Advanced Econometric Methods EMET3011/8014

Lecture 1

John Stachurski

Semester 1, 2011

Today

Organization

Course content

Comments on the R language

Background material

- Algorithms
- Math preliminaries

Course Homepage

The course homepage is

`http://johnstachurski.net/emet`

All course material will be posted here

- course outline
- course notes
- lecture material
- assignments
- news

Please check this page regularly

Wattle

How about Wattle?

`http://wattlecourses.anu.edu.au/`

Not active—has links to the course homepage

Instructor

Name: John Stachurski

Office: Room 1023, HW Arndt Building 25a

Email: `john.stachurski@anu.edu.au`

Contact hours: Monday 8:30–10:00 AM.

Tutor

Name: Varang Wiriyawit

Office: Room 1025, HW Arndt Building 25a

Contact hours: Wednesday 2:00–3:30 PM

Timetable

- Lecture: Thursday 9–11 AM, Arndt LT2

No audio recordings, but lecture slides online.

- Comp Lab 1: Thur 12–1 PM, COP G021 (Bld. 24)
- Comp Lab 2: Thur 1–2 PM, COP G021 (Bld. 24)

Students should attend Comp Lab 1 or 2, but **not both**

- No computer lab today (first one March 3th)

No tutes on theory—make use of office hours

Assessment

Two assignments, a mid-term and final exam

Weighting:

Assignment 1:	15%
Assignment 2:	15%
Mid-term Exam:	25%
Final Exam:	45%

Scaling: final grade not an exact sum of these scores

Separate scaling for EMET3011 and EMET8014

Reading Material

No course text

Primary source is a PDF file of course notes (see course web page)

Lecture material taken directly from these notes

Please don't print the whole file

- Still undergoing revision
- Suitable for viewing on line

Supplementary stats/econometrics reading:

- Hayashi, F. (2000): *Econometrics*.
- Davidson R. and J. MacKinnon (2004): *Econometric Theory and Methods*.
- Amemiya, T. (1994): *Introdution to Statistics and Econometrics*.
- Casella, G. and L. Berger (1990): *Statistical Inference*.
- Greene, W. H. (1990): *Econometric Analysis*.

Prerequisites

Nominally, prereqs = EMET2008 or equivalent

If you don't have them and want to take course come and see me

- You should know some calculus
- Preferably some knowledge of matrix algebra
- Basic familiarity with regression
- General aptitude for mathematical/logical arguments

Programming experience is not required

General comments

Please refer **administrative** questions to course administrator

- Terry Embling, room 1013, HW Arndt Bulding (25a)

Please do not use email for instructional purposes

- Make use of tutor's/my office hours

Course Description

An advanced course in econometric theory

- Emphasis on general concepts
- Relatively rigorous, mathematical

You will learn to

- Understand key methods of inference
- Derive theoretical results from first principles

Necessary mathematics will receive careful treatment

Will also introduce you to the stats programming environment R

More programming oriented than Eviews/STATA/S-Plus/etc

Working with R a requirement of the course

You will learn to

- Interact with R's statistical/graphical functions
- Complete basic programming tasks

Why all the theory/programming?

Hal Varian, chief economist at Google:

> *I keep saying the sexy job in the next ten years will be statisticians.*

Why sexy? Because computers are revolutionizing the field

Lots of new data

Lots of new methods

You need good theory to keep abreast

And good programming skills to manipulate the data, implement new methods

Course Topics

1. Introduction to R
2. Probability
3. Fundamentals of Inference
4. Linear Algebra
5. Regression
6. Time Series Analysis

Introduction to R

- Getting started with the R interpreter
- Variables
- Vector operations and linear algebra
- Data import/export
- Graphics
- Common statistical functions
- Programming

Probability

Econometric theory is all about statistics

In turn, statistics is built on probability theory

Hence, we need to know the basics:

- Distributions (cdfs, pmfs, pdfs) and expectations
- Conditioning and independence
- Asymptotic results (convergence concepts, LLN, CLT)

Fundamentals of Inference

How do we learn from data?

- What is inference?
- Estimators
- Principles of estimation (Max likelihood, ERM)
- Confidence intervals and hypothesis tests
- Nonparametrics
- Empirical distributions

Linear Algebra

Multiple regression is all about vectors and matrices

Hence, we need to learn about

- Vectors
- Systems of equations
- Span, rank, linear independence
- Working with matrices
- Orthogonal projections

Regression

- The OLS model
- Solving for the OLS estimator
- Key properties of the OLS estimator
- The Gauss-Markov theorem
- Statistical properties under normality
- Misspecification, Goodness of fit
- Large sample properties

Time Series Analysis

- Time series models
- Stationarity, ergodicity, martingales, etc.
- Maximum likelihood for time series

The R Language

What is R?

- A language and environment for statistical computing
- An open source implementation of S
- Provides wide variety of statistical techniques

Cons of R:

- Steep learning curve relative to Eviews/STATA/etc.
- Less elegant than Python, Ruby, etc.

Pros of R:

- Contains complete programming language
  - Can tackle sophisticated problems
  - But not too hard to learn
- Huge range of statistical functions/tests
  - See `http://cran.r-project.org/`
- Good graphics and visual presentation
  - Sensible defaults, good user control
  - Publication quality output
- Free

How to get it?

Can be obtained from the R homepage

- `http://www.r-project.org/` (click on CRAN)

Or go straight to local mirror

- `http://cran.ms.unimelb.edu.au/`

(Install `base`)

See the course home page for more instructions

Examples of R in action

Read in CSV data on mortgage rates and 10 year treasury yields

```
mr <- read.csv("H15_MORTG_NA.txt")
ty <- read.csv("H15_TCMNOM_Y10.txt")
```

Regress mortgage rates on treasury yields and print summary:

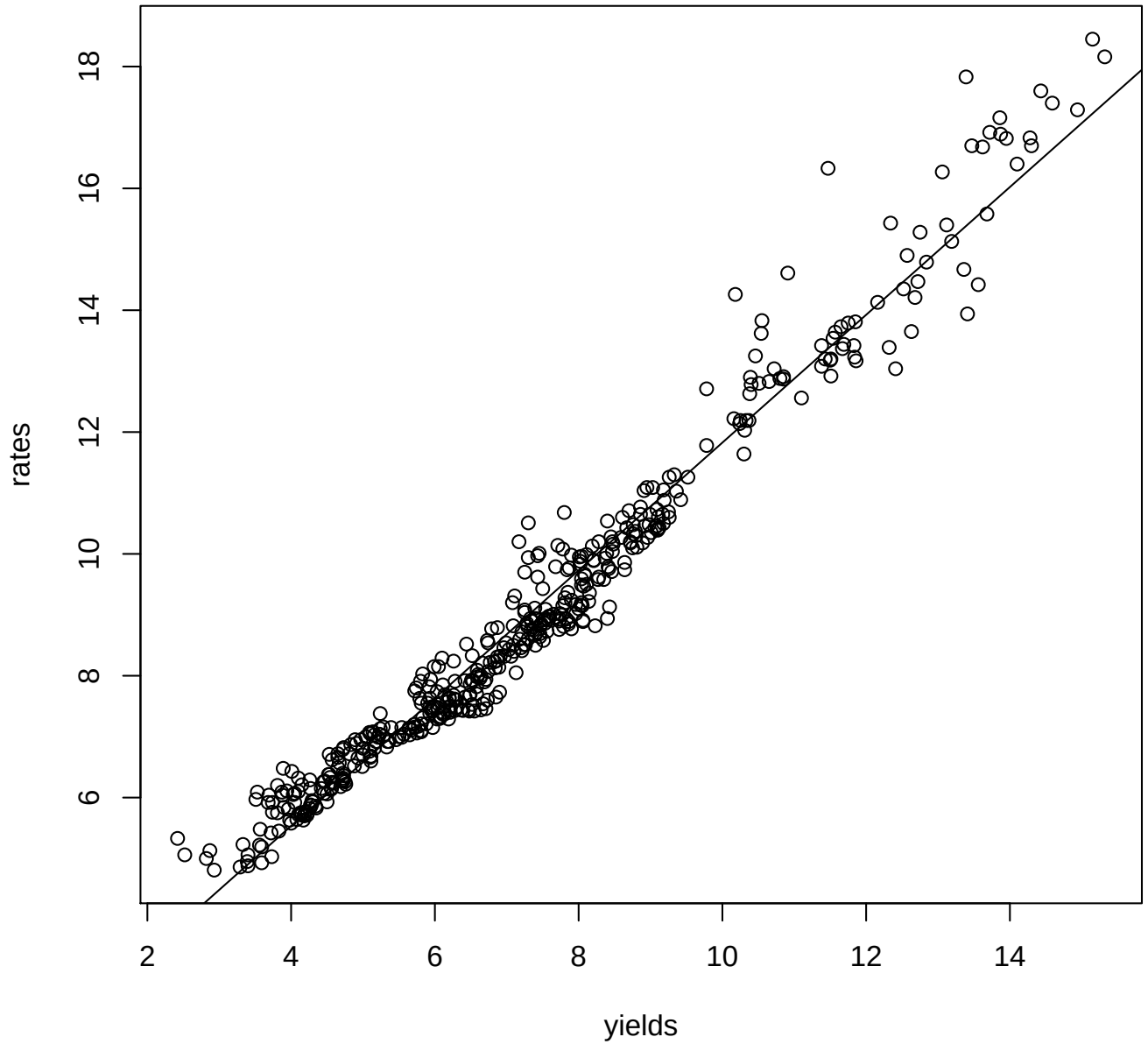
```
results <- lm(mr$MORTGNA ~ ty$TCMNOMY10)
summary(results)
```

Produces output as follows

```
                Estimate Std. Error t value Pr
                   (>|t|)
(Intercept)    1.338709   0.071350   18.76
   <2e-16 ***
ty$TCMNOMY10 1.048804   0.009188  114.14
   <2e-16 ***
```

Visual presentation of results:

```
plot(ty$TCMNOMY10, mr$MORTGNA)
abline(results)
```


rates
yields
2
4
6
8
10
12
14
6
8
10
12
14
16
18

Programming:

```
num.repetitions <- 10000
outcomes <- numeric(num.repetitions)
for (i in 1:num.repetitions) {
    num.tails <- 0
    num.heads <- 0
    while (num.heads < 10) {
        b <- runif(1)
        num.heads <- num.heads + (b < 0.4)
        num.tails <- num.tails + (b >= 0.4)
    }
    outcomes[i] <- num.tails
}
print(mean(outcomes))
```

Web searches

Not the most google-friendly name

Several people maintain R-specific search engines:

- R search: `http://www.r-project.org/search.html`
- R seek: `http://www.rseek.org`

These links on the course homepage

Next week we start learning R

Finish today's lecture by reviewing basic maths, algorithms

Algorithms

A sequence of instructions for completing a given task

Simple example: A day at work

```
1 get up;
2 go to work;
3 close office door, surf Internet;
4 go home;
5 go to bed;
```

Actions executed sequentially: one line after another.

Algorithms are implemented using programming languages.

Benefit of programming languages: **computer** can understand

Cost of programming languages: hard for **humans** to read:

```
my $cmd;
if ($args{prog} !~ /^\//) {
  $cmd = ``$ENV{ZEUSHOME}/$args{prog} $opt'';
} else {
  $cmd = ``$args{prog} $opt'';
}
```

Hence pseudocode: **algorithms for humans**

Algorithms in Pseudocode

Variables:

1 $x \leftarrow 1$;
2 print x;
3 $x \leftarrow 2$;
4 print x;
5 $x \leftarrow x + x$;
6 print x;

(The "=" symbol often used instead of $\leftarrow$)

Note: "print" means send output to screen/user

Flow control:

- **if–then–else**
- **while**
- **for**

Determine when/if statements will be executed

Syntax of **if–then–else** construct is

```
1 if condition then
2 |   first sequence of actions;
3 else
4 |   second sequence of actions;
5 end
```

```
1 if there are cookies in the jar then
2 |   eat them;
3 else
4 |   go to the shops;
5 |   buy more cookies;
6 |   eat them;
7 end
```

The **while** construct creates a loop with test condition

```
1 while condition do
2 |   perform sequence of actions;
3 end
```

The flow is like this:

```
1 if condition then
2 |   perform sequence of actions;
3 |   go to line 1;
4 else
5 |   terminate loop;
6 end
```

Example:

```
1 while there are cookies in the jar do
2 |   eat one;
3 |   watch TV for 10 minutes;
4 end
```

The **for** construct:

```
1 for index in list do
2 |   sequence of actions;
3 end
```

Example:

```
1 for cookie in cookie jar do
2 |   eat cookie;
3 |   watch TV for 10 minutes;
4 end
```

A better example:

Let $S \leftarrow (1, 2, 3)$ and consider

```
1 for x in S do
2 |   print x^2;
3 end
```

Value of x steps through S from start to finish

The different constructs can be combined...

Simulate RV counting number of heads in 10 flips of biased coin

Heads occurs with prob $p \in (0,1)$

---

1 $H \leftarrow 0$;
2 **for** *i in 1 to 10* **do**
3 | draw U from the uniform distribution on $[0,1]$;
4 | **if** $U \leq p$ **then**
5 | | $H \leftarrow H + 1$;
6 | **end**
7 **end**
8 print H;

Note: **Indentation is important in maintaining readability**

What does this algorithm do for given f and finite list S?

```
1 c ← −∞;
2 for x in S do
3 |   y ← f(x);
4 |   if y > c then
5 |   |   c ← y;
6 |   end
7 end
8 print c;
```

Lines 3–6 equivalent to $c \leftarrow \max\{c, f(x)\}$

---

1 $c \leftarrow -\infty$;
2 **for** x *in* S **do**
3 $\quad c \leftarrow \max\{c, f(x)\}$;
4 **end**
5 print c;

So what does it do?

Exercise: Modify algorithm to print maxi**mizer**

A very common structure is nested for loops

Here's a spell checker that checks each word in a file

```
1 for each line in file do
2 |   for each word in line do
3 |   |   check spelling of word;
4 |   end
5 end
```

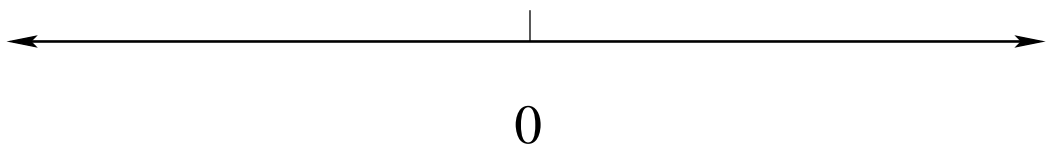
Here's another version of the spell checker

```
1 while another line exists do
2     read in line;
3     for each word in line do
4         if spelling of word is wrong then
5             offer correction;
6         end
7     end
8 end
```

Mathematical Preliminaries

Will often refer to the **real numbers**, $\mathbb{R}$

Understand it to contain "all of the numbers" on the "real line"



Contains both the rational and the irrational numbers

$\mathbb{R}$ is an example of a **set**

A set is a collection of objects viewed as a whole

(In case of $\mathbb{R}$, the objects are numbers)

Other examples of sets:

- set of all rectangles in the plane
- set of all prime numbers
- set of sets
- set of monkeys in Japan

Elementary probability is all about sets

E.g.,

- If A and B are independent, then $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$
- If A and B are disjoint, then $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B)$

You do remember what "disjoint" means, don't you?

Definitions are reviewed in appendix to course notes

Let A and B be sets

Statement $x \in A$ means that x is an element of A

$A \subset B$ means that any element of A is also an element of B

- E.g., irrationals are a subset of $\mathbb{R}$

$A = B$ means that A and B contain the same elements

- Equivalently, $A \subset B$ and $B \subset A$

Let S be a set and A and B be subsets of S

Union of A and B

$$A \cup B := \{x \in S : x \in A \text{ or } x \in B\}$$

Intersection of A and B

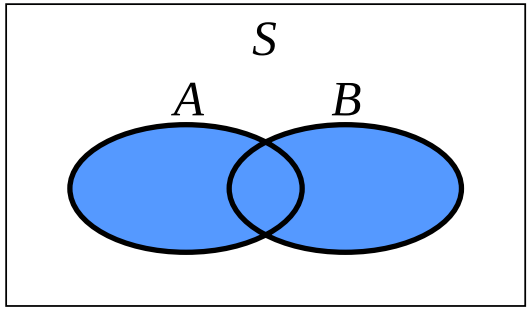
$$A \cap B := \{x \in S : x \in A \text{ and } x \in B\}$$

$A \setminus B$ is all points in A that are not points in B:

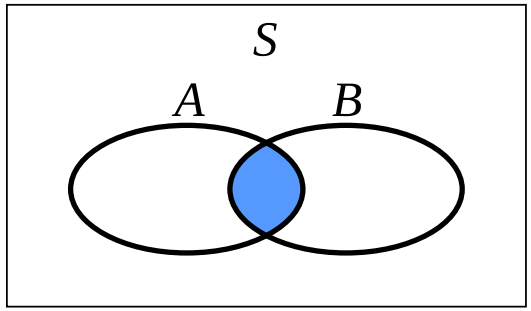
$$A \setminus B := \{x \in S : x \in A \text{ and } x \notin B\}$$

Complement of A is all elements of S that are not in A:

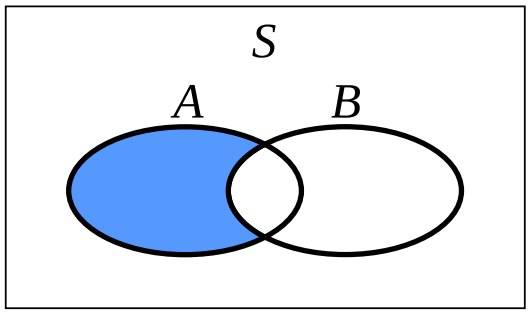
$$A^c := S \setminus A :=: \{x \in S : x \notin A\}$$



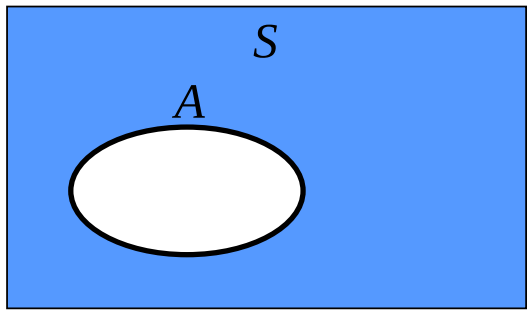
$A \cup B$



$A \cap B$



$A \setminus B$



$A^C := S \setminus A$

Figure: Unions, intersection and complements

The **empty set** $\emptyset$ is the set containing no elements

If $A \cap B = \emptyset$, then A and B said to be **disjoint**

Set operations:

If A and B subsets of S, then

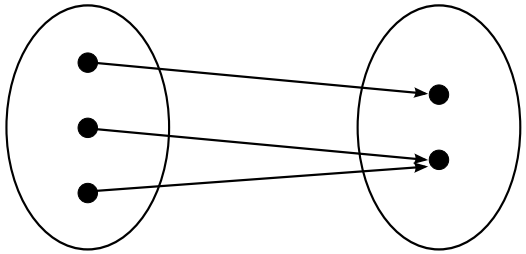
1. $A \cup B = B \cup A$ and $A \cap B = B \cap A$
2. $(A \cup B)^c = B^c \cap A^c$ and $(A \cap B)^c = B^c \cup A^c$
3. $A \setminus B = A \cap B^c$
4. $(A^c)^c = A$

Functions

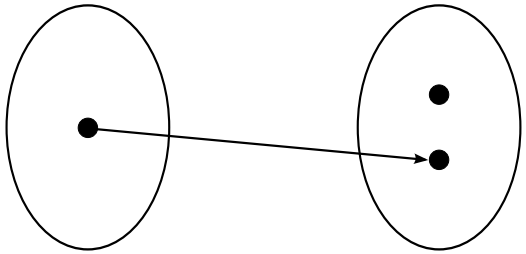
A **function** f from set A to set B is a rule that associates to each element x of A one and only one element of B

Notation:

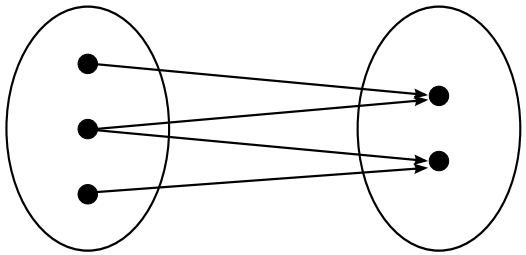
- $f: A \to B$ means that f is a function from A to B
- $f(x)$ called the **image of x under f**



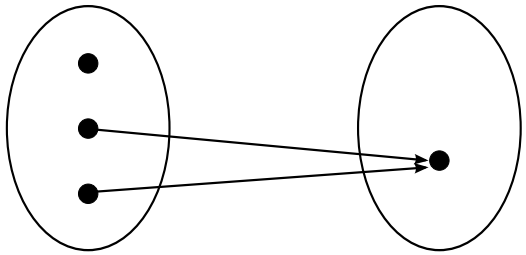
A function



A function



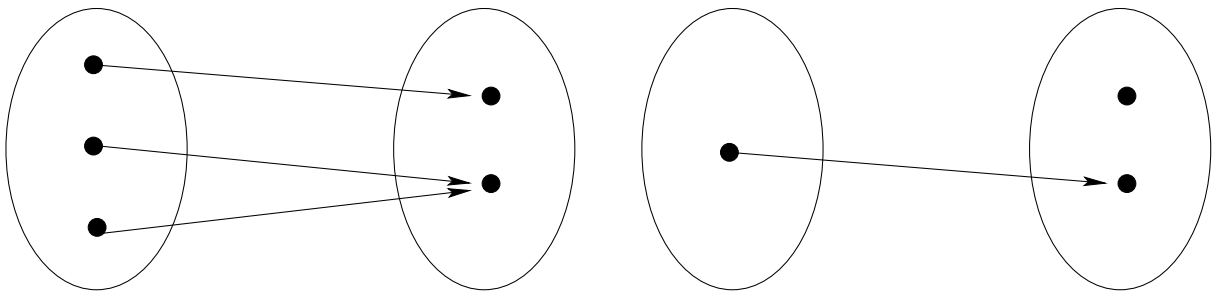
Not a function



Not a function

The **range** of $f\colon A \to B$ is the set

$$\operatorname{rng}(f) := \{y \in B : f(x) = y \text{ for some } x \in A\}$$



(a) $\operatorname{rng}(f) = B$ (b) $\operatorname{rng}(f) \neq B$

Let $\{x_n\}_{n=1}^{\infty}$ be a sequence in $\mathbb{R}$

Def: $x_n \to x$ means that x_n gets arbitrarily close to x as $n \to \infty$

(Formal def: see course notes)

A function $f : \mathbb{R} \to \mathbb{R}$ called **continuous at** x if

$$f(x_n) \to f(x) \text{ whenever } x_n \to x$$

The function called **continuous** if it is continuous at every $x \in \mathbb{R}$

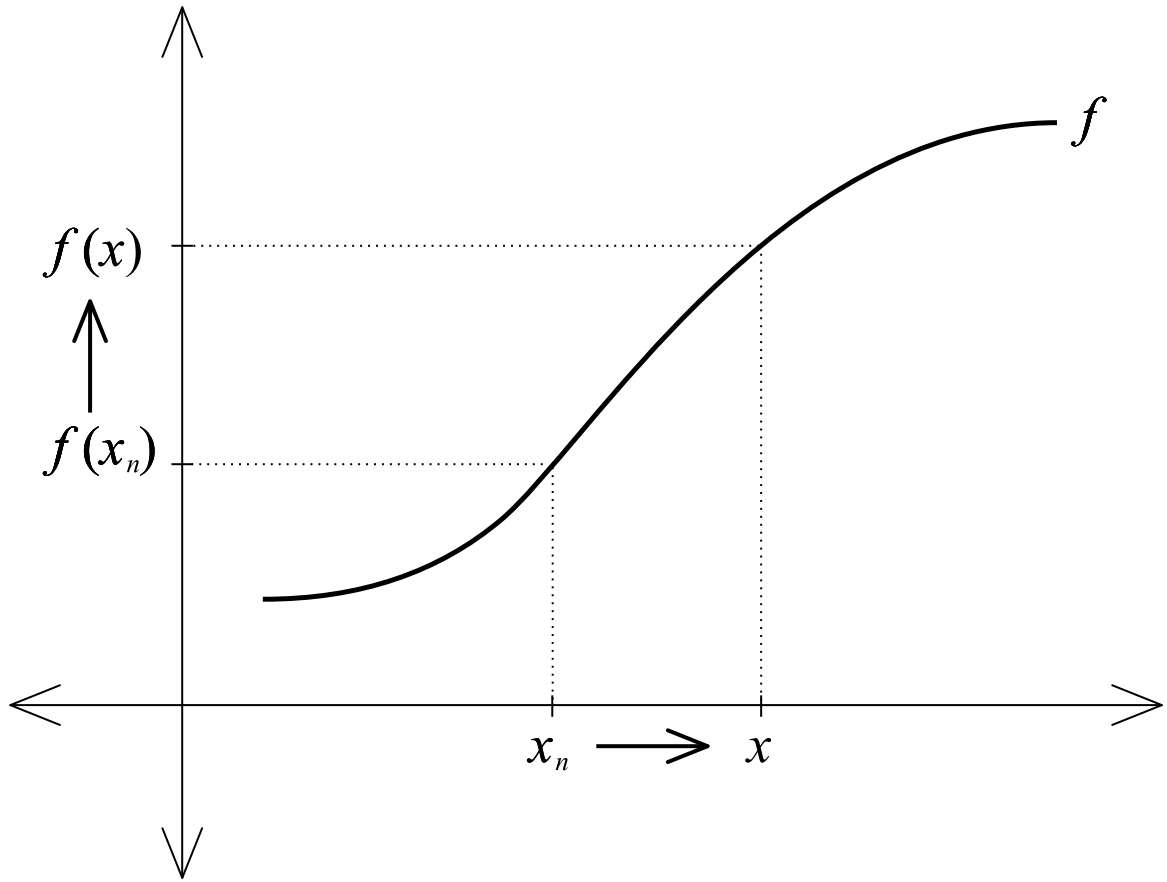


Figure: Continuity

We'll also deal with maximization of functions

Example: "Maximizing the likelihood function is the same as maximizing the log of the likelihood function."

Reason: Increasing transformations don't affect maximizers

What does this mean?

Let A be any set, and let $f : A \to \mathbb{R}$

A **maximizer** of f on A is a point $a^* \in A$ such that

$$f(a^*) \geq f(x) \text{ for all } x \in A$$

$m\colon \mathbb{R} \to \mathbb{R}$ called **increasing** if $m(x) \leq m(x')$ whenever $x \leq x'$

Example: $m(x) = x - 3$ is increasing

To see this, suppose that we have x and x' with $x \leq x'$

If this is true, then $x - 3 \leq x' - 3$ also true

That is, $m(x) \leq m(x')$

Let $g(x) = m(f(x))$ where m increasing

Claim: Any maximizer of f on A is also a maximizer of g on A

Proof: Let a^* be a maximizer of f on A

Let x be any element of A

Since a^* is a maximizer of f, we have $f(x) \leq f(a^*)$

But then $m(f(x)) \leq m(f(a^*))$ (why?)

In other words, $g(x) \leq g(a^*)$

Since x was arbitrary,

$$g(x) \leq g(a^*) \text{ for all } x \in A$$

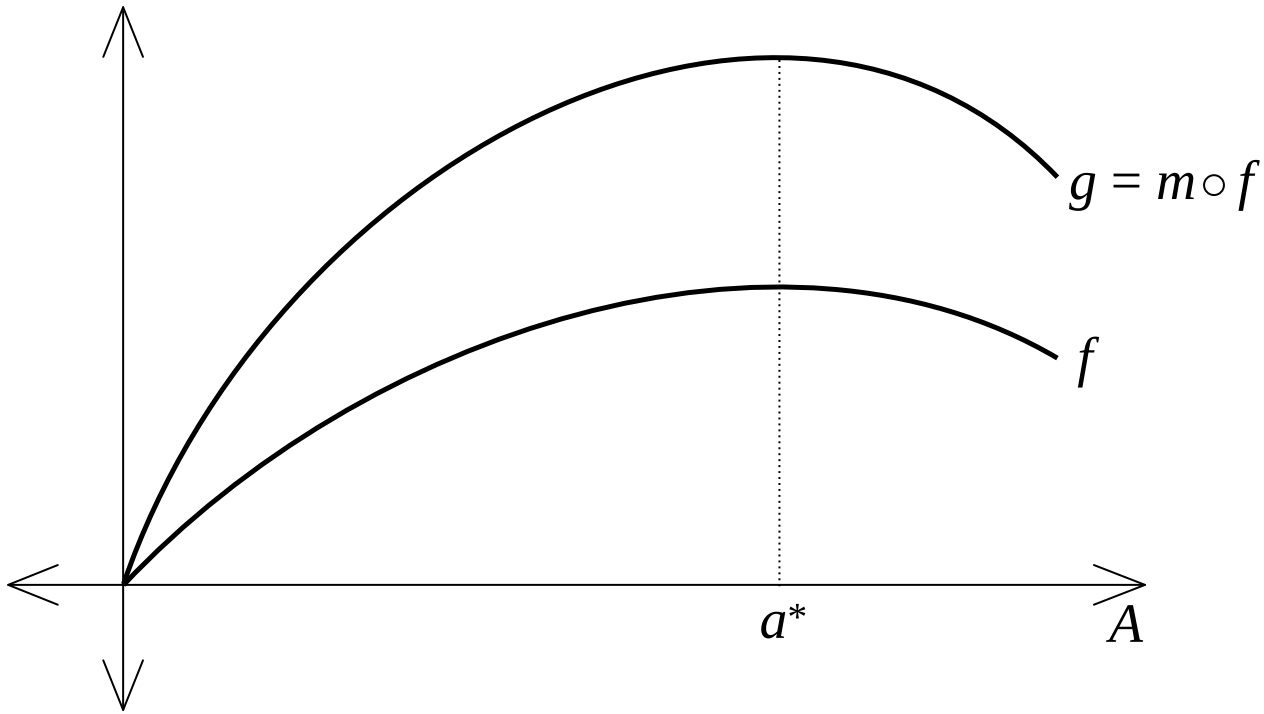


Figure: Increasing transforms preserve maximizers

A **minimizer** of f on A is a point $a^* \in A$ such that

$$f(a^*) \leq f(x) \text{ for all } x \in A$$

Exercise:

Let $f : A \to \mathbb{R}$, and let $g(x) = -f(x)$

Show: Any maximizer of f on A is also a minimizer of g on A

Homework for Week 1

Download the course notes from the homepage

- But don't print the whole thing!

Review maths in appendix of course notes

Install R—see course homepage

Once you start it, greeted with welcome message, prompt:

```
>
```

Try it out:

```
> 1 + 1
[1] 2
> print("foo")
[1] "foo"
```

Watch the R video tutorials listed on the course homepage

Replicate, experiment

Have a look at the R resource links on the course homepage

Remember: Computer labs start **next** week