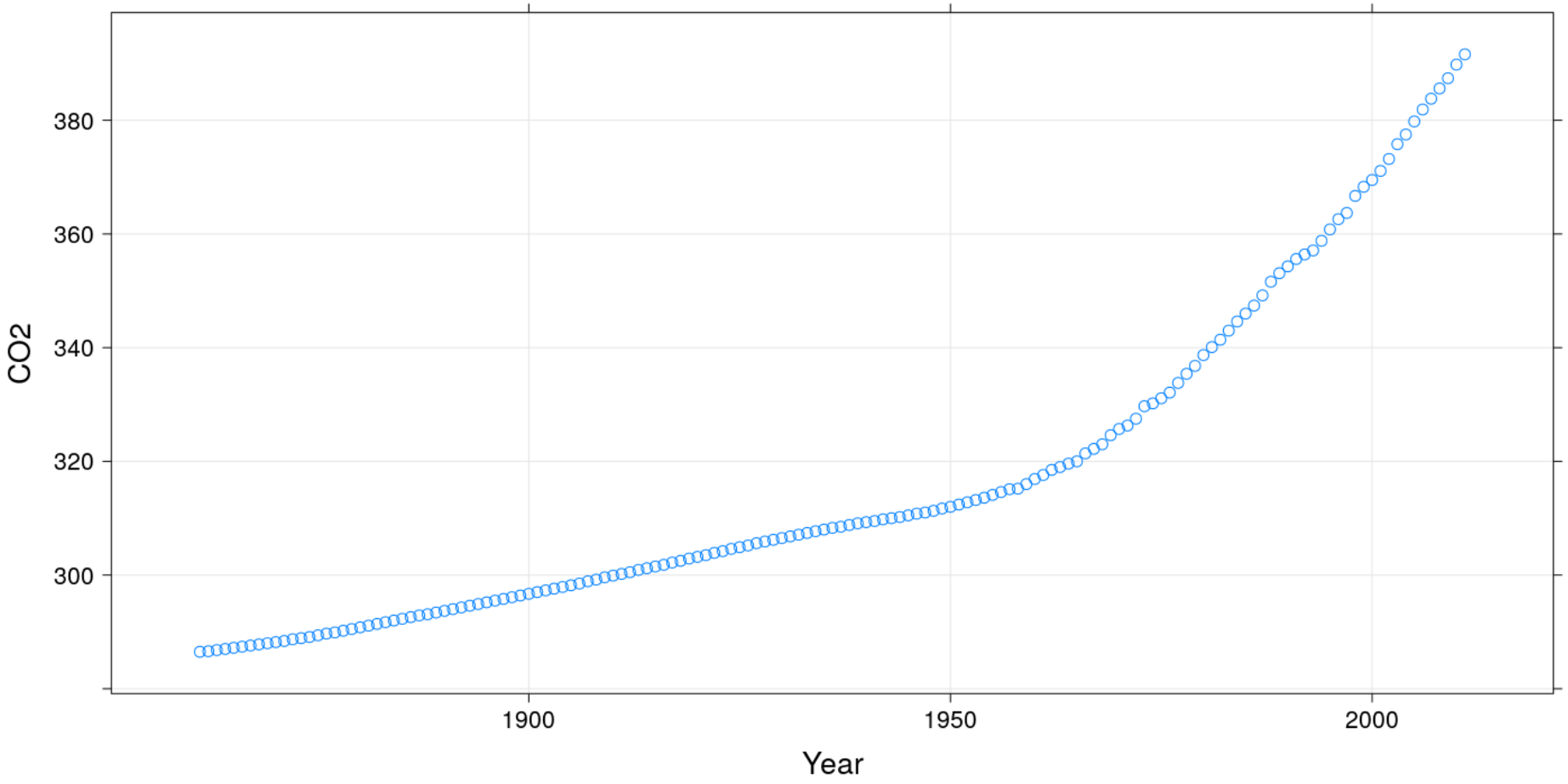
Change in temperature (global average deviation) since 1851

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



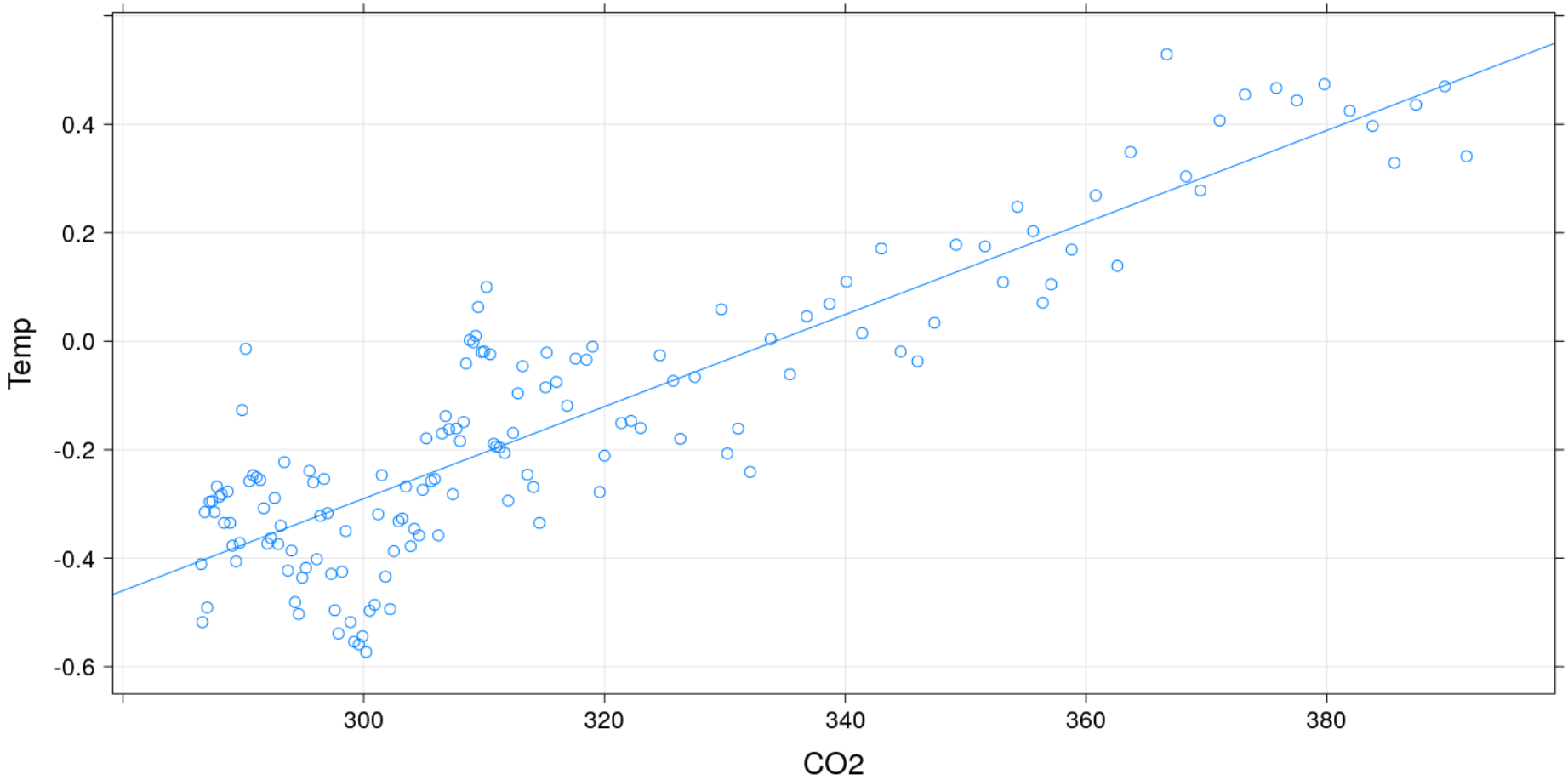
Change in atmospheric carbon dioxide

```
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")) # in
```



Fitting the regression model

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

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

Fitting the regression model

```
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:

```
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
```

Fitting the regression model

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

```
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

Residual standard error: 0.1218 on 149 degrees of freedom
Multiple R-squared:  0.7884,    Adjusted R-squared:  0.787
F-statistic: 555.1 on 1 and 149 DF,  p-value: < 2.2e-16
```

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
```

Changing the model-fitting criteria

- Compare with least squares line

```
coef(fm) # least squared errors
```

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

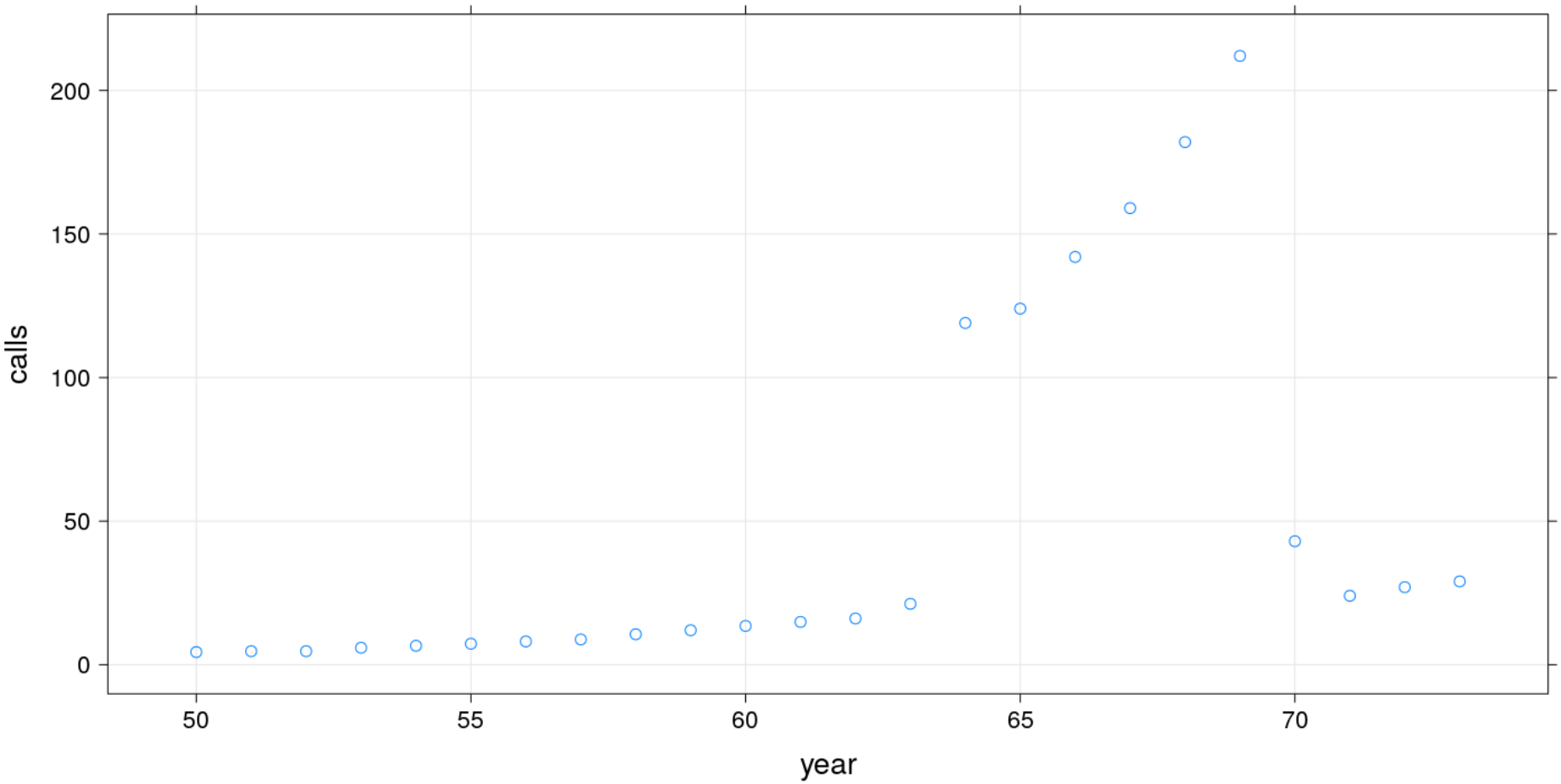
```
opt$par  # least absolute errors
```

```
[1] -2.832090898  0.008471257
```

- The two lines are virtually identical in this case
- This is not always true

Another example: number of phone calls per year in Belgium

```
data(phones, package = "MASS")
xyplot(calls ~ year, data = phones, grid = TRUE)
```



Another example: number of phone calls per year in Belgium

```
fm2 <- lm(calls ~ year, data = phones)
SAE = function(beta)
{
    with(phones,
         sum(abs(calls - beta[1] - beta[2] * year)))
}
opt = optim(c(0, 0), fn = SAE)
```

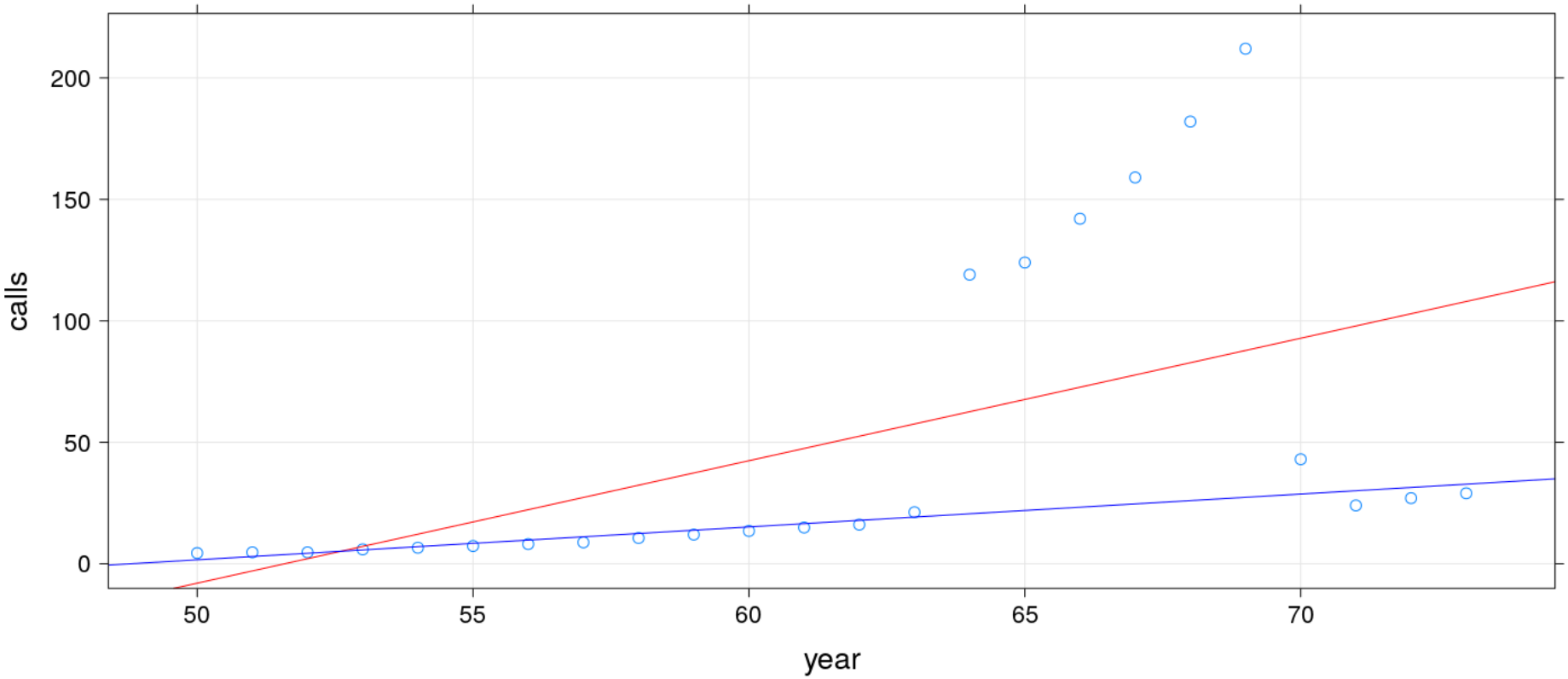
Another example: number of phone calls per year in Belgium

```
fm2 <- lm(calls ~ year, data = phones)
SAE = function(beta)
{
    with(phones,
         sum(abs(calls - beta[1] - beta[2] * year)))
}
opt = optim(c(0, 0), fn = SAE)

coef(fm2) # least squared errors

(Intercept)        year
-260.059246    5.041478

opt$par  # least absolute errors

[1] -66.053297   1.353735
```

- The two lines are quite different
- The second line is an example of *robust regression*

Another example: number of phone calls per year in Belgium

```
xyplot(calls ~ year, data = phones, grid = TRUE,
       panel = function(x, y, ...) {
           panel.xyplot(x, y, ...)
           panel.abline(fm2, col = "red") # least squared errors
           panel.abline(opt$par, col = "blue") # least absolute errors
       })
```



Summary

- Conventional statistical learning focuses on problems that can be "solved" analytically

Summary

- Conventional statistical learning focuses on problems that can be "solved" analytically
- Numerical solutions are also valid solutions... but potentially difficult to obtain
- R makes it *easy* to obtain numerical solutions and compare with traditional solutions
- We will come back to this idea when we next discuss the origins of R

A very brief history of R

What is R?

From its own website:

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

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

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

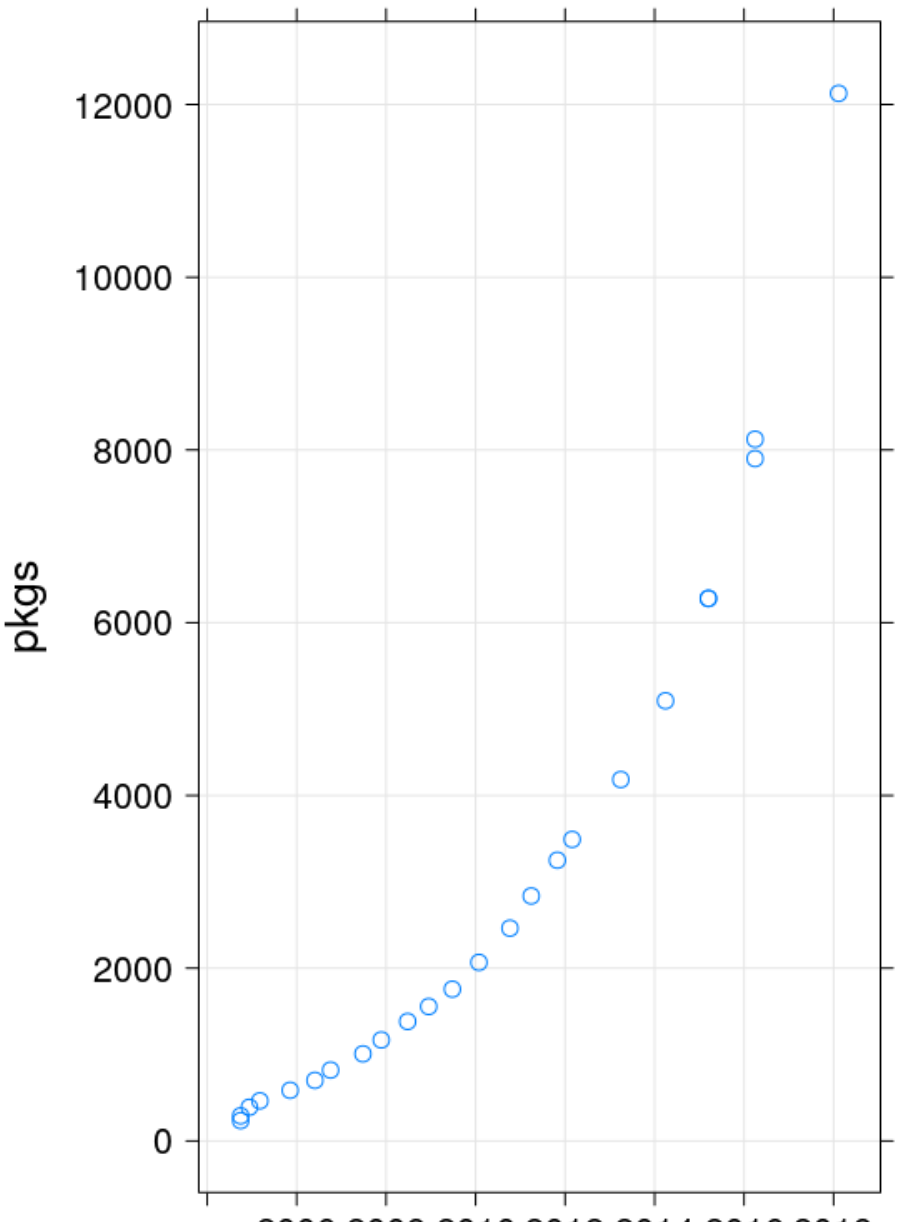
> *For The S system, which has forever altered how people analyze, visualize, and manipulate data.*

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.*

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`:

```
#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:

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

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
```

Rapid prototyping: flexibility and extensibility

- Powerful built-in tools
- Programming language
- Compiled code for efficiency

Another strength: Interfaces

- Not all useful software developed by R community
- Core open source philosophy: code re-use
- Creating interfaces with external software is relatively easy
- Example: Keras / TensorFlow

Keras

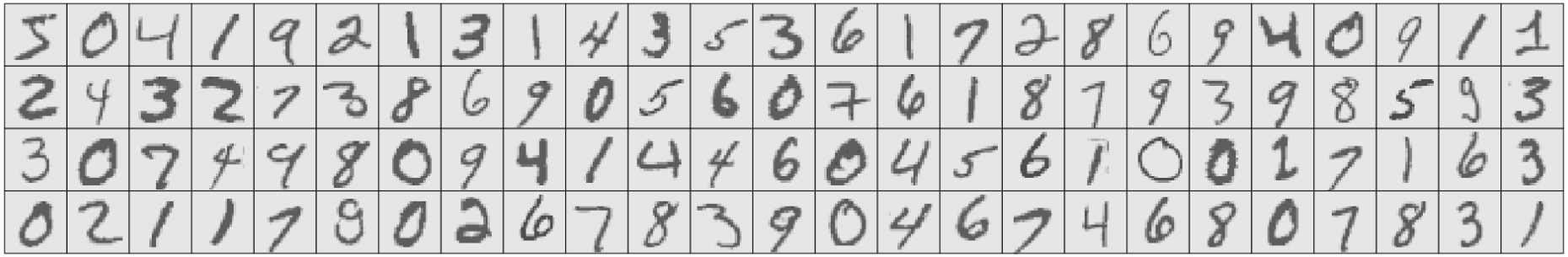
- Deep learning framework based on TensorFlow
- R interface through package keras

Example: classify handwritten digits

```
library(keras)
mnist <- dataset_mnist()
x_train <- mnist$train$x # each sample is a 28x28 grayscale image
y_train <- mnist$train$y # correct classification (0,1,2,...,9)
x_test <- mnist$test$x
y_test <- mnist$test$y
```

Example: classify handwritten digits

```
library(keras)
mnist <- dataset_mnist()
x_train <- mnist$train$x # each sample is a 28x28 grayscale image
y_train <- mnist$train$y # correct classification (0,1,2,...,9)
x_test <- mnist$test$x
y_test <- mnist$test$y

xtrain.100 <- as.data.frame.table(x_train[1:100,,])
levelplot(Freq ~ Var3 + Var2 | Var1, data = xtrain.100, strip = FALSE, scal
          ylim = c(28, 1), colorkey = FALSE, col.regions = rev(grey.colors(
```



Transform data

- Reshape data (to vector) and rescale

```
# reshape each 28x28 image matrix to 784-vector
dim(x_train) <- c(nrow(x_train), 784)
dim(x_test) <- c(nrow(x_test), 784)

# rescale grayscale values (0-225) to (0,1)
x_train <- x_train / 255
x_test <- x_test / 255

y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)
```

Define model

- A Keras model is a way to organize layers
- Define a sequential model (a linear stack of layers)

```
model <- keras_model_sequential()
layer_dense(model, units = 256, activation = "relu", input_shape = c(784))
layer_dropout(model, rate = 0.4)
layer_dense(model, units = 128, activation = "relu")
layer_dropout(model, rate = 0.3)
layer_dense(model, units = 10, activation = "softmax")
summary(model)
```

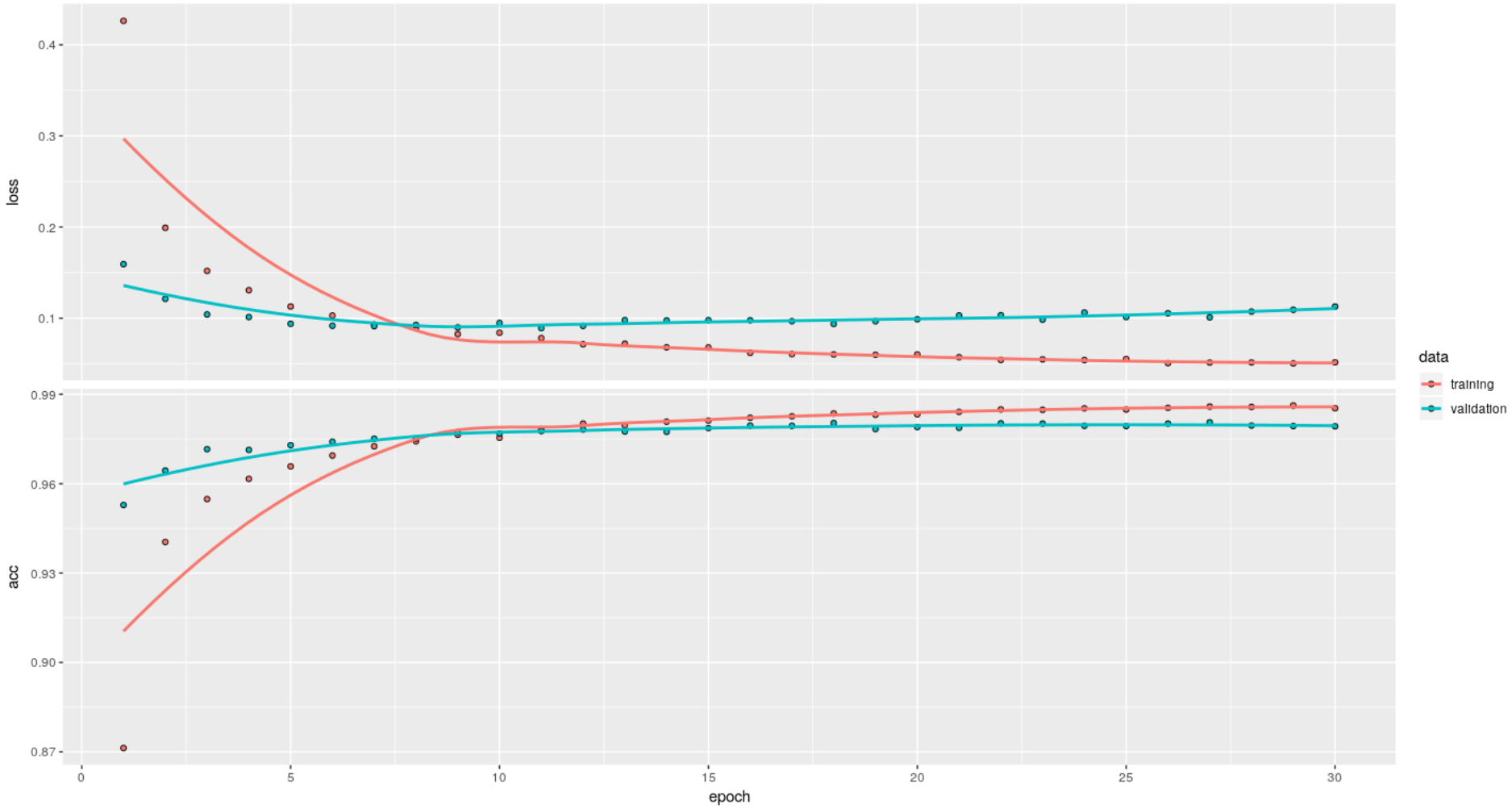
```
_________________________________________________________________
Layer (type)                 Output Shape
=================================================================
dense_1 (Dense)              (None, 256)
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)
_________________________________________________________________
dense_2 (Dense)              (None, 128)
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)
_________________________________________________________________
dense_3 (Dense)              (None, 10)
```

Compile and train model

```
compile(model,
        loss = "categorical_crossentropy",
        optimizer = optimizer_rmsprop(),
        metrics = c("accuracy"))
history <- fit(model,
               x_train, y_train,
               epochs = 30, batch_size = 128,
               validation_split = 0.2)
```

Evaluate model

```
p <- plot(history)
p
```



Results on test data

```
pred_class <- predict_classes(model, x_test)
pred_class[1:20]
```

```
 [1] 7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4
```

```
y_test[1:20,]
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    0    0    0    1    0     0
 [2,]    0    0    1    0    0    0    0    0    0     0
 [3,]    0    1    0    0    0    0    0    0    0     0
 [4,]    1    0    0    0    0    0    0    0    0     0
 [5,]    0    0    0    0    1    0    0    0    0     0
 [6,]    0    1    0    0    0    0    0    0    0     0
 [7,]    0    0    0    0    1    0    0    0    0     0
 [8,]    0    0    0    0    0    0    0    0    0     1
 [9,]    0    0    0    0    0    1    0    0    0     0
[10,]    0    0    0    0    0    0    0    0    0     1
[11,]    1    0    0    0    0    0    0    0    0     0
[12,]    0    0    0    0    0    0    1    0    0     0
[13,]    0    0    0    0    0    0    0    0    0     1
[14,]    1    0    0    0    0    0    0    0    0     0
[15,]    0    1    0    0    0    0    0    0    0     0
```

Misclassification rate in test data

```
ctab <- table(pred_class, apply(y_test, 1, which.max)-1)
ctab
```

```
pred_class    0    1    2    3    4    5    6    7    8    9
         0  971    0    2    0    0    2    4    3    4    5
         1    1 1126    2    0    1    0    3    3    3    2
         2    2    3 1020    4    4    0    0    8    3    1
         3    0    0    0  987    0    2    1    1    5    5
         4    0    0    1    0  957    0    3    0    1    9
         5    2    1    0    9    0  877    3    0    5    4
         6    2    2    0    0    5    5  943    0    1    0
         7    1    0    4    6    2    1    0 1009    3    4
         8    1    3    3    2    1    4    1    1  947    2
         9    0    0    0    2   12    1    0    3    2  977
```

```
sum(diag(ctab)) / sum(ctab)
```

```
[1] 0.9814
```

Another interface: plotly

- Plotly: a Javascript library for visualization
- R interface provided by the `plotly` R package

```
library(plotly)
ggplotly(p)
```

More HTML-based applications

- DataTable (plug-in for jQuery) - example earlier
- HTML widgets
- Shiny Apps

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