Module scope

ST4060/ST6015 aims to provide a broad understanding of methodological and implementational aspects involved with current techniques used for **statistical learning**.

This includes reviewing various statistical concepts and techniques used in data exploration and analysis, methods for simulating statistical frameworks, and basic concepts of machine learning.

The course introduces programming in R, commencing at a basic level.

The objective is not so much to cover these techniques in depth, but rather to develop a sensitivity to different aspects and issues related to statistical analysis, from a practical angle.
Introduction

Importance of implementation skills

How is this useful to us at UCC?

- Stochastic modelling
- Regression & GLM
- Hypothesis testing

- Time Series Analysis
- Survival Analysis
- Projects (simulation work)...

### Linear Regression

\[
\begin{align*}
\text{y} & = 5 + 2 \cdot \text{x} \\
2 & \quad 4 & \quad 6 & \quad 8 & \quad 10 & \quad 15 & \quad 20 & \quad 25 & \quad 30
\end{align*}
\]

### Time Series

\[
\begin{align*}
\text{AirPassengers} & = 100 + 50 \cdot (\text{Time} - 1950) \\
\end{align*}
\]
Why should you be interested?

- Employers really like applied experience (quant jobs, ...)
- First in your new job: hit the ground running!

- Exploratory analysis
- Performance analysis
- Benchmarking
Main objectives

- Book of modules:
  - “simple non-parametric density and regression in R”;
  - “regularization, penalized likelihood, Bayesian estimation”;
  - computing “graduated estimates of mortality and survival”;
  - “using cross-validation and bootstrapping”;
  - “machine learning for model selection”

- Another important aim of this course is to help you become self-sufficient at R-based development. This includes learning to be organised and find helpful resources, but also building up self-confidence by exploring a variety of examples.
Course outline

1. R Basics and exploratory statistics
2. Parametric model fitting
   - Linear regression
   - Nonlinear models
   - Optimisation
   - Diagnostics
3. Smoothing / graduation
   - Regularization
   - NP regression
   - Curve estimation
   - Prediction
4. Importance sampling
   - Bootstrap, CV
   - Validation
   - NP CI’s
   - Simulation
5. Introduction to “Machine Learning” (Statistical Learning)
Course outline

1. R basics and simple statistics
   - R basics
   - Distribution models
   - Nonparametric density estimation

2. Parametric model fitting
   - Linear regression
   - Generalized linear models
   - Nonlinear regression
   - Regularization
   - Bayesian framework

3. Smoothing and graduation
   - Nonparametric regression
   - Splines
   - Graduation

4. Importance sampling
   - Monte-Carlo sampling
   - Bootstrapping
   - Markov Chain Monte Carlo
   - Cross-validation

5. Introduction to Machine Learning
   - Basic concepts
   - Learning framework
   - Performance assessment
   - Some techniques of reference
LearnR workouts (TBC)

(1) **Warm-up workout**
- Basic R, vectors, data frames, ...
- Plotting
- I/O (Input/Output)
- Functions
- Distribution models and density estimation

(2) **Regression workout**
- Linear model, diagnostics, simulation
- GLM
- Nonlinear regression models
- Optimization
- Ridge regression
- R scripting

(3) **Smoothing workout**
- Polynomial regression
- B-splines
- P-splines

(4) **Resampling workout**
- Monte Carlo
- Bootstrapping
- Cross-validation

(5) **Machine learning workout**
- Supervised vs Unsupervised...
- Classification vs Regression...
- Some classifiers
Using R...

```r
# Question 10: predict from P-spline
# split data and smooth early part:
# plotsmooth.spline(x,y)
# yhat = predict(smooth.spline(x,y))
# lines(yhat)

# Question II: real application... interpolation from spline
# Use the spotly life dataset...

data <- read.csv("data/lifedata.csv")

# male
male <- data[sex=="Male",]
age <- male$age
yhat <- predict(smooth.spline(~age), newdata=male)
lines(yhat)

# female
female <- data[sex=="Female",]
age <- female$age
yhat <- predict(smooth.spline(~age), newdata=female)
lines(yhat)
```

---

**USA 2000 male rates**

- **Description**
  - Fits a cubic smoothing spline to the supplied data.

- **Usage**
  - `smooth.spline(x, y = NULL, w = NULL, df = 5, sp = NULL, penalty = 1, n = NULL, df.knots, degree = 3, nknots, intercept = TRUE, tol = 1e-04)`

- **Arguments**
  - `x`: a vector giving the values of the predictor vari-
  - `y`: response. If `y` is missing or null, the response
  - `w`: optional vector of weights of the same length
  - `df`: the desired equivalent number of degrees of freedom
  - `sp`: smoothing parameter, typically (but not neces-
  - `penalty`: ordinary roughness or “generalized” cross-vali-
  - `nknots`: number of knots. If NULL, nknots = df - degree

```r
# Question 9: R-spline vs P-spline
# B-spline from a dataframe structure
# x = data[,c(1,3,5) ]
# y = log(x[,1])
# control = list(df=degree)
# P = smooth.spline(x,y,df=degree)
# B = smooth.spline(x,y,degree)

# Question 11: repeat Q11 but perform analysis on the log-rates...
# data <- read.csv("data/lifedata.csv")
# male <- data$maleMale, force
# age <- data$age
# yhat <- predict(smooth.spline(~age), newdata=male)
# lines(yhat)
```

Introduction

Course outline

... or RStudio...
Module timetable

**Teaching Period 1**
Wednesday 4pm in WGB_107
Thursday 8am in WGB_107
Friday 1pm in Kane_G01

**Assessment:** (Total Marks 100)
15 marks test/assignment #1 (details TBC), about mid-course
15 marks test/assignment #2 (details TBC), towards end of term
70 marks practical exam (i.e. lab-based), end of term

**R clinics (extra)**
1 or 2 one-hour sessions during term (so not every week)
Learn to start and manage an R session
Tuesday 10m in WGB_G33? (TBC)
1.0 - R basics and simple statistics
1.1 - R basics
R basics

- See appendix document for a brief recap of R basics
- So many resources available online... you could start with:
  - CRAN documents (manuals, contributed docs)
  - Hadley Wickham’s online website
  - Garrett Grolemund (RStudio)’s Twitter account
- Go through the ST4060 warm-up workout
1.2 - Distribution models
Distributions in R

- **Synopsis** (where `distr` abbreviates a chosen distribution):
  - `rdistr(n, ...)`: random generation of \( n \) realizations
  - `ddistr(x, ...)`: probability density function \( f(x) \)
  - `pdistr(q, ...)`: probability distribution function \( F(x) \)
  - `qdistr(p, ...)`: quantile function

- **Example with the Normal distribution:**
  - `rnorm(10)` randomly generates 10 realizations of \( \mathcal{N}(0, 1) \)
  - `dnorm(0) == 1/sqrt(2*pi)` should return `TRUE`
  - `pnorm(1.645)` returns \( 0.9500151 \)
  - `qnorm(0.95)` returns \( 1.644854 \)

- ?Distributions for a list of available distributions: `dbeta`, `dbinom`, `dexp`, `dpois`, `dlnorm`, `dunif`, `dt`, `dcauchy`, etc.

- **See also:**
  - [http://cran.r-project.org/web/views/Distributions.html](http://cran.r-project.org/web/views/Distributions.html)
Random generation

- **rnorm(2,1,3)** generates 2 realizations of $\mathcal{N}(\mu = 1, \sigma = 3)$
- The following loop:
  ```r
  par(mfrow=c(1,3))
  for(i in 1:3){v=rnorm(50); hist(v)}
  ```
  plots the histograms of 3 samples of 50 realizations of $\mathcal{N}(0, 1)$
- This is how new samples can be generated in a loop
- **set.seed(s)** may be used to fix pseudo-random generation:
  ```r
  set.seed(1); v=rnorm(10); set.seed(1); w=rnorm(10);
  ```
  (using seed $s=1$) will ensure that $v=w$
- This is particularly useful to **reproduce** simulated data exactly
Examining the distribution of a dataset

- `summary(object)` returns statistics for the input dataset:
  
  ```r
  R> summary( rnorm(10) )
  ```

  may return something like

  ```
  Min. 1st Qu. Median Mean 3rd Qu. Max.
  -1.56300 -0.30350 -0.11910 -0.01992 0.44800 1.10500
  ```

- `summary()` also returns a summary of the output of many fitting functions

- Plots: `hist()`, `stem()`, `density()`, `boxplot()`...

- QQ-plots: `qqnorm()`, `qqplot()`

- `shapiro.test()` (Shapiro-Wilk normality test)
A simple example: empirical CDF

```r
x <- rnorm(30)
F10 <- ecdf(x)
plot(F10, verticals= TRUE, do.p = FALSE, lwd=2)
curve(pnorm, from=-5, to=5, add=TRUE, col="gray70")
rug(x)  # plots the locations of x’s below the curve
```

![Empirical CDF diagram](image-url)
1.3 - Nonparametric density estimation
Nonparametric Statistics?

*Nonparametric* may mean that:

- the method does not assume an underlying distribution;
  - Ex: rank statistics
- the model is allowed to change with the data.
  - Ex: nonparametric regression

Some nonparametric methods are also sometimes termed *adaptive*, because they adapt to the underlying distribution of the data.
Nonparametric density estimation

- Estimate a density $f(x)$ with a smooth object $\hat{f}(x)$
- Use e.g. a Kernel Density Estimator (KDE)
- Consider sample points as “kernels”
- Centre a given “template” density of size $h$ at each kernel:

  $$K_h(u - x_i) = \frac{1}{h} K \left( \frac{u - x_i}{h} \right)$$

- Sum contributions of all these kernel densities:

  $$\hat{f}(u) = \frac{1}{N} \sum_{i=1}^{N} K_h(u - x_i)$$
Kernel Density Estimation

\[ x_i(*), \quad K \left( \frac{x-x_i}{h} \right), \quad \frac{1}{h} \sum_{i=1}^{N} K \left( \frac{x-x_i}{h} \right), \quad \frac{1}{N} \sum_{i=1}^{N} K_h(x - x_i) \]

(example with \( N = 8 \) points)
In R, estimate the pdf of a sample \( x \) by \( \text{density}(x) \)

Example:

```r
xs = rnorm(10)
fx = density(xs)
names(fx)
plot(fx)
points( density(xs, bw=2*fx$bw) )
lines(fx$x, fx$y, col='blue', lwd=2)
```
Kernel Density Estimation

Visualize influence of choice of bandwidth $h$:
kernel densities are in blue, final density estimate is in red...

\begin{align*}
h &= 0.065 \\
\text{Sample points} &\quad \text{KDE} \\
\text{Sample points} &\quad \text{KDE} \\
\end{align*}

\begin{align*}
h &= 0.13 \\
\text{Sample points} &\quad \text{KDE} \\
\text{Sample points} &\quad \text{KDE} \\
\end{align*}

(example with $N = 8$ points)
Other common nonparametric density estimators

- **Naive estimators using** $F_N(u) = \frac{1}{N} \sum_{i=1}^{N} 1_{X_i \leq u}$:

  $$f_N(u) = \frac{1}{Nh} \sum_{i=1}^{N} W \left( \frac{u - X_i}{h} \right)$$

  where $W(u) = \frac{\delta(|u|<1)}{2}$ is a rectangular kernel

- **$k$-th nearest neighbour estimators ($k \geq 2$)**

  $$\hat{f}_N(u) = \frac{k}{2Nd_k(u)} = \frac{1}{Nd_k(u)} \sum_{i=1}^{N} K \left( \frac{u - X_i}{d_k(u)} \right)$$

  where $d_k(u) = |u - X_{[k]}|$ are in increasing order, and $\int K = 1$
2 - Regression and estimation
Linear regression is a central tool of statistical analysis. It is used extensively and often is a key component within a more complex analytical procedure. The approach was notably extended to generalized linear modelling (GLM) and nonlinear regression, to address a large range of problems.

Regression analysis is critical to a large number of methodological aspects, including:

- trend estimation and prediction
- diagnostic tests for model selection
- benchmarking and (methodological) explorative analysis

In this section we review elementary notions related to the implementation of regression analysis.
2.1 - Linear regression
In linear regression models, observations depend linearly on the parameters of interest:

$$Y = \theta_0 + \theta_1 X + \varepsilon$$

where $\varepsilon$ traditionally represents a collection of realizations of a Gaussian r.v.

In this context, ordinary least squares are most often applied for estimation of $\theta = (\theta_0, \theta_1)$. When the error term $\varepsilon$ is not assumed Gaussian, maximum likelihood estimation is the more generic framework. Note that MLE boils down to maximising the OLS when $\varepsilon \sim N(\mu, \sigma)$. 
Parametric model fitting
Linear regression

M-estimation

M-estimation: an **objective function** \( \rho \) is applied to the **residuals**

\[
e(\tilde{\theta}_0, \tilde{\theta}_1) = Y - \tilde{Y} = Y - \tilde{\theta}_0 - \tilde{\theta}_1 X
\]

to form a criterion that gets optimised with respect to \((\theta_0, \theta_1)\):

\[
(\hat{\theta}_0, \hat{\theta}_1) = \arg \min_{\theta_0, \theta_1} \sum_{i=1}^{N} \rho(e_i(\theta_0, \theta_1))
\]

For example \( \rho(u) = u^2 \) and we optimise the sum of squares

\[
\sum_{i=1}^{N} (e_i(\theta_0, \theta_1))^2
\]

Any choice of a function \( \rho \) may be implemented and optimised.
Note that if \((\hat{\theta}_0, \hat{\theta}_1) \equiv (\theta_0, \theta_1)\) then the residuals are identical to the error term \(\varepsilon\).

Recall the main assumptions on the (traditional) linear model:

- the error term has zero mean and constant variance
- the errors are uncorrelated
- the errors are Normally distributed or sample size is large enough for large sample theory to be applied

This means that if the model fit (estimation) is good, the residuals should look like they satisfy these properties – although residuals are not independent r.v.’s.
Unless $N \gg p$, where $p$ is the number of parameters of interest, residuals should be adjusted (standardized).

Various techniques exist, among which:

- standardized residuals
- Studentized residuals
- Jacknife residuals
Standardized residuals:

\[ z_i = \frac{e_i}{\sqrt{MSE}} \approx N(0, 1) \]

where \( MSE = \sum_{i=1}^{N} (e_i)^2 / (n - p - 1) \).

Studentized residuals:

\[ r_i = \frac{e_i}{\sqrt{MSE(1 - h_i)}} \approx t(n - p - 1) \]

where \( h_i = \text{diag}[X(X^TX)^{-1}X^T]_i \) and \( 0 \leq h_i \leq 1 \).
Jacknife residuals:

$$r_{(-i)} = r_i \sqrt{\frac{MSE}{MSE_{(-i)}}} \approx t(n - p - 2)$$

where $MSE_{(-i)}$ is computed by leaving the $i$th observation out.

Jacknife residuals have mean $\approx 0$ and variance given by

$$\frac{1}{n - p - 2} \sum_{i=1}^{N} r^2_{(-i)}$$
Once estimates \((\hat{\theta}_0, \hat{\theta}_1)\) are obtained for \((\theta_0, \theta_1)\), **diagnostic checks** are performed to assess the model fit:

- Residual plots and scatterplots
- “Normal” and quantile plots (histogram, boxplot, density estimate, ...)
- Correlation tests (Portmanteau, turning points, ...)
- Examination of spectral density

When implementing a statistical analysis, such diagnostic tests **must** be incorporated in the methodology.
Linear regression via least squares is performed in R using `lm`:

- `lm(obs~regressor)` implements fitting of \( y = \beta_0 + \beta_1 x + \epsilon \)
- `lm(obs~regressor+0)` implements fitting of \( y = \beta_1 x + \epsilon \)
- Example: the following Least Squares estimation
  \[
  x = c(1:10); \quad y = 3*x + rnorm(10); \quad \text{lm}(y\sim x)
  \]
  returns the estimated model coefficients:
  \[
  \begin{array}{ll}
  \text{(Intercept)} & x \\
  -0.8818 & 3.0975 \\
  \end{array}
  \]
  i.e. \( \hat{\beta}_0 = -0.8818 \) and \( \hat{\beta}_1 = 3.0975 \)
One can capture the output of `lm`, inspect it, and use it:

- `out = lm(y ~ x)`  
  # capture output
- `summary(out)`  
  # inspect output
- `plot(out$residuals)`  
  # plot residuals from model fit
- `acf(out$residuals)`  
  # check these residuals!
- etc.
2.2 - Generalized linear models
Parametric model fitting
Generalized linear models

Generalised Linear Models

- GLM is an extension of the simple linear model
- Allows for categorical data: e.g. treatment effect on biomarker
- Ex: measure activity in diseased muscle by medical imaging
  - $B$ and $A$ resp. denote "Before" and "After"
  - $z$ is the observed activity in terms of image intensities
  - $x$ denotes time of imaging (baseline=”B”, after therapy=”A”)
  - $v$ contains voxel index: $v = (1 \ldots n)^T$
- In R: vectorize the problem into

$$
\begin{pmatrix}
  z_B \\
  z_A
\end{pmatrix}
= \alpha^T \begin{pmatrix}
  v \\
  v
\end{pmatrix}
+ \beta \begin{pmatrix}
  x_B \\
  x_A
\end{pmatrix}
+ \begin{pmatrix}
  \varepsilon_B \\
  \varepsilon_A
\end{pmatrix}
$$
In R: vectorize the problem into

\[
\begin{pmatrix}
    z_{B,1} \\
    \vdots \\
    z_{B,n} \\
    z_{A,1} \\
    \vdots \\
    z_{A,n}
\end{pmatrix}
= \begin{pmatrix} 1 \\ \vdots \\ n \end{pmatrix} \alpha^T + \begin{pmatrix} B \\ \vdots \\ A \end{pmatrix} \beta + \begin{pmatrix} \varepsilon_{B,1} \\ \vdots \\ \varepsilon_{A,n} \end{pmatrix}
\]

- \( \beta = \text{treatment effect} \);
- \( \alpha = (\alpha_1 \ldots \alpha_n)^T = \text{local effects} \)
- Implementation comparable to that of `lm`:

```r
x1 = rep(c(1:n),2)  
x2 = c(rep("B",n),rep("A",n))  # Before-After effect  
out = glm(c(zB,zA) ~ 0 + factor(x1) + factor(x2))
```
A summary of the output includes the coefficient estimates:

\[ \text{summary(out)} \text{$\$coef}$

\[ \text{summary(out)} \text{$\$coef[c(1:2*n),1]} = \alpha \]

\[ \text{summary(out)} \text{$\$coef[(2*n+1),1]} = \beta \]

In this example, it may be preferable to use a quasi-Poisson family since we analyze measured counts per voxel...

Various options can be changed, incl. assumed distribution:

\[
\text{glm}(c(zz) \sim 0 + \text{factor(x1)} + \text{factor(x2)}, \\
\text{family=quasipoisson()}, \text{epsilon=1e-8, maxit=25})
\]
2.3 - Nonlinear regression
Polynomial regression

There is a lot of flexibility (choice) in how one may tackle nonlinear relationships. Nonlinearity may be expressed with respect to the parameter of interest, the model covariate, or both.

**Polynomial regression** uses a model that is linear in the parameter of interest but nonlinear in the regression covariate:

\[ Y = \theta_0 + \theta_1 X + \theta_2 X^2 + \cdots + \theta_p X^p + \varepsilon \]

Goodness of fit depends on the order \( p \) of the polynomial (ideally, \( p = N - 1 \)), but polynomial regression is hard to interpret.

**NB:** \( Y = \theta^2 X + \varepsilon \) is actually \( Y = \beta X + \varepsilon \) with \( \beta = \theta^2 \in \mathbb{R}^+ \ldots \)
Polynomial regression

**Exercise:** `attach(pressure); x=temperature; y=pressure`
Implement the linear regression to obtain the following plot:
Nonlinear regression (using least squares)

Nonlinear regression allows the model structure (i.e. conditional expectation \( E(Y \mid X) \) of the observations \( Y \)) to depend nonlinearly on the parameters of interest.

Examples:

\[
Y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \varepsilon \\
Y = e^{\theta X} + \varepsilon
\]

Nonlinear regression via Least Squares is performed in R using \texttt{nls}.

\textbf{NB:} nonlinear methods are usually (very) sensitive to initialisation.
Parametric model fitting

Nonlinear regression

Example: 2-parameter exponential model (LearnR workout)

par(font.lab= 2, font.axis=2)
plot(x, y, pch=20, cex=2,
     xlab="Temperature (Celsius)",
     ylab="Vapor pressure (ml of mercury)",
     main="Example: polynomial regression")

nlreg <- nls(y~exp(a+b*x), start=list(a=0,b=.5))
    # error... need for better initialisation
nlreg <- nls(y~exp(a+b*x), start=list(a=0,b=.02))
summary(nlreg)
cc = coef(nlreg)
curve(exp(cc[1]+cc[2]*x), col='blue', add=T, lwd=3)
points(x, exp(0+0.018*x), col='red', t='b', lwd=3, cex=.5)

nlreg2 <- nls(y~exp(a*x), start=list(a=.02))
cc = coef(nlreg2)
curve(exp(cc[1]*x), col='cyan', add=T, lwd=3)
legend("topleft", col=c('black','blue','red','cyan'), lwd=3,
        legend=c("Data","Fitted (p=2)","Guessed","Fitted (p=1)"))
Example: nonlinear regression

**Temperature (Celsius)**

**Vapor pressure (ml of mercury)**

- Data
- Fitted (p=2)
- Guessed
- Fitted (p=1)
Example: US population (LearnR workout)

library(car)
data(USPop)
attach(USPop)
par(font.lab= 2, font.axis=2)

plot(year, population, cex=1.5, pch=20,
     main="Example: nonlinear regression (USPop data)"
)
time <- 0:21

pop.mod <- nls(population ~ beta1/(1 + exp(beta2 + beta3*time)),
                start=list(beta1 = 350, beta2 = 4.5, beta3 = -0.3),
                trace=TRUE)

summary(pop.mod)
lines(year, fitted.values(pop.mod), lwd=3, col=’red’)

Example: nonlinear regression (USpop data)
More details on NLS may be found e.g. at:
http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-nonlinear-regression.pdf
If one wants to implement another criterion, optim and optimx become useful for (general purpose) optimization. (The optimx package extends former optim.)

`optim` implements Nelder-Mead, quasi-Newton and conjugate-gradient methods, and includes the possibility of box constraints.

It takes definitions of the cost function (estimation criterion) and (optionally) its gradient as arguments, along with starting values. This allows users to define their own estimating functions.
Example: optim **(LearnR workout)**

```r
par(mfrow=c(1,2),font.lab= 2, font.axis=2)
model <- function(x, theta){
  return(theta*x)
}
crit <- function(theta,x,y){
  # must have theta as 1st parameter and return a single value...
  return( sum( y-model(x,theta) ) )
}

thbar = 1.8
x = rep(c(1,3,7,8), len=100)
y = model(x,thbar) + rnorm(x)

plot(x, y, pch=20, cex=1.5, main="optim example 1: data")
points(x,model(x,thbar),col='green',pch=20,cex=2)

(optim.out <- optim(par=c(1), fn=crit, x=x, y=y,
  method="L-BFGS-B", lower=c(0.01), upper=c(3.05)))
```

**Exercise:** Why does it not “work” ??????
Example: optim (continued)

```r
crit <- function(theta,x,y){
  # must have theta as 1st parameter and return a single value...
  return( sum( (y-model(x,theta))^2 ) )
}
thbar = 1.8
x = rep(c(1,3,7,8), len=100)
y = model(x,thbar) + rnorm(x)
(optim.out <- optim(par=c(1), fn=crit, x=x, y=y, method="L-BFGS-B",
                   lower=c(0.01), upper=c(3.05)))

# Visualize objective shape and estimated optimum:
plot(x, y, pch=20, cex=1.5, main="optim example 2: data")
points(x, model(x,thbar), col='green', pch=20, cex=2)
ths = seq(0,5,len=50)
obj = 0*ths
for(i in 1:length(ths)){ obj[i] = crit(ths[i],x,y) }

plot(ths, obj, main="Criterion...", xlab="theta",
     ylab="Objective value")
points(optim.out$par, crit(optim.out$par,x,y),col='red',pch=20,cex=3)
```
Example: optim (continued again)

```r
model.f <- function(x, theta){
  return(theta*x)
}
model.g <- function(x, theta){
  gradient of model.f...
  return(x)
}
crit.f <- function(theta,x,y){
  return( sum( (y-model.f(x,theta))^2 ) )
}
crit.g <- function(theta,x,y){
  # gradient of crit.f...
  return( sum( -2*model.g(x,theta)*(y-model.f(x,theta)) ) )
}

thbar = 1.8
x = rep(c(1,3,7,8), len=100)
y = model(x,thbar) + rnorm(x)

(optim.out <- optim(par=c(1), fn=crit.f, gr=crit.g, x=x, y=y,
  method="L-BFGS-B", lower=c(0.01), upper=c(3.05)))
```
Exercise: implement a simulation of the nonlinear model

\[ Y = \exp(-\theta X) + \varepsilon \]

and minimize the sum of least squares for this model using `optim`.

Exercise: implement the same simulation and minimize the sum of least squares for this model using `optimx`, this time.
2.4 - Regularization
Ridge regression (shrinkage)

- Add an $L_2$ penalty on to the regression cost function
- *Shrinks* the effect of some variables
- Output coefficient estimates depend on the choice of tuning parameter $\lambda$ and minimise the criterion

$$\sum_{i=1}^{N} \left( Y_i - \beta_0 - \sum_{j=1}^{p} \beta_j X_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = RSS + \lambda \sum_{j=1}^{p} \beta_j^2$$

- $\lambda$ is usually selected by cross-validation
- Shrinkage is not applied to the intercept parameter
- The final model includes all $p$ covariates
Add an $L_1$ penalty on to the regression cost function

- Allows to effectively “get rid of” some variables
- Output coefficient estimates depend on the choice of tuning parameter $\lambda$ and minimise the criterion

$$\sum_{i=1}^{N} \left( Y_i - \beta_0 - \sum_{j=1}^{p} \beta_j X_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

- $\lambda$ is usually selected by cross-validation
- Shrinkage is not applied to the intercept parameter
- The final model may have less than $p$ covariates
- “Least Absolute Selection and Shrinkage Operator”
2.5 - Bayesian framework
Bayesian inference

- **Frequentist**: the data are random and the model parameters are deterministic unknowns
- **Bayesian**: the data are fixed and the model parameters are random variables
- Recall Bayes theorem:

\[
f(x \mid y) = \frac{f(y \mid x)f(x)}{f(y)} \propto f(y \mid x)f(x)
\]

- **Inference** uses prior and posterior distributions to update how much is known about model parameters:

\[
p(\theta \mid Y) \propto f(Y \mid \theta) \pi(\theta) \quad \text{(posterior = likelihood x prior)}
\]

- Uses: EM algorithm, Kalman filtering (state-space modelling), Gibbs sampling, real-time, credibility, ...
Loss functions and risk functions

- Squared error loss: \( L_{SEL}(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2 \)
- Absolute error loss: \( L_{AEL}(\theta, \hat{\theta}) = |\theta - \hat{\theta}| \)
- Other loss functions: \( L(\theta, \hat{\theta}) = \lambda(\theta) |\theta - \hat{\theta}|^k, k = 1, 2, \ldots \)

NB: These loss functions are random variables, since they are functions of r.v.’s.

- Risk function: \( R(\hat{\theta}) = E_{\theta|x} \left( L(\theta, \hat{\theta}) \right) = \int_{\Omega} L(\theta, \hat{\theta}) f_{\theta, \hat{\theta}}(\theta, \hat{\theta}) d\theta \)
- \( \hat{\theta}^* = \arg\min_{\theta \in \Omega} R(\hat{\theta}) \)
- Bayes estimate wrt the SEL is the posterior mean
- Bayes estimate wrt the AEL is the posterior median
EM algorithm (Expectation-Maximization) consists in iteratively maximising an updated likelihood.

Iterative 2-step procedure:

repeat

1. update conditional expectation of objective function
2. maximise criterion wrt parameters of interest

until (some form of) convergence

Famously slow, but often reliable in finding global optimum.

Usual workhorse procedure e.g. for machine learning (clustering, classification, ...)

3 - Smoothing and graduation
3.1 - Nonparametric regression
Nonparametric regression model

Parametric regression models are usually of the form

\[ Y_i = g(\theta, X_i) + \varepsilon_i, \quad i = 1, \ldots, N \]

where \( g \) and the \( X_i \)'s are specified (i.e. “known”), and the additive noise term characterized, e.g. \( \varepsilon \sim iid \mathcal{N}(0, \sigma^2) \). The problem is then to estimate \( \theta \).

Nonparametric regression consists in leaving \( g \) unspecified; the problem becomes that of estimating \( g \) in

\[ Y_i = g(X_i) + \varepsilon_i, \quad i = 1, \ldots, N \]
Various forms may be considered for \( g(x) \), such as additive regression models

\[
g(x) = \alpha + g_1(x_1) + g_2(x_2) + \cdots + f_k(x_k) \\
g(x) = \alpha + \theta_1 x_1 + g_2(x_2) + \cdots + f_k(x_k) \\
g(x) = \alpha + g_{12}(x_1, x_2) + g_3(x_3) + \cdots + f_k(x_k)
\]

\( g \) is often assumed to be continuous and smooth.

Local polynomial regression, smoothing splines and (Nadaraya-Watson) kernel regression are among the most popular “smoothing” techniques available to estimate the shape of \( g \).
Consider the simple regression problem of estimating $g$ in

$$Y_i = g(X_i) + \varepsilon_i$$

A $p$-th order weighted least squares (WLS) polynomial regression of observations $y = (y_1, \ldots, y_N)^T$ on design points $x = (x_1, \ldots, x_N)$ evaluated at a given point $x_0$ yields the approximation

$$y_i^0 = \alpha + \theta_1(x_i - x_0) + \theta_2(x_i - x_0)^2 + \cdots + \theta_p(x_i - x_0)^p$$
Using WLS, the weights are usually chosen to account for the distance between $y_i$ and the reference design point $x_0$, using e.g. the tricube function

$$W(z) = \begin{cases} 
(1 - |z|^3)^3 & \text{for } |z| < 1 \\
0 & \text{for } |z| \geq 1 
\end{cases}$$

with $z_i^0 = \frac{x_i - x_0}{h}$ and $h$ is a control parameter that is picked according to the scale of the data.
library(car)  
attach(Prestige)
plot(income, prestige, xlab="Average income", ylab="Prestige")
lines(lowess(income, prestige, f=0.25, iter=0), lwd=2, col='blue')
lines(lowess(income, prestige, f=0.5, iter=0), lwd=2, col='red')
lines(lowess(income, prestige, f=1, iter=0), lwd=2, col='green')

degree of smoothing: too jittery
degree of smoothing: about right
degree of smoothing: too smooth
weight control: “do not refit to address outliers”  (ref [Cox2002])
Local polynomial regression: multiple regression

Consider the **multiple regression problem** of estimating \( g \) in

\[
Y_i = g(X_{i1}, \ldots, X_{ik}) + \varepsilon_i
\]

Local polynomial regression consists in fitting a weighted polynomial, e.g. of the linear form at a given point \( x_0 = (x_{01}, \ldots, x_{0k}) \):

\[
y_i^0 = \alpha + \theta_1(x_{i1} - x_{01}) + \theta_2(x_{i2} - x_{02})^2 + \cdots + \theta_k(x_{ik} - x_{0k})^p + \varepsilon_i
\]

This approach requires choosing a distance metric and a weighting scheme for multivariate observations.
The default distance metric is commonly the Euclidean distance

\[ D(x_i, x_0) = \left[ \sum_{j=1}^{k} (z_{ij} - z_{0j})^2 \right] \]

using standardized values \( z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j} \) (with sample mean and std dev)

The weights are usually defined using scaled distances between \( y_i \) and the reference point \( x_0 \), e.g.

\[ w_i = W \left( \frac{D(x_i, x_0)}{h} \right) \]

where \( h \) denotes the bandwidth (e.g. the half-width of the neighbourhood).
mod.lo = loess(prestige~income+education, 
                span=.5, degree=2)
summary(mod.lo)

# plot this smoothing...
inc <- seq(min(income), max(income), len=25)
ed <- seq(min(education), max(education), len=25)
newdata <- expand.grid(income=inc,education=ed)
fit.prestige <- matrix(predict(mod.lo, newdata), 25, 25)
persp(inc, ed, fit.prestige, theta=45, phi=30, 
      ticktype='detailed', expand=2/3, shade=0.5 
      xlab='Income', ylab='Education', zlab='Prestige')
Nadaraya-Watson regression

In this approach one likes to represent the regression problem based on

\[ g(x) = \mathbb{E}(Y_i \mid X_i = x) \]

and to evaluate the importance of the neighbourhood \( x \pm h \) around each design point \( x \) (for \( h > 0 \)).

Then the average of observations \( Y_i \)'s yields the following nonparametric estimator for \( g \):

\[
\hat{g}(x) = \frac{\sum_{i=1}^{N} \mathbb{1}(|X_i - x| \leq h) Y_i}{\mathbb{1}(|X_i - x| \leq h)} = \frac{\sum_{i=1}^{N} K \left( \frac{X_i - x}{h} \right) Y_i}{K \left( \frac{X_i - x}{h} \right)}
\]

using Uniform kernels \( K \).
Note that the estimator may also be defined by plugging $\hat{f}(x, y)$ into

$$g(x) = \frac{\int y f(x, y) dy}{f(x)}$$

This approach extends to the choice of other kernels; the Normal distribution is (once again) very commonly used.

The density estimator used should be well-defined (i.e. $\hat{f}(x) > 0$) for the kernel regression estimator to be well defined.

Kernel regression extends easily to multivariate problems by using multivariate kernel constructs (this is beyond our scope).
library(car)
attach(Prestige)

plot(income, prestige)
inc.100 <- seq(min(income), max(income), len=100)

mod.lo.inc <- loess(prestige ~ income, span=.7, degree=1)
pres <- predict(mod.lo.inc, data.frame(income=inc.100))
# This other implementation works better:
library(KernSmooth)
lp = locpoly(income, prestige, bandwidth=1500)
lines(lp, lwd=2, col='red')
SMOOTHING AND GRADUATION

NONPARAMETRIC REGRESSION

Car example

```r
with(cars, {

plot(cars$speed, cars$dist)
ex2 = ksmooth(cars$speed, cars$dist,
             "normal", bandwidth = 2)
ex5 = ksmooth(cars$speed, cars$dist,
             "normal", bandwidth = 5)

lines(ex2, lwd=2, col = 2)
lines(ex5, lwd=2, col = 3)
})
```
Smooth control!?

At this point one ought to wonder: how do we set (or select) the value for $h$?

This is a very common question for most nonparametric methods (cf. KDE’s!)...

Typical approaches are:

- simulation study (using plotting and trial-and-error)
- some type of criterion optimization (usually some likelihood or MSE function)
- cross-validation (typically based on the MSE)

The choice of strategy usually depends on the problem at hand (and on the user’s taste!)...
3.2 - Splines
Splines

- Spline = continuous function (i.e. curve) constructed by piecewise linkage of polynomials/functions
- Free splines (Bezier curves): not suitable for mortality curves
- Regression splines: use equidistant knots
- Smoothing splines: penalize roughness, knots are data points

[6] Bronze spline weights
B-splines

- Explain $f(x)$ in terms of a basis ($B = 'basis'$)
- A B-spline is defined by its order $m$ and number of interior knots $K$ ($x_0$ and $x_{K+1}$ are end-knots)

\[ x_0 \leq x_1 \leq \cdots \leq x_K \leq x_{K+1} \]

- Polynomial is of order $m - 1$ (one often picks $m = 4$)
- Univariate construction:

\[ S(x) = \alpha_0 + \sum_{j=1}^{J} \alpha_j B_j(x, m) \]

where $J = K - 1 + m$ is the number of basis functions
- Basis elements are defined in a recursive manner [cf. lit.]
- Fitting: basis coefficients are calculated via linear regression
B-splines: specifications

- Placing knots can be achieved in a number of ways
- One way is to define regular intervals within \([x_{\text{min}}, x_{\text{max}}]\)
- Another way is to define the interior knots as the quantiles from the empirical distribution of the underlying variable
  - enforces an equal number of observations in each interval
  - intervals have different lengths
B-splines: specifications

The basis elements (or design matrix) are evaluated at the control points for the specified number of knots. Ex: compare

```r
require(splines)
bs(dat$Age)
matplot(dat$Age,BM,xlab='Age')
```

with

```r
xs = seq(min(dat$Age),max(dat$Age),length=1000)
ks = quantile(xs,seq(0.1,0.9,by=.1))
(BM2 = bs(dat$Age,knots=ks))
matplot(dat$Age,BM2,xlab='Age',t='b')
abline(v=ks,lty=2,col=8)
```
Choice of design matrix (Life dataset)
Choice of design matrix (Life dataset)
B-spline questions

Question (LearnR workout)

(a) Compute the design matrix for the Life dataset, and quote the projection of age 40 onto that basis (cf. figure below).

(b) Reconstruct the B-spline approximation to the crude male force data using linear regression. Plot the output over the data points and criticize.
B-spline questions

Question  

(a) Continuing from the previous question, compute the B-spline as a power series, i.e. compute the traditional cubic spline representation

\[ S(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{j=1}^{J} \alpha_j (x - x_j)^{3+} \]

where \((x - x_j)^{3+} = \max((x - x_j)^3, 0)\), using \(J = 3\) interior knots at 40, 60 and 80.

(b) Compare with the fitted values from the regression onto the design matrix obtained from `bs()` for those knots.


**Splines** are piece-wise polynomial functions of the underlying target \( \text{“response function”} \). The pieces are joined at knots \( \{X_1, \ldots, X_N\} \) to yield a continuous functional.

A simple smoothing spline \( \hat{g} \) is defined as the function that **minimizes** the penalized sum of squares

\[
SS_h(g) = \sum_{i=1}^{N} (Y_i - g(X_i))^2 + h \int_{X_{\text{min}}}^{X_{\text{max}}} (g''(u))^2 \, du
\]

defined over a given grid \( [X_{\text{min}}, \ldots, X_{\text{max}}] \).
Smoothing splines

\[ SS_h(g) = \sum_{i=1}^{N} (Y_i - g(X_i))^2 + h \int_{X_{\text{min}}}^{X_{\text{max}}} (g''(u))^2 \, du \]

- P-splines use all data points as knots and penalize for smoothness
- Penalty term is usually controlled by one parameter \( h \)
- This smoothing parameter \( h \) is *controlled*
- Can also define periodic splines
Smoothing and graduation
Splines

\[ SS_h(g) = \sum_{i=1}^{N} (Y_i - g(X_i))^2 + h \int_{X_{min}}^{X_{max}} (g''(u))^2 \, du \]

- The solution (which defines the smoothing spline)

\[ \hat{g} = \arg \min_g SS_h(g) \]

is a cubic spline with knots located at the design points \( X_1, \ldots, X_N \).

- The cubic spline approach assumes the existence of two continuous derivatives, which defines a roughness penalty in the integral term: the larger \( g'' \), the more jittery \( g \).
library(car)
attach(Prestige)

plot(income, prestige)
inc.100 <- seq(min(income), max(income), len=100)

mod.lo.inc <- loess(prestige ~ income, span=.7, degree=1)
pres <- predict(mod.lo.inc, data.frame(income=inc.100))

ssp <- smooth.spline(income, prestige, df=3.85)

lines(inc.100, pres, lty=2, lwd=2)
lines(ssp, lwd=2, col='blue')
There are many, many implementations of nonparametric regression methods in R, including

- `lowess` for simple regression, i.e. \( g(x) = g_1(x_1) \)
- `loess` for local polynomial regression
- `smooth.spline` for fitting simple regression smoothing splines
- `ksmooth` for Nadaraya-Watson kernel regression
- the `sm` library for local regression, local likelihood and pdf estimation
- the `gss` library for generalized smoothing splines and regression models
- `locfit`, `gam` and `mgcv` also come up regularly
3.3 - Graduation
Dealing with rough data (simulated life data, $D=585$)

Crude force of mortality (Males)

<table>
<thead>
<tr>
<th>Age (years)</th>
<th>Force of mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>0.00</td>
</tr>
<tr>
<td>40</td>
<td>0.05</td>
</tr>
<tr>
<td>50</td>
<td>0.10</td>
</tr>
<tr>
<td>60</td>
<td>0.15</td>
</tr>
<tr>
<td>70</td>
<td>0.20</td>
</tr>
<tr>
<td>80</td>
<td>0.25</td>
</tr>
<tr>
<td>90</td>
<td></td>
</tr>
</tbody>
</table>
Or with unbalanced information (Irish Males, 2005-07)
Fitting mortality rates

**Objective:** to fit a curve of mortality rates from either dense information (e.g. life table from population census) or small, patchy sample (e.g. late-life characteristics for pensioners)

**Steps:**
- Obtain observed mortality rates
- Fit some functional model to it
- Forecast late-life, age-derived features, or future generations

**Challenges:**
- High variability due to small sample information
- Missing information for specific ages
- Parametric models maybe ill-suited (late-life)
Some standard smoothers

- Running mean / median
- Linear time-invariant filters
- Kernel smoothing
- Loess / polynomial regression
- Splines
- ... and parametric models?
Smoothing and graduation

Graduation

Running mean/median

- Write out an algorithmic formula for the running mean
- What calibration must be done? What must be controlled?
- What advantage(s)/inconvenient(s) do these methods have?

Crude force of mortality (Males)

Age (years)

Force of mortality

Empirical
5-MA
9-MA
10-MA
Running mean/median

cf. example on 2000-2010 USA mortality rates... (LearnR workout)
Linear time-invariant filters

- Algorithmic formula for an LTI?
- What calibration must be done? What must be controlled?
- What advantage(s)/inconvenient(s) does this method have?
Kernel smoothing

- Write out an algorithmic formula for a kernel smoother
- What calibration must be done? What must be controlled?
- What advantage(s)/inconvenient(s) do these methods have?
Smoothing and graduation

Polynomial regression

- Write out an algorithmic formula for a polynomial smoother
- What calibration must be done? What must be controlled?
- What advantage(s)/inconvenient(s) do these methods have?

Question – cf. (LearnR workout):

1. Apply lowess() to both the USA 2000 log-mortality rates and the (non-log) crude mortality rates from the Life dataset, and visualize the outputs. Adjust the smoother span to successively 2/3, 1/3 and 0.1, and compare outputs.

2. Compute the MSE for each smooth, for both datasets.

3. Based on the above, describe the main steps for a grid-search approach for selecting an appropriate value for the span.
Assessment of graduation

Typical issues:
- Lack of goodness-of-fit, either overall or at specific locations
- Consistent bias
- Inconsistency in overall shape

Comparing smoothing procedures: use tests, e.g. $\chi^2$-test
Assessment of graduation / comparisons

χ²-test

- \( \mathcal{H}_0 \): graduated estimates \( \{\mu_x\} \) are true
- Assumes independent numbers of deaths at different ages
- Assuming \( D_x \sim \mathcal{N}(E_x^c \mu_x^0, E_x^c \mu_x^0) \) (Poisson-Normal approx.),

Actual deaths - Expected deaths = \( D_x - E_x^c \mu_x^0 \)

\( (E_x^c \) is the central exposed-to-risk at age \( x \) nearest birthday)

\[
z_x = \frac{D_x - E_x^c \mu_x^0}{\sqrt{E_x^c \mu_x^0}} \sim \mathcal{N}(0, 1) \quad (\text{under } \mathcal{H}_0)
\]

and

\[
\xi = \sum_{i=1}^{m} z_{x_i}^2 \sim \chi^2_m, \quad m = \text{number of age groups}
\]
Plenty methods yielding smooth curves... 

- Choice of method should be data-driven (and user-driven?)
- Parametric modelling is a form of smoothing
- Ex: fit Makeham model to a rough mortality curve...
- Exponential smoothing (cf. time series) is another adaptive procedure
- Weighted functional polynomials
- Etc.
Mortality rates projections

- IoFA: in-house method for mortality projection
- Cf. implementation of 2009 model for mortality rates proj.
- Office Nat. Stat. population mortality data
- Use of age-cohort P-spline early on in processing step
- Recent CMI documents [7] illustrate concerns that current projection methods are not statistical in nature
- Lee-Carter typically used for projection of smoothed data
4.0 - Importance sampling
With the increase in computational power, resampling techniques have invaded the scientific world in the last few decades. They are useful in a number of ways, for implementing simulations as well as analysing statistical properties of a technique applied to real data.

Resampling methods treat an observed sample as a finite population and sample randomly from it in order to estimate the underlying population’s characteristics and make inferences about the population. They allow us to quantify uncertainty by calculating standard errors and confidence intervals and performing significance tests. They require fewer assumptions than traditional parametric methods of statistical inference.
Importance sampling

Main concepts

- **Monte-Carlo** methods are generally used on data that are simulated rather than observed. Repetitions of the simulated statistical experiment are performed; the experiment outputs are collected and then analysed as outcomes of a random variable. This way, finite-sample properties can be derived for the technique used in the experiment.

- There may be circumstances when we cannot know whether the underlying population is indeed Normal or when we cannot take a large enough sample size for the Central Limit Theory to apply. **Bootstrapping** is a way of tackling these problems by finding the sampling distribution, at least approximately, from just one sample.
4.1 - Monte-Carlo sampling
Monte-Carlo methods

Monte Carlo methods use resampling techniques to estimate the distribution of a population.

They rely on the law of large numbers, to allow us to approximate an expectation by the sample mean of a random variable (or a function thereof).

For example, risk analysis may be performed by MC repetitions of a simulated model outcome. Each outcome is generated for a different set of random values from the model distribution. The (sample) distribution of outcome values can then be used as a basis for assessing (i.e. approximating) the theoretic risk and/or functionals of it.
Monte-Carlo integration

Let our objective be to compute

$$\theta = \int_a^b g(x) dx$$

for a given function $g$, assuming this integral exists. Recall that if $X$ is a r.v. with density $f(x)$, then

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x) dx$$

Then if a random sample is available from $F(X)$, an unbiased estimator of $E[g(X)]$ is the sample mean

$$\bar{g}_M(X) = \frac{1}{M} \sum_{i=1}^{M} g(X_i)$$
This provides an algorithm for the evaluation of integrals.

**Example:** we wish to evaluate

\[ \theta = \int_0^1 g(x) \, dx \]

If \( X_1, \ldots, X_M \) is a random \( U(0, 1) \) sample, then by the strong LLN,

\[ \hat{\theta} = \frac{1}{M} \sum_{i=1}^{M} g(X_i) \]

converges to \( \theta = E[g(X)] \) (since \( X \sim U(0, 1) \)) with probability 1.
Generalisation to any interval: in order to evaluate

$$\theta = \int_{a}^{b} g(x) \, dx$$

we can perform a change of variables to fall back to the $\mathcal{U}(0, 1)$ case, OR ELSE simply sample from a $\mathcal{U}(a, b)$:

$$\hat{\theta} = (b - a) g_M(X) = \frac{(b - a)}{M} \sum_{i=1}^{M} g(X_i)$$

Exercise: evaluate $e^{-x}$ over $(2, 4)$ and check with the theoretic value.
**Variance:** we have

\[
\text{Var} \left( \overline{g_M(X)} \right) = \frac{1}{M} \text{Var} \left( g(X) \right)
\]

and thus

\[
\text{Var} \left( \hat{\theta} \right) = (b - a)^2 \text{Var} \left( \overline{g_M(X)} \right) = \frac{(b - a)^2}{M} \text{Var} \left( g(X) \right)
\]

For \( M \) large, the Central Limit Theorem (generally) applies and one may assume that \( \hat{\theta} \) is approximately Normally distributed. This is useful for deriving confidence intervals and for selection of \( M \).

**Efficiency:** given two estimators, \( \hat{\theta}_1 \) is more efficient than \( \hat{\theta}_2 \) if \( \text{Var}(\hat{\theta}_1) < \text{Var}(\hat{\theta}_2) \)
Generalization of the principle

Now suppose we want to evaluate the unknown functional $g$ of $X$, a r.v. with any density $f(x)$,

$$E[g(X)] = \int_{-\infty}^{\infty} g(x) f(x) \, dx$$

The same approach is applicable, using a random sample from the distribution of $X$:

$$\hat{\theta} = \frac{1}{M} \sum_{i=1}^{M} g(X_i)$$

Then with probability 1,

$$\hat{\theta} \xrightarrow{M \to \infty} E[\hat{\theta}] = \theta$$
Note: **Importance sampling** is an extension of the principle of Monte Carlo simulations. It consists in choosing an adequate sampling distribution (from which random realizations are generated) in order to approximate the expected value with “good” control on the estimation variance.
Monte-Carlo repetitions of a statistical experiment

Suppose we need to evaluate a functional from observations

\[ Y_i = \eta(\beta, X_i) + \varepsilon_i, \quad i = 1, \ldots, N \]

where:

- \( \eta \) and \( X_i \in \mathcal{X} \) are known
- \( \beta \) describes a finite number of parameters to be estimated
- \( \varepsilon_i \overset{iid}{\sim} f_\varepsilon(0, \sigma^2) \)

The Monte Carlo principle can be applied to estimate

\[ E[Y] = \eta(\beta, X) \]
Example: consider estimating $\beta$ in a linear regression model

$$Y_i = \beta X_i + \varepsilon_i, \quad i = 1, \ldots, N$$

with deterministic $X_i \in \mathcal{X}$ and $\varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, 1)$... Then

$$f_{Y|\beta, X} = \mathcal{N}(\beta X, 1)$$

1. Analyse a synthetic statistical problem with Monte-Carlo repetitions of the experiment...
2. This implies generating $M$ random samples of size $N$ for the same model
3. The statistical solution is derived for each repetition and stored in an adequate R object
4. This yields an empirical distribution for the estimator $\hat{\beta}$
1 Assume a true value for $\beta$, e.g. $\beta = 3$:

$$Y_i = 3X_i + \varepsilon_i, \quad i = 1, \ldots, N$$

and regressors $X_i$, e.g. $X = (1, 2, 3, 4, 5)$

2 Set experiment parameters, e.g. $N = 50$ and $M = 100$

3 Implement repetitions of the set experiment using a loop:

```r
x = rep(c(1:5), 10); # init
ests = matrix(0, M, 1) # init
for(i in 1:M){
  y = 3*x + rnorm(N)
  ests[i] = lm(y ~ x+0)$coef[1]
}
```
Analyse the distribution of estimates (stored in `ests`):

\begin{itemize}
\item `summary(ests)`
\item `hist(ests)`
\item `boxplot(ests)`
\item `mean(ests)`
\item `sd(ests)`
\item ...
\end{itemize}

- This allows comparing several estimators in terms of their distributions.
- One could for example compare the distributions of the LS estimator and of a robust alternative in a linear regression problem with non-Gaussian noise (hint: `?MASS::rlm` and `?rexp`)
4.2 - Bootstrapping
Boostrapping

One often refers to Monte Carlo repetitions as *parametric bootstrapping*, since in that case the distribution is known and parametrized.

The Bootstrap has been introduced by Bradley Efron in the late 1970’s.

Bootstrap methods are often used when the target population is not specified and the sample is the only available information to us. The distribution of the finite population represented by the sample can be regarded as a pseudo-population with similar characteristics as the true population.
Importance sampling

Bootstrapping

By repeatedly generating random samples from the pseudo-population (resampling) the sampling distribution of a statistic can be estimated. Properties of an estimator such as bias, standard error etc can then be estimated.

Bootstrap estimates of a sampling distribution are analogous to the idea of density estimation. We construct a histogram of a sample to obtain an estimate of the shape of the density function. The histogram is not the density but can be viewed as a reasonable estimate of it.
General idea of the Bootstrap procedure

1. Create lots of new samples (Bootstrap samples) by sampling from the original random sample with replacement. All samples have same size $N$ as original.

2. Calculate the statistic in question for each one of the Bootstrap samples.

→ The distribution of these statistics is called a Bootstrap distribution.

→ It will give information about the shape, centre, and spread of the sampling distribution of the statistic.
General algorithm

Given an initial sample $X_1, \ldots, X_N$, assuming we are interested in estimating parameter $\theta$,

1. Repeat the following steps for $b = 1, \ldots, B$:
   1. Draw a sample $X^{*(b)}$ from $X$ with replacement
   2. Evaluate the estimate $\hat{\theta}^{(b)}$ from $X^{*(b)}$

2. Deduce the bootstrap estimate of $F_{\hat{\theta}}$ as the empirical distribution of replicates $\hat{\theta}^{(1)}, \ldots, \hat{\theta}^{(B)}$

Example: Bootstrap standard error estimate

Bootstrap SE estimate for estimator $\hat{\theta}$ (e.g., $\hat{\theta}_b = \bar{x}_b$):

$$SE_B = \sqrt{\frac{1}{B - 1} \sum_{b=1}^{B} \left( \hat{\theta}_b - \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}_b \right)^2}$$
Common bootstrap statistics

Given:
- an initial sample \( \{X_1, \ldots, X_N\} \) with initial statistic \( \hat{\theta} \),
- \( B \) bootstrap estimates \( \{\hat{\theta}_b\}^B_{b=1} \) of parameter of interest \( \theta \)
  (e.g. \( \hat{\theta} = \bar{X} \) and \( \hat{\theta}_b = \bar{X}_b^* \)):

  - **Bias:** \( B\text{Bias}(\hat{\theta}) = \frac{1}{B} \sum_{b=1}^{B} (\hat{\theta}_b - \hat{\theta}) \)
  - **Variance:** \( \hat{\text{Var}}(\hat{\theta}) = \frac{1}{B} \sum_{b=1}^{B} (\hat{\theta}_b - B\text{Bias}(\hat{\theta}))^2 \)
  - **MSE:** \( \hat{\text{MSE}}(\hat{\theta}) = B\text{Bias}(\hat{\theta})^2 + \hat{\text{Var}}(\hat{\theta}) \)
  - **Standard error:**

\[
SE_B = \sqrt{\frac{1}{B - 1} \sum_{b=1}^{B} \left( \hat{\theta}_b - \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}_b \right)^2}
\]
Importance sampling
Bootstrapping

Accolade: International prize for the Bootstrap

“With the bootstrap […], scientists are able to learn from limited data in a simple way that enables them to assess the uncertainty of their findings. In essence, it is possible to simulate a potentially infinite number of data sets from an original data set and—in looking at the differences—measure the uncertainty of the result from the original data analysis.”

“”While statistics offers no magic pill for quantitative scientific investigations, the bootstrap is the best statistical pain reliever ever produced”, says Xiao-Li Meng, Whipple V. N. Jones Professor of Statistics at Harvard University. ”It has saved countless scientists and researchers the headache of finding a way to assess uncertainty in complex problems by providing a simple and practical way to do so in many seemingly hopeless situations.””

“”Because the bootstrap is easy for a computer to calculate and is applicable in an exceptionally wide range of situations, the method has found use in many fields of science, technology, medicine and public affairs”, says Sir David Cox […].”
Accolade: International prize for the Bootstrap

The impact of the bootstrap across research fields as measured by citation

The dataset contains over 200,000 articles from over 200 journals between 1980 and 2018

<table>
<thead>
<tr>
<th>Field</th>
<th>Citations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agricultural and Biological Sciences</td>
<td>44,476</td>
</tr>
<tr>
<td>Arts and Humanities</td>
<td>6,585</td>
</tr>
<tr>
<td>Biochemistry, Genetics and Molecular Biology</td>
<td>29,136</td>
</tr>
<tr>
<td>Business, Management and Accounting</td>
<td>8,248</td>
</tr>
<tr>
<td>Chemical Engineering</td>
<td>1,823</td>
</tr>
<tr>
<td>Chemistry</td>
<td>3,734</td>
</tr>
<tr>
<td>Computer Science</td>
<td>33,177</td>
</tr>
<tr>
<td>Decision Sciences</td>
<td>11,843</td>
</tr>
<tr>
<td>Dentistry</td>
<td>71</td>
</tr>
<tr>
<td>Earth and Planetary Sciences</td>
<td>9,744</td>
</tr>
<tr>
<td>Economics, Econometrics and Finance</td>
<td>12,088</td>
</tr>
<tr>
<td>Energy</td>
<td>2,814</td>
</tr>
<tr>
<td>Engineering</td>
<td>21,106</td>
</tr>
<tr>
<td>Environmental Science</td>
<td>14,663</td>
</tr>
<tr>
<td>Health Professions</td>
<td>2,105</td>
</tr>
<tr>
<td>Immunology and Microbiology</td>
<td>12,605</td>
</tr>
<tr>
<td>Materials Science</td>
<td>3,407</td>
</tr>
<tr>
<td>Mathematics</td>
<td>30,072</td>
</tr>
<tr>
<td>Medicine</td>
<td>31,022</td>
</tr>
<tr>
<td>Multidisciplinary</td>
<td>108</td>
</tr>
<tr>
<td>Neuroscience</td>
<td>6,827</td>
</tr>
<tr>
<td>Nursing</td>
<td>1,033</td>
</tr>
<tr>
<td>Pharmacology, Toxicology and Pharmaceutics</td>
<td>3,331</td>
</tr>
<tr>
<td>Physics and Astronomy</td>
<td>13,071</td>
</tr>
<tr>
<td>Psychology</td>
<td>9,152</td>
</tr>
<tr>
<td>Social Sciences</td>
<td>15,461</td>
</tr>
<tr>
<td>Veterinary</td>
<td>926</td>
</tr>
</tbody>
</table>

Courtesy of Cornell University and EPAM Systems Inc.
4.3 - Markov Chain Monte Carlo
MC simulations are particularly useful for multi-dimensional problems or when the problem structure or geometry is complex - although MC methods introduce a stochastic error.

**Markov Chains Monte Carlo (MCMC)** are a particularly popular technique which uses the previous sample value to randomly generate a sample value (which constitutes a Markov chain).

A large range of algorithms have been proposed to implement Markov Chain Monte Carlo analyses.
The **Metropolis algorithm** is one such example. It is one of the simplest, and earliest attempts.

The **Metropolis-Hastings algorithm** is a generalization of Metropolis. It may be used when the simulation framework requires sampling from more “complicated” distributions. In fact it was proposed by physicists to compute the integral of difficult functions by random sampling.

**Gibbs sampling** is another particularly famous MCMC method (in fact a particular case of Metropolis-Hastings sampling that uses univariate distributions).

The independence sampler and the random walk are other popular MCMC methods.
Importance sampling

Markov Chain Monte Carlo

**Metropolis-Hastings: general principle**

The objective is to generate a Markov chain \( \{X_t \mid t = 0, 1, 2, \ldots\} \) such that the target distribution is its stationary distribution.

- Initialize \( X_0 \) according to some prior knowledge
- At time \( t \), generate a candidate \( Y \) from a proposal (or jumping) distribution \( g(\cdot \mid X_t) \)
- Set acceptance of \( Y \) with probability

\[
\alpha(X_t, Y) = \min \left(1, \frac{f(Y)g(X_t \mid Y)}{f(X_t)g(Y \mid X_t)} \right)
\]

\((g(B \mid A) \text{ describes the likelihood of jumping from } A \text{ to } B)\)

- Set \( X_{t+1} = Y \) with probability \( \alpha \), otherwise \( X_{t+1} = X_t \)
Metropolis-Hastings sampler

1. Choose a proposal distribution $g(\cdot \mid X_t)$
2. Generate $X_0$ from a distribution $g$
3. Repeat until convergence:
   1. Generate $Y$ from $g(\cdot \mid X_t)$
   2. Generate $U$ from $U(0,1)$
   3. If
      
      $$U \leq \min \left(1, \frac{f(Y)g(X_t \mid Y)}{f(X_t)g(Y \mid X_t)} \right)$$

      accept $Y$ and set $X_{t+1} = Y$, otherwise set $X_{t+1} = X_t$
4. Set $t = t + 1$
The original Metropolis sampler

The Metropolis-Hastings sampler is a generalization of the Metropolis algorithm. The latter requires a symmetric proposal distribution, i.e. such that

$$g(Y \mid X) = g(X \mid Y)$$

The original Metropolis sampler thus sets the acceptance probability as

$$\alpha = \min \left( 1, \frac{f(Y)}{f(X_t)} \right)$$

Note that both methods assume that the target distribution satisfies usual regularity conditions.
MCMC: practical implementation issues

- How many iterations for burn-in sequence?
- How many iterations after it (for inference)?
- Run one long chain or several smaller ones?
- Starting point determination?
- ...

4.4 - Cross-validation
Cross-validation (CV)

- A single error/performance measurement is not reliable
- Resampling techniques such as Cross-validation or Bootstrapping are usually used to obtain a more stable assessment of performance
- Cross-validation (CV) may be:
  - Leave-one-out CV
  - Leave-k-out CV
  - k-fold CV
  - Monte Carlo CV
Importance sampling
Cross-validation

CV: general framework (source: ISLR)

\[
CV(n) = \frac{1}{n} \sum_{i=1}^{n} MSE_i
\]

(figure from ISLR)
Cross-validated error is the average of all LOO CV sample MSE’s (or any other performance criterion):

\[ CV(n) = \frac{1}{n} \sum_{i=1}^{n} MSE_i \]
k-fold CV (source: ISLR)

\[ CV(n) = \frac{1}{n} \sum_{i=1}^{n} MSE_i \]

(figure from ISLR)
Monte Carlo CV

Instead of splitting into distinct folds (as in k-fold CV), split dataset randomly at every iteration (using same training and test sample sizes every time)

```r
N = nrow(x)  # x is the dataset of predictors, y the response variable
M = 100
err = numeric(M)
set.seed(1)
for(m in 1:M){
    # split set randomly every time, e.g. 70%-30%:
    itrain = sample(c(1:N),round(.70*N))
    x.train = x[itrain,]
    x.test = x[-itrain,]
    y.train = y[itrain]
    y.test = y[-itrain]
    lmo = lm(y.train~x.train)  # train
    err[m] = mean((predict(lmo,newdata=x.test)-y.test)^2)  # test + store
}
```
5.0 - Introduction to Machine Learning
5.1 - Basic concepts
“Machine learning”? 

Essentially, it is statistical learning

Essentially, we’re looking for a pattern (unseen up to now)

If there is no pattern, then ML will be counter-productive as it is likely to produce one!

Assumes availability of relevant data

Examples: consumer taste/habits, online advertising, election forecasts, risk prediction

Warning: linear regression is now referred to as “artificial intelligence” by a lot of people
Variable of interest: \textbf{Wage} (continuous)

\textbf{Figure 1.1.} Wage data, which contains income survey information for males from the central Atlantic region of the United States. Left: wage as a function of age. On average, wage increases with age until about 60 years of age, at which point it begins to decline. Center: wage as a function of year. There is a slow but steady increase of approximately $10,000 in the average wage between 2003 and 2009. Right: Boxplots displaying wage as a function of education, with 1 indicating the lowest level (no high school diploma) and 5 the highest level (an advanced graduate degree). On average, wage increases with the level of education.
Statistical learning: classification

- Variable of interest: **Default** (categorical)

**FIGURE 4.1.** The Default data set. Left: The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue. Center: Boxplots of balance as a function of default status. Right: Boxplots of income as a function of default status.

(figure from ISLR)
Statistical learning: supervised learning

- Variable of interest: \textbf{Species} (categorical)
- With prior experience available (\textit{supervised classification})

Iris dataset

```
<table>
<thead>
<tr>
<th>Sepal.Length</th>
<th>Sepal.Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>2.0</td>
</tr>
<tr>
<td>5.0</td>
<td>2.5</td>
</tr>
<tr>
<td>5.5</td>
<td>3.0</td>
</tr>
<tr>
<td>6.0</td>
<td>3.5</td>
</tr>
<tr>
<td>6.5</td>
<td>4.0</td>
</tr>
<tr>
<td>7.0</td>
<td>4.5</td>
</tr>
<tr>
<td>7.5</td>
<td>5.0</td>
</tr>
<tr>
<td>8.0</td>
<td>5.5</td>
</tr>
</tbody>
</table>
```

![Iris dataset scatter plot](image)
Statistical learning: unsupervised learning

- Variable of interest: **Species** (categorical)
- With no prior experience available (**clustering**)
5.2 - Learning framework
Data splitting

- A dataset is randomly split into a *training set* and a *test set*
- The training set is used to fit (or train) the model
- The test set is used to validate this model
- Model is both tuned and fitted during training
- We expect $MSE_{\text{train}} < MSE_{\text{test}}$
- **Overfitting** occurs when too much emphasis is put on training the data:
  - Training set yields a *much* smaller MSE than test set
  - Model is describing the training data too well and unable to adapt to new data
  - Yields poor prediction performance
General framework with large data

Example in model selection:

- Random split into training, validation and test sets (e.g. 50%, 25%, 25% resp.)
- Training set: used to fit the model
- Validation set: used to measure prediction error and choose best model
- Test set: used to measure generalization error of final model (i.e. ability to predict from new data)

[The Elements of Statistical Learning, T. Hastie, R. Tibshirani, J. Friedman, Springer]
General framework with small data

Cross-validation is usually used when dealing with small samples.

Example: model selection for classification with $N = 50, p = 5000$

1. Randomly divide samples into $K$ cross-validation folds
2. For each fold $k = 1, 2, \ldots, K$:
   a. Find a subset of predictors with higher (univariate) correlation with class labels, using all data except fold $k$
   b. Using just this subset of predictors, build a multivariate classifier, using all data except fold $k$
   c. Use the classifier to predict the class labels for fold $k$ and compute corresponding prediction error

[The Elements of Statistical Learning, T. Hastie, R. Tibshirani, J. Friedman, Springer]
General framework: summary

- Randomly split training and test set

Training set:
- Model calibration (tuning)
- Fit model on whole training set

Test set:
- Model validation (test)
Key aspects of the data (typical challenges)

- **Data scales**
  - Beware of scale effects in heterogeneous data
  - Illustrative example: *iris* dataset

- **Dimensionality (too many covariates)**
  - Many variables may be “aligned” / redundant
  - Data pre-filtering: must be done independently of class labels
  - Illustrative example: *iris* dataset

- **Dimensionality (too many dimensions in the observed data)**
  - Pre-process data using dimension reduction techniques
  - Factor analysis, PCA are predominant and common choices
  - Illustrative examples: *iris* and *EuStockMarkets* datasets
5.3 - Performance assessment
Performance indicators for regression

- **Mean Square Error (MSE):**
  \[
  MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{f}(X_i))^2
  \]

- **LOO CV test MSE:**
  \[
  MSE_{(n)} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{f}^{-i}(X_i))^2
  \]

- **k-fold CV test MSE:**
  \[
  MSE_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i^{-i}
  \]
Performance indicators for classification

- Prediction accuracy (error rate)

\[ Err = \frac{1}{n} \sum_{i=1}^{n} I(Y_i \neq \hat{Y}_i) \]

- LOO CV error rate:

\[ Err(n) = \frac{1}{n} \sum_{i=1}^{n} I(Y_i \neq \hat{Y}_i^{-i}) \]

- k-fold CV test MSE:

\[ Err(k) = \frac{1}{k} \sum_{i=1}^{k} Err_i^{-i} \]
Performance indicators for classification

**Recall:** one seeks to retain or reject a null hypothesis $H_0$ on the basis of evidence. Let us denote $H_1$ the alternative hypothesis.

<table>
<thead>
<tr>
<th></th>
<th>$H_0$ is true</th>
<th>$H_1$ is true</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_0$ is accepted</td>
<td>Correct decision</td>
<td>Type II error</td>
</tr>
<tr>
<td>$H_1$ is accepted</td>
<td>Type I error</td>
<td>Correct decision</td>
</tr>
</tbody>
</table>

- The Null hypothesis *can never be proven*
- Type I error occurs when $H_0$ is true but rejected
- $P$(Type I error) = *significance level* of the test
- $P$(Type II error) = *false negative rate*
- $1 - P$(Type II error) = *statistical power* of the test (*sensitivity*)
Performance indicators for classification

- False Positive rate = \( \frac{FP}{N} \) = specificity (I)
- True Positive rate = \( \frac{TP}{N} \) = sensitivity = recall (I-II)
- AUC of the ROC

<table>
<thead>
<tr>
<th>Actual value</th>
<th>Prediction outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TOTAL</th>
<th>P</th>
<th>N</th>
</tr>
</thead>
</table>

ROC for some dataset

Sensitivity vs. Specificity

AUC = 0.7314
5.4 - Some techniques of reference
Logistic regression

- Use a scrambled subsample \( x \) of the iris dataset:
  \[
  \text{is} = \text{sample}(1:150); \ x = \text{iris}[\text{is}[1:100],]
  \]

- Recode \( x \)$Species into \text{is.virginica} with values in \((0; 1)\)
  (we’re changing the problem formulation slightly)

- Fit model:
  \[
  \text{fit} \leftarrow \text{glm(Species} \sim ., \text{data}=x, \\
  \text{family}=\text{binomial(logit)})
  \]

- Use fit to predict Species of remaining data points:
  \[
  \text{testset} = \text{iris}[-\text{is},]
  \]
  \[
  \text{y} \leftarrow \text{testset}[,1:4]
  \]
  \[
  \text{pred} \leftarrow \text{predict(fit, newdata=y, type='response')}
  \]

- Assess prediction performance
Naive Bayes classification

- Recall: find unknown true label $l_i$ for each observation $Y_i$, given the observed predictor vector $x_0$
- Naive Bayes Classifier: for all $i = 1, \ldots, n$

$$\hat{l}_i = \arg \max_j Pr(Y_i = j|X_i = x_0)$$

- Example: for a two-class problem,

$$\hat{l}_i = 1 \text{ if } Pr(Y = 1|X = x_0) > 0.5$$
Regression and Classification Trees

Ex: ISLR::Hitters (http://www-bcf.usc.edu/~gareth/ISL)

FIGURE 8.1. For the Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to $\text{Years}<4.5$, and the right-hand branch corresponds to $\text{Years}\geq4.5$. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.
Regression and Classification Trees

Ex: ISLR::Hitters
Clustering

- Idea: arrange $n$ individuals into groups wrt a set of measures
- The choice of measure is key and determines classification
- Variables should be rescaled first (and weighted)
- Highly dimensional data may require prior reducing (PCA not necessarily pertinent here)
- [http://cran.r-project.org/web/views/Cluster.html](http://cran.r-project.org/web/views/Cluster.html)
Hierarchical clustering

Hierarchical clustering: split-and-merge to construct a dendrogram

```r
M = matrix(c(0,3,5,8,4,
             3,0,2,6,8,
             5,2,0,3,4,
             8,6,3,0,1,
             4,8,4,1,0), nrow=5)
dM = data.frame(M, row.names=c("A","B","C","D","E"))
dmat = dist(dM)
plclust( hclust(dmat) )
```
Hierarchical clustering

Cluster Dendrogram

```
dmat
hclust (*, "complete")
```

```
Height
D E A B C
4 6 8 10 12
```
Hierarchical clustering

- Example: `data(eurodist)`
- Creating a dendrogram:
  ```r
  hc = hclust(eurodist, method="ward")
  ```
- Plotting dendrograms:
  ```r
  plot(hc)  # (or plclust)
  plot(hc, hang=-1)
  rect.hclust(hc, k=3)
  ```
**k**-means clustering

- **k**-means clustering is another popular clustering method
- It is comparable to an Expectation-Maximisation algorithm
- \( ?\text{kmeans} \ldots \)
- Initialize at \( k \) clusters (use hierarchical to choose \( k \)?)
- Move an individual to another cluster if criterion is optimized
- Risk of convergence to local solution
Principal component analysis

- Project data according to highest variance components
- Linear orthogonal transformation:
  each component is uncorrelated with preceding ones
- Reveals internal structure of the data, explaining its variance
- PC1 = greatest variance by any projection of the data
- Theoretically optimal transform for a given data in LS terms
- PCA is widely used e.g. to reduce problem dimensionality
- `prcomp`...
Principal component analysis

Example: ?USArrests

- Statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973
- Also contains the % of the population living in urban areas

```r
plot(USArrests, main="USArrests data") # scatterplots

pairs(USArrests, panel = panel.smooth, main = "USArrests data")
```
Introduction to Machine Learning

Some techniques of reference

Principal component analysis

USArrests data

- Murder
- Assault
- UrbanPop
- Rape
Principal component analysis

data(USArrests)

 cor(x) # correlation matrix
eigen(cor(x)) # eigenvalue decomposition

# compare result with prcomp:
prcomp(USArrests, scale = TRUE)  # same vectors!

prcomp(~Murder+Assault+Rape, data=USArrests, scale=T)
plot(prcomp(USArrests, scale = TRUE))
# equiv.  cov(prcomp(USArrests, scale = TRUE)$x)
# i.e.  (prcomp(USArrests, scale = TRUE)$sdev)^2,
# i.e.  the eigenvalues of the cov/correl matrix

summary(prcomp(USArrests, scale = TRUE))
Some techniques of reference

Principal component analysis

prcomp(USArrests, scale = TRUE)

Variances

0.0 0.5 1.0 1.5 2.0
References

1. An Introduction to Statistical Learning (with Applications in R), Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, 8th Edition, Springer 2017
4. An introduction to R, the R Development Core Team, Version 2.15.1, CRAN website, 2012
References for section 3 (splines/smoothing)

7. The Future of the CMI Mortality Projections Model, CMI Mortality Projections Committee, 26 Oct 2015
Comments/feedback?

For any comments or queries about this document, please contact eric.w@ucc.ie