Session 2

Antoine Vernet

Lesson plan

Today we will cover:

- Programming (more on this in the workshops)
  - Getting started
  - Finding help
- Calculus
- Fundamentals of probability
  - Random variables
  - Probability distribution
  - Independence

We will finish the class with some time for Q & A

R programming

Short version

The Hitchhiker's Guide to the Galaxy (trailer 1)

R is cool!

Why?

- R is a complete programming language
  - you can do statistics and more! (e.g. web scraping)
- the community is huge and growing!
  - lots of packages
  - lots of code examples and tutorials
  - lots of places to find help
- many companies are seeking skilled R analysts!

Examples of things made with R

- Scientific papers
- Websites
- Web apps: example 1, example 2
- these slides

RStudio

The standard R gui is underwhelming. Rstudio offers lots of shortcuts and helpful integration:

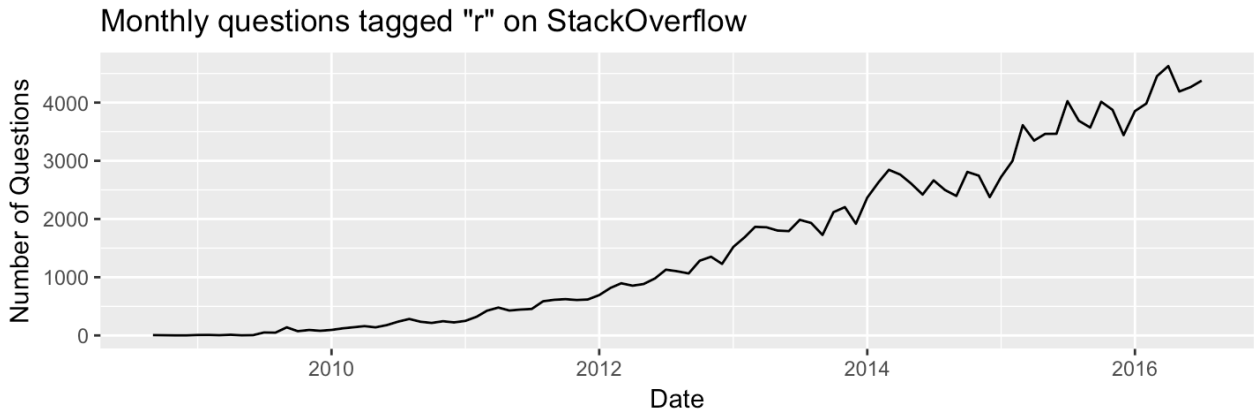
- code completion
- git integration
- debugging
- package development tools
- authoring (Rmarkdown)

You can learn more about RStudio (and R) by reading the documentation on the RStudio website and by watching their webinars

Getting help with R

There are many places to get help with R. The main ones are:

- Rseek, for a version of google that understands that R is not a letter.
- Stack Overflow, for programming questions
- Cross Validated, for statistics questions
- R mailing lists, for everything else



Other places to find information and help

- R-bloggers, for the freshest news about the R community
- Coursera, particularly the classes by the team from John Hopkins University
- Datacamp, remember you have free access to Datacamp for the duration of the course

Version control

a very short introduction (more in the workshop)

Git

Git is a version control software started by Linus Torvalds (of Linux's fame).

It allows you to track changes in your file on a specific project.

Useful when writing code:

- when things break, you can rewind to figure it out
- it makes collaboration easier
- it forces you to describe what you do (through commit messages)
- one day it will save you a lot of tears and pain

Basics of Git

Git is integrated with RStudio:

- no need to use the shell
- but knowing how to use the shell is useful when things break
- you need to know a few functions:
    - git init
    - git add
    - git commit
    - git checkout

Learn more: RStudio git integration, an intro to git and RStudio

Bitbucket and Github

- Two platforms to host your code
- Gitbub: great for public code
- Bitbucket: great for non-public code (on an academic licence)
- Both use git and:
  - store your code remotely
  - allow you to collaborate with yourself and others

R Markdown

R Markdown is a flavour of Markdown

It allows to write literate programming document that can be rendered in different format (e.g. HTML, PDF, Word)

We will use R Markdown documents heavily in this class (Usually, I will give you a template).

Because it is a text file, it can be version controlled.

A note about formats

Schematically, there are two large families of formats:

- proprietary
- open

Open format are usually longer lived than proprietary format and because their specifications are public one could always write a new parser if needed.

Text formats

Textual formats (.txt, .csv, .md, .Rmd, .html) are open formats. They have two main advantages:

- they are portable
- they can be easily version controlled

They present some disadvantages:

- not always compact
- might require extra code to load/write

```
#for example, for a csv file:
data <- read.csv("./path/to/file.csv", stringsAsFactors = FALSE)

#for RData format
load("./path/to/file.RData")
```

Coding style

Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. Code for readability.

John F Woods

Coding style

Why do you need a coding style?

- Code is also read by humans
- Hard to read code is less likely to be used
- You have to read your own code, save yourself some time by:
    - Following a style guide
    - Commenting your code heavily

There are plenty of coding style guides around. I find the one in Hadley Wickham's book *Advanced R* simple and effective.

Calculus

Function

A function is a relation between a set of inputs and a set of permissible outputs, each input is related to exactly one output.

The notation is:

$$f(x) = y$$

where f is the function, x is the domain, y is the image and (x, y) is the graph of the function. Usually, we say that f maps x to y.

Linear functions

A linear function is a map between 2 vector spaces. It preserves vector addition and scalar multiplication. Most (all) of the functions we will be encountering in this class are linear functions.

They are very useful to study linear processes but also as approximations for non-linear processes (e.g. logistic regression).

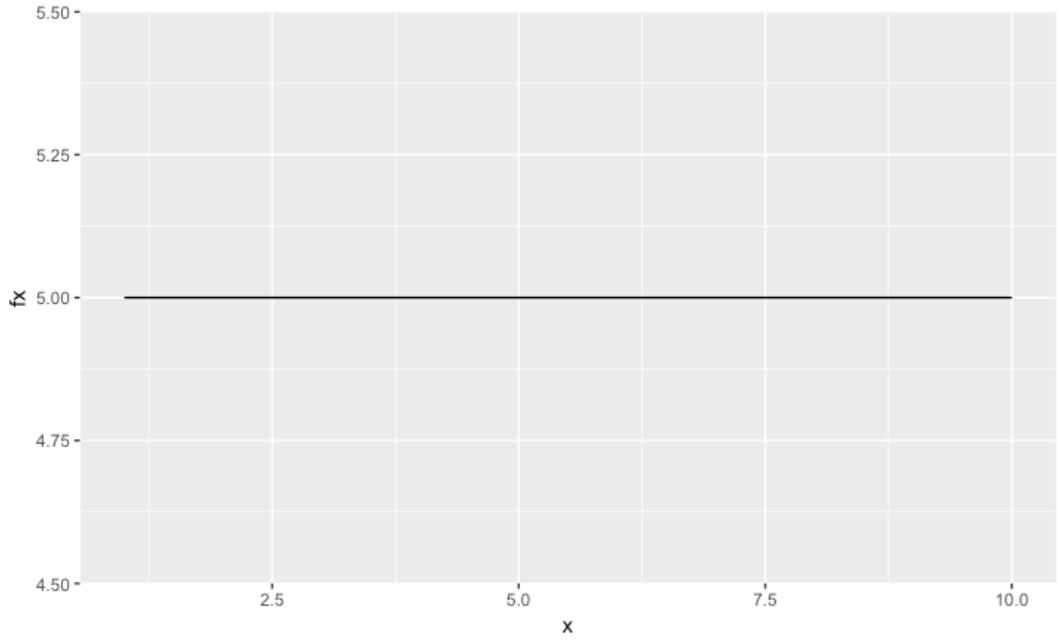
Some example of functions

The constant function takes the form: $f(x) = a$

A simple linear function: $f(x) = ax + b$

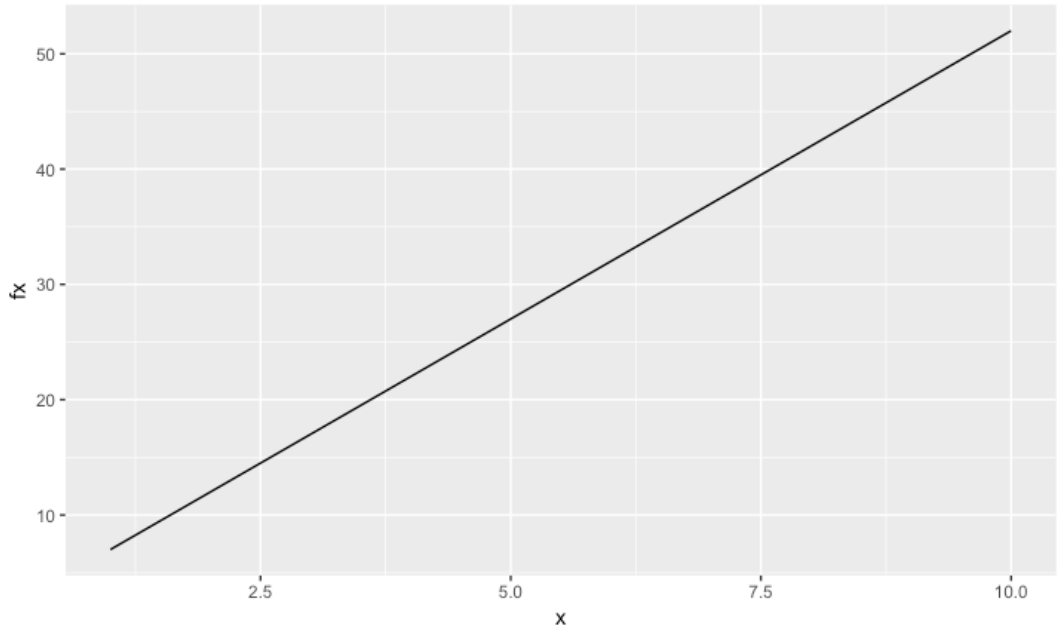
$$f(x) = a$$

```
library(ggplot2)
data <- data.frame(x = 1:10, fx = rep(5, 10))
ggplot(data, aes(x = x, y = fx)) + geom_line()
```



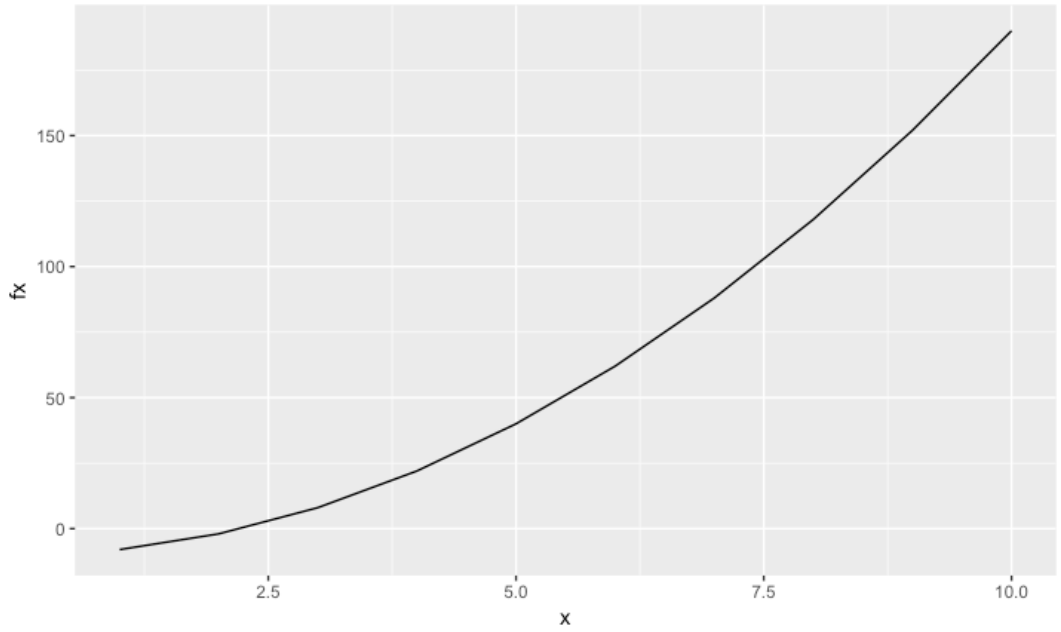
$$f(x) = ax + b$$

```
x <- 1:10
fx <-  5 * x + 2
data <- data.frame(x = x, fx = fx)
ggplot(data, aes(x = x, y = fx)) + geom_line()
```



$$f(x) = ax^2 + b$$

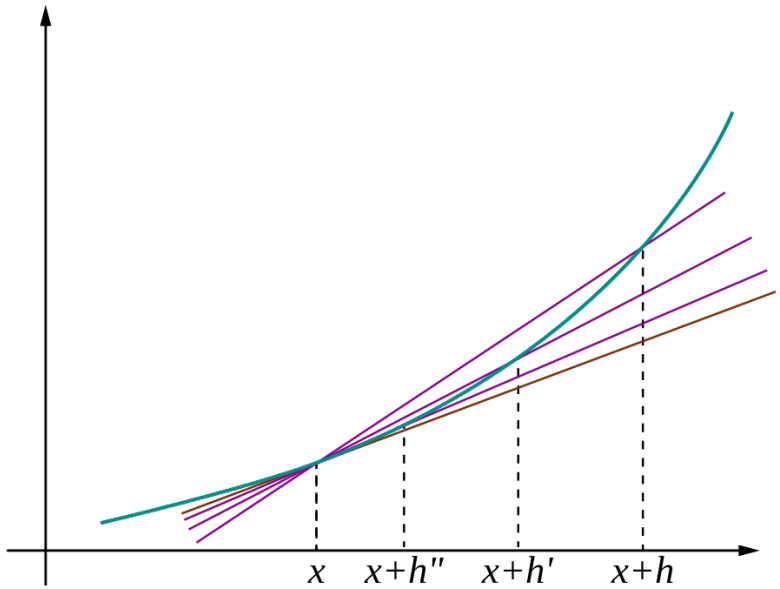
```
x <- 1:10
fx <-  2 * x ^ 2 - 10
data <- data.frame(x = x, fx = fx)
ggplot(data, aes(x = x, y = fx)) + geom_line()
```



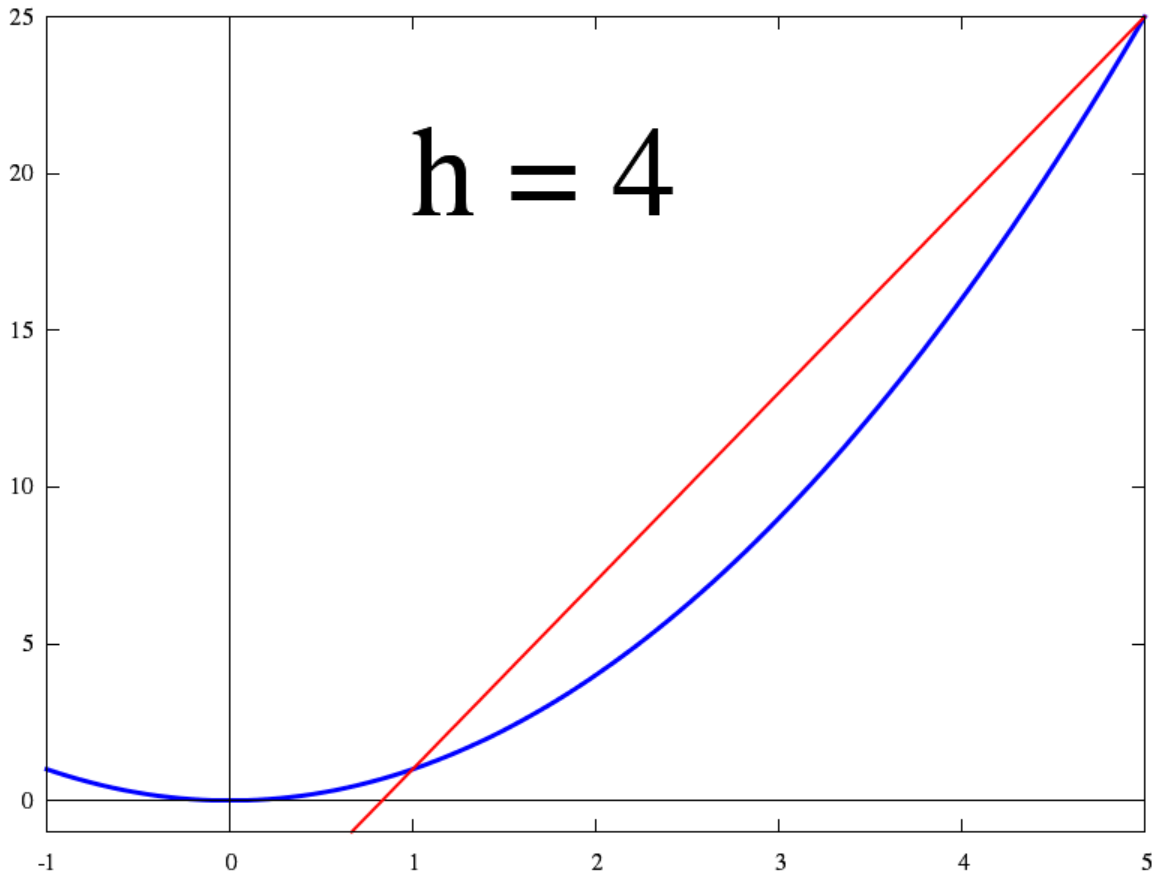
Derivatives

The first derivative of a univariate function $f : \mathbb{R} \mapsto \mathbb{R}$ is the function $f' : \mathbb{R} \to \mathbb{R}$ defined as:

$$f'(x) = \lim_{h \downarrow 0} \frac{f(x+h) - f(x)}{h}$$



Illustration



Interpretation

The first derivative tells you about:

- the rate of change
- the slope of the tangent at x
- describes the best **linear approximation** of f near x
  - $f'(x) > 0$ (< 0) then f is increasing (decreasing) around x
  - $f'(x) = 0$ then $f(x)$ is a critical point and x is a critical value (the behaviour of f is unclear)

Second derivative

The second derivative of f, denoted f'' is the first derivative of f' ($f'' = (f')'$). The second derivative tells us about:

- describes the change in the rate of change of f.
- describes the curvature of f near x (best **quadratic approximation**):
  - $f''(x) > 0$ (< 0) then f is curving upwards (downwards) around x
  - $f''(x) = 0$ then the behaviour of f is unclear
  - what if $f'(x^*) = 0$ and $f''(x^*) > 0$ (or < 0)?

Interpretation of derivatives in physics

If $f(x)$ describes the position of x:

- $f'(x)$ describes the **velocity** of x
- $f''(x)$ describes the **acceleration** of x
- Useful for pub quizzes:
  - $f'''(x)$ describes the **jerk** of x
  - $f^{iv}(x)$ describes the **jounce** of x

Minimum and maximum

When $f'(x) = 0$, $f(x)$ is a critical point:

- if $f''(x) > 0$ then $f(x)$ is a local minimum of f
- if $f''(x) < 0$ then $f(x)$ is a local maximum of f

Derivatives in R

```
D(expression(cos(x)), "x")
```

```
## -sin(x)
```

```
D(expression(x^2), "x")
```

```
## 2 * x
```

```
D(D(expression(x^3), "x"), "x")
```

```
## 3 * (2 * x)
```

Derivatives in R (2)

The function `deriv` helps construct a function that will return both $f(x)$ and $f'(x)$:

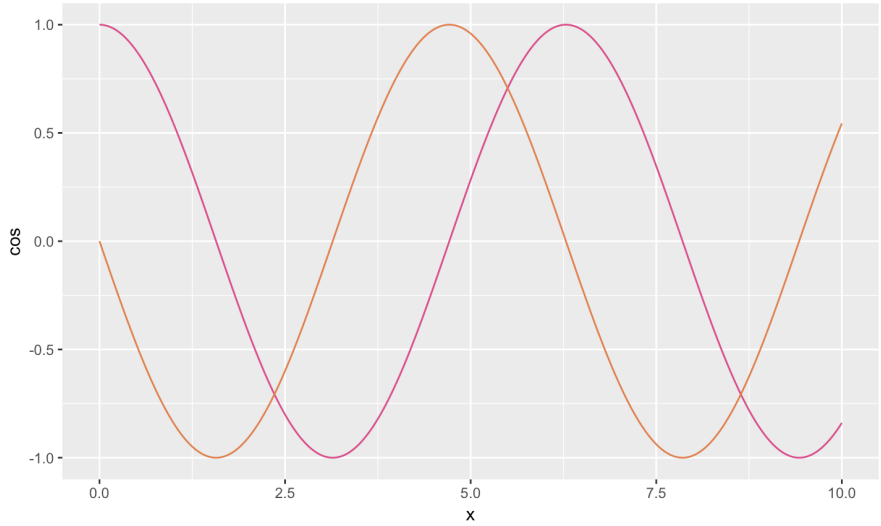
```
cos_deriv <- deriv(expression(cos(x)), "x", function.arg = TRUE)
cos_deriv(1:5)
```

```
## [1]  0.5403023 -0.4161468 -0.9899925 -0.6536436  0.2836622
## attr(,"gradient")
##               x
## [1,] -0.8414710
## [2,] -0.9092974
## [3,] -0.1411200
## [4,]  0.7568025
## [5,]  0.9589243
```

Derivatives in R (3)

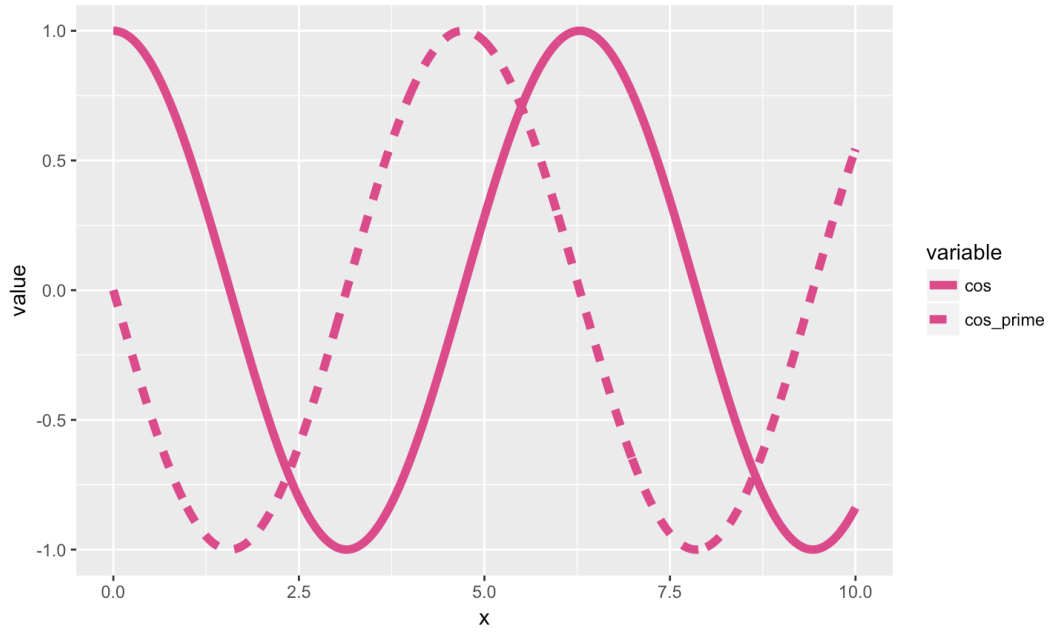
```
cd <- data.frame(x = seq(0, 10, by = 0.01),
                 cos = as.numeric(cos_deriv(seq(0, 10, by = 0.01))),
                 cos_prime = as.numeric(attr(cos_deriv(
                   seq(0, 10, by = 0.01)), "gradient")))

ggplot(data = cd, aes(x = x)) +
       geom_line(aes(y = cos), color="#FF3489") +
  geom_line(aes(y = cos_prime), color="#FF7645")
```



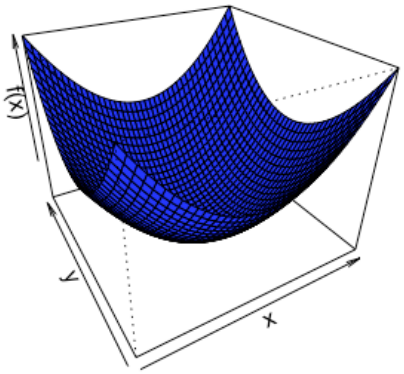
Derivatives in R (4): a better graph

```
library(reshape2)
cd2 <- melt(cd, id.vars = "x")
ggplot(data = cd2, aes(x = x, y = value, linetype = variable)) +
       geom_line(size = 2, colour="#FF1789")
```



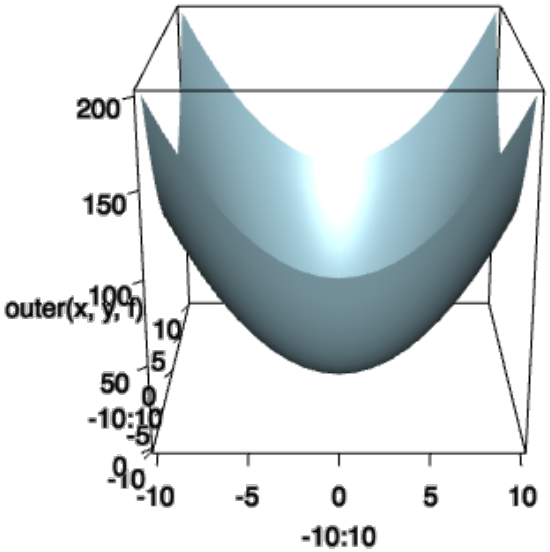
Multivariate functions

```
x <- -10:10; y <- -10:10
x_seq <- seq(min(x), max(x), length.out = 40)
y_seq <- seq(min(y), max(y), length.out = 40)
fx <- outer(x_seq, y_seq, function(x, y){x ^ 2 + y ^ 2})
persp(x = x_seq, y = y_seq, z = fx, theta = -30, phi = 30,
      xlab = "x", ylab = "y", zlab = "f(x)", col = "blue", expand = 0.8)
```



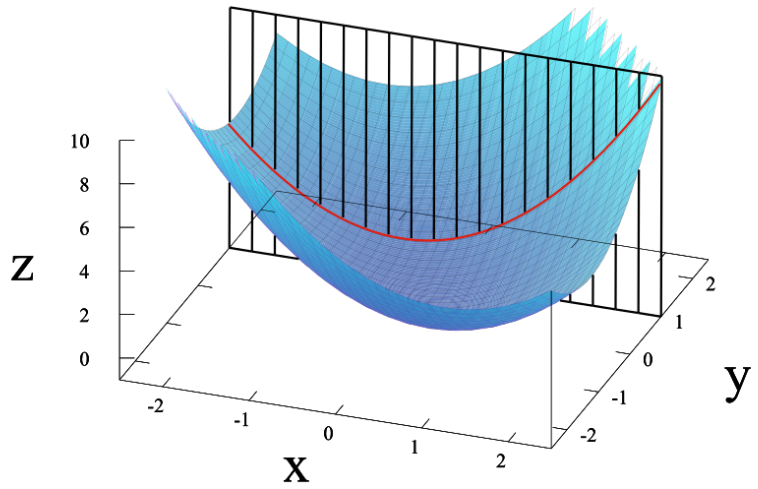
RGL

```
#library(rgl)
f <- function(x, y){x ^ 2 + y ^ 2}
persp3d(x = -10:10, y = -10:10, z = outer(x, y, f), color = 'lightblue')
```



Partial derivative

For multivariate function, such as $f(x, y)$, a partial derivative is the derivative of this function with respect to one variable, with others held constant. It is usually noted $f'_x(x, y)$ (another common notation is $\frac{\partial f}{\partial x}$).



Properties of partial derivative

- One can define the second partial derivative of f, noted $\frac{\partial^2}{\partial x_i \partial x_j} f$
- Schwarz' theorem: $\frac{\partial}{\partial x_i \partial x_j} f = \frac{\partial}{\partial x_j \partial x_i} f$ if the second derivatives are continuous

Gradient

The gradient of a multivariate function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is noted ∇f and defined as the vector of partial derivatives of f:

$$\nabla f(a) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(a) \\ \vdots \\ \frac{\partial}{\partial x_n} f(a) \end{bmatrix}$$

∇f points in the direction of the greatest rate of increase of f

The magnitude of ∇f is the slope of the graph in that direction

Hessian

The Hessian (or Hessian matrix) is a square matrix of the partial second derivatives of a function.

$$H(a) = \begin{bmatrix} \frac{\partial^2}{\partial^2 x_1} f(a) & \frac{\partial^2}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n} \\ \frac{\partial^2}{\partial x_2 \partial x_1} f(a) & \frac{\partial^2}{\partial^2 x_2} f(a) & \cdots & \frac{\partial^2}{\partial x_2 \partial x_n} f(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} f(a) & \frac{\partial^2}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2}{\partial^2 x_n} \end{bmatrix}$$

If all second derivates are continuous, the Hessian is symmetric (Schwartz' theorem)

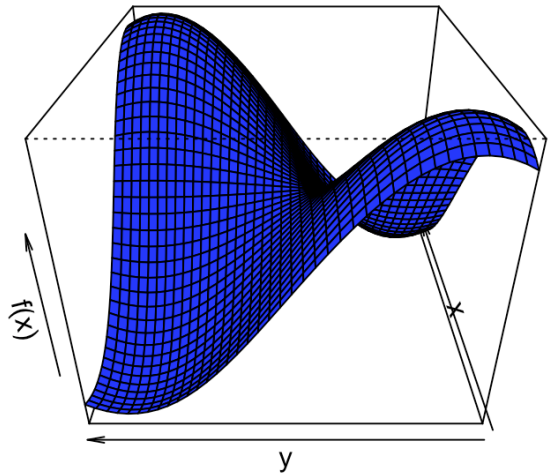
Hessian, maximum and minimum

Studying the properties of the Hessian of a function f in a specific point x will help determine some properties of the function on that point:

- If the Hessian is positive definite, then f attains a local minimum in x
- If the Hessian is negative definite, then f attains a local maximum in x
- If the Hessian has both positive and negative eigenvalues, x is a saddle point for f

Hessian

```
library(pracma); library(matrixcalc)
f <- function(x, y) sin(x) * sin(y); x <- y <- seq(-2, 2, length.out = 40)
persp(x = x, y = y, z = outer(x, y, f), theta = -90, phi = 30,
      xlab = "x", ylab = "y", zlab = "f(x)", col = "blue", expand = 0.8)
```



Hessian (2)

```
library(pracma)
library(matrixcalc)
f <- function(x) sin(x[1]) * sin(x[2])
x0 <- c(0, 0)
hessian(f, x0)
```

```
##      [,1] [,2]
## [1,]    0    1
## [2,]    1    0
```

Hessian (3)

```
is.positive.definite(hessian(f, x0))
```

```
## [1] FALSE
```

```
is.negative.definite(hessian(f, x0))
```

```
## [1] FALSE
```

```
eigen(hessian(f, x0))
```

```
## $values
## [1]  1 -1
##
## $vectors
##           [,1]       [,2]
## [1,] 0.7071068 -0.7071068
## [2,] 0.7071068  0.7071068
```

Hessian (4)

```
f <- function(u) {
    x <- u[1]; y <- u[2]; z <- u[3]
    return(x^3 + y^2 + z^2 + 12*x*y + 2*z)
}
x0 <- c(1, 1, 1)
hessian(f, x0)
```

```
##      [,1] [,2] [,3]
## [1,]    6   12    0
## [2,]   12    2    0
## [3,]    0    0    2
```

```
is.positive.definite(hessian(f, x0))
```

```
## [1] FALSE
```

```
is.negative.definite(hessian(f, x0))
```

```
## [1] FALSE
```

```
eigen(hessian(f, x0))
```

```
## $values
## [1] 16.165525  2.000000 -8.165525
##
## $vectors
##           [,1] [,2]       [,3]
## [1,] 0.7630200    0  0.6463749
## [2,] 0.6463749    0 -0.7630200
## [3,] 0.0000000    1  0.0000000
```


That's all Folks!